

1 Prezentarea metodologiei de lucru cu plăcile didactice

1.1 Obiective

Sunt enumerate resursele necesare pentru activitatea de laborator. Se prezintă noțiunile de bază în modelarea circuitelor numerice. Se studiază metodologia de lucru cu plăcile de dezvoltare precum și familiarizarea cu mediul de dezvoltare Xilinx ISE pentru programarea circuitelor logice programabile de tip FPGA (Field Programmable Gate Array). Se prezintă proiectul de lucru care conține biblioteca de componente TTL (Transistor-Transistor Logic) studiate și se implementează circuite simple folosind porți logice fundamentale. Se testează funcționarea acestora cu ajutorul tabelelor de adevăr corespunzătoare.

1.2 Resurse necesare


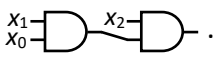
Pentru proiectarea și testarea circuitelor studiate sunt utilizate următoarele elemente:

- Placa de dezvoltare Nexys A7, Manualul de utilizare [1];
- Mediul de dezvoltare Xilinx ISE WebPACK 14.7 [2]; se va lucra în cadrul unui proiect existent, care pune la dispoziție o librărie cu circuitele TTL uzuale.

1.3 Elemente de bază în modelarea circuitelor numerice



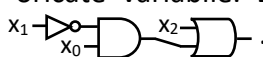
Modelarea comportamentului circuitelor numerice se realizează folosind un formalism matematic care are la bază conceptele din *algebra booleană* (după numele matematicianului George Boole [3]). Aceasta conține un set de operații logice și proprietăți ale acestora cu ajutorul cărora se poate descrie funcționalitatea circuitelor utilizate în proiectarea logică. Un astfel de circuit are unul sau mai multe intrări și ieșiri, denumite *terminale* (sau pini). Fiecare ieșire poate fi modelată ca o funcție de mai multe variabile. Variabilele sunt intrările circuitului. Deoarece în algebra booleană funcțiile și variabilele pot să aibă doar valorile 1 (adevărat) și 0 (fals) acestea se mai numesc și *funcții booleene*. Din punct de vedere electronic valorile sunt asociate cu proprietăți electrice precum tensiunea. De exemplu, pentru circuitele realizate cu tranzistoare în tehnologie Transistor-Transistor Logic (TTL) [4], valoarea 0 este asociată cu o tensiune mică și valoarea 1, cu o tensiune ridicată. Spunem că reprezentarea este în *logica pozitivă*. Atunci când se schimbă polaritatea (0 este asociat cu tensiuni ridicate și 1 este asociat cu tensiuni scăzute) reprezentarea este în *logica negativă*.

O funcție booleană se poate descrie folosind *tabelul de adevăr*. În partea stângă tabelul conține combinații de valori posibile pentru variabile, iar în dreapta sunt trecute valorile funcției pentru combinațiile respective. Un exemplu de tabel de adevăr pentru o funcție de 2 variabile x_1, x_0 este prezentat în Tabelul 1. 1, partea stângă. Conform acestuia, circuitul asociat va avea ieșirea 1, când intrarea $x_1=1$ și $x_0=1$, iar în rest 0. Din acest motiv mai poartă denumirea de funcția **ȘI**. Datorită simplității sale, un astfel de circuit se mai

numește *poartă logică fundamentală*. În algebra booleană operația ȘI se notează cu \cdot (a nu se confunda cu operația de înmulțire), iar simbolul grafic, folosit în cadrul unui circuit, este , cu intrările la stânga și ieșirea la dreapta. Există variante de porți ȘI cu număr extins de intrări, care realizează operația $x_{n-1} \cdot \dots \cdot x_0$. În acest caz, rezultatul operației este 1, numai când toate intrările sunt 1. Fiind o operație asociativă, extinderea la mai multe intrări se poate realiza și conectând mai multe porți cu număr redus de intrări. De exemplu, o funcție ȘI cu 3 intrări se poate implementa conform relației $x_2 \cdot x_1 \cdot x_0 = x_2 \cdot (x_1 \cdot x_0)$, folosind porți ȘI cu 2 intrări, în felul următor: .

Tabelul 1. 1 Tabelele de adevăr ale porților ȘI (stânga), SAU (mijloc), NOT (dreapta)

x_1	x_0	ȘI	x_1	x_0	SAU	x	NOT
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Există și alte porți fundamentale, precum poarta **SAU** și poarta **NOT** (sau **INV**). Operatorul pentru operația SAU este simbolul $+$ (atenție!, nu reprezintă adunare). Expresia $x_1 + x_0$ va avea rezultatul 1, dacă $x_1=1$ **sau** $x_0=1$ (vezi Tabelul 1. 1, mijloc). Simbolul porții SAU în circuite este . O poartă logică NOT = \bar{x} are o singură intrare și inversează valoarea acesteia la ieșire. Simbolul porții NOT este . Cele 3 porți realizează operațiile de bază din algebra booleană. Prin interconectări ale acestora se pot realiza circuite corespunzătoare unui număr nelimitat de funcții booleene, cu oricâte variabile. De exemplu, expresia $x_2 + (\bar{x}_1 \cdot x_0)$ se poate implementa cu circuitul: .

Într-o expresie ordinea priorităților este: 1) parantezele; 2) NOT; 3) ȘI; 4) SAU. Operațiile se realizează de la intrare spre ieșire, conform schemei circuitului, folosind tabelele de adevăr. Dacă $x_2=0$, $x_1=0$ și $x_0=1$, înlocuind în expresie, se obține $x_2 + (\bar{x}_1 \cdot x_0) = 0 + (\bar{0} \cdot 1) = 0 + (1 \cdot 1) = 0 + 1 = 1$. Similar, se poate calcula rezultatul expresiei pentru orice combinație de valori pe intrări și se poate determina tabelul de adevăr. În continuare, sunt prezentate alte combinații posibile și rezultatul obținut:

- $x_2=1, x_1=0, x_0=0 \rightarrow x_2 + (\bar{x}_1 \cdot x_0) = 1 + (\bar{0} \cdot 0) = 1 + (1 \cdot 0) = 1 + 0 = 1;$
- $x_2=0, x_1=1, x_0=1 \rightarrow x_2 + (\bar{x}_1 \cdot x_0) = 0 + (\bar{1} \cdot 1) = 0 + (0 \cdot 1) = 0 + 0 = 0;$
- $x_2=0, x_1=0, x_0=0 \rightarrow x_2 + (\bar{x}_1 \cdot x_0) = 0 + (\bar{0} \cdot 0) = 0 + (1 \cdot 0) = 0 + 0 = 0.$

1.4 Ghid de lucru cu Xilinx ISE Project Navigator

Utilitarul Project Navigator face parte din pachetul Xilinx ISE și oferă suport pentru proiectarea circuitelor într-un editor schematic.

1.4.1 Mediul de lucru Project Navigator

După pornirea utilitarului Project Navigator se încarcă proiectul **t1l_env** cu comanda **File > Open Project...** și se selectează fișierul **t1l_env.xise**. În cadrul utilitarului se pot distinge panourile de lucru (panels), ca în Figura 1. 1. O parte din acestea sunt folosite

pentru organizare (sunt amplasate în lateral), iar o parte afișează informații utile în procesul de prelucrare a unei scheme (sunt amplasate în partea inferioară) [Notă: Dacă interfața grafică este diferită se poate reseta cu comanda **Layout > Load Default Layout.**]:

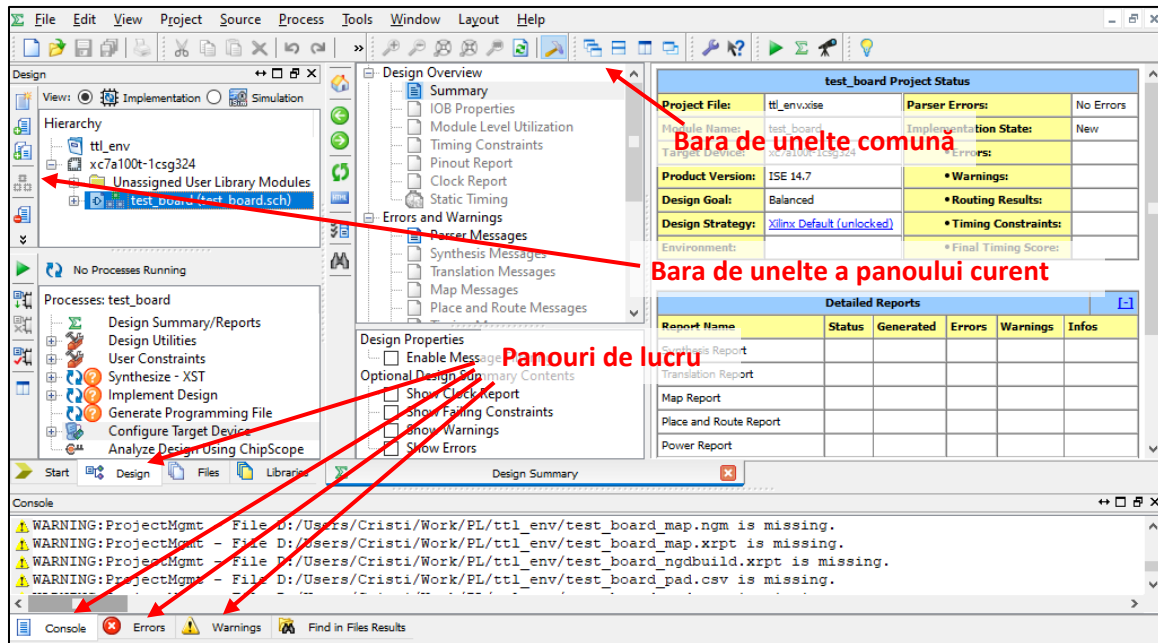



Figura 1. 1 Interfața utilitarului Project Navigator

- Panoul **Design** (Figura 1. 1) – conține structura proiectului organizată pe circuite; inițial, proiectul conține circuitul de test **test_board**. Dacă se apasă dublu-click pe numele circuitului se va afișa editorul schematic, care conține **schema** circuitului și se face automat saltul la panoul **Symbols**.
- Panoul **Symbols** (Figura 1. 2) – pune la dispoziție lista de componente, care se pot introduce în schema unui circuit, apăsând butonul **Add Symbol**  în bara de unelte a editorului schematic. În cadrul panoului componentele sunt organizate pe categorii. Prin selectarea unei categorii în partea superioară se pot vizualiza dedesubt circuitele corespunzătoare. Dacă se selectează <--All Symbols--> va apărea lista tuturor componentelor, din toate categoriile. O componentă poate fi căutată, în cadrul categoriei curente, după numele său, prin introducerea acestuia în caseta **Symbol Name Filter**. Printre categoriile cele mai importante se pot enumera: *Arithmetic*, *Buffer*, *General*, *Logic*, *Mux*, *Decoder*. Întotdeauna, opțiunea <--All Symbols--> este secundată de librăria locală a proiectului, care conține toate circuitele TTL studiate. **Notă:** Circuitele TTL au numele simbolului format din prefixul **TTL_** urmat de codul circuitului.
- Panourile **Console**, **Errors**, **Warnings** (Figura 1. 1) – afișează informații de procesare a schemei curente. În **Console** apar toate informațiile, iar în **Errors** și **Warnings** doar cele legate de eventuale erori sau avertismente. De exemplu, pentru intrările neconectate, se emite avertisment de conectare implicită la 0 (masă sau Ground – GND). Ieșirile neconectate se vor elimina. În schimb, erorile detectate trebuie corectate.

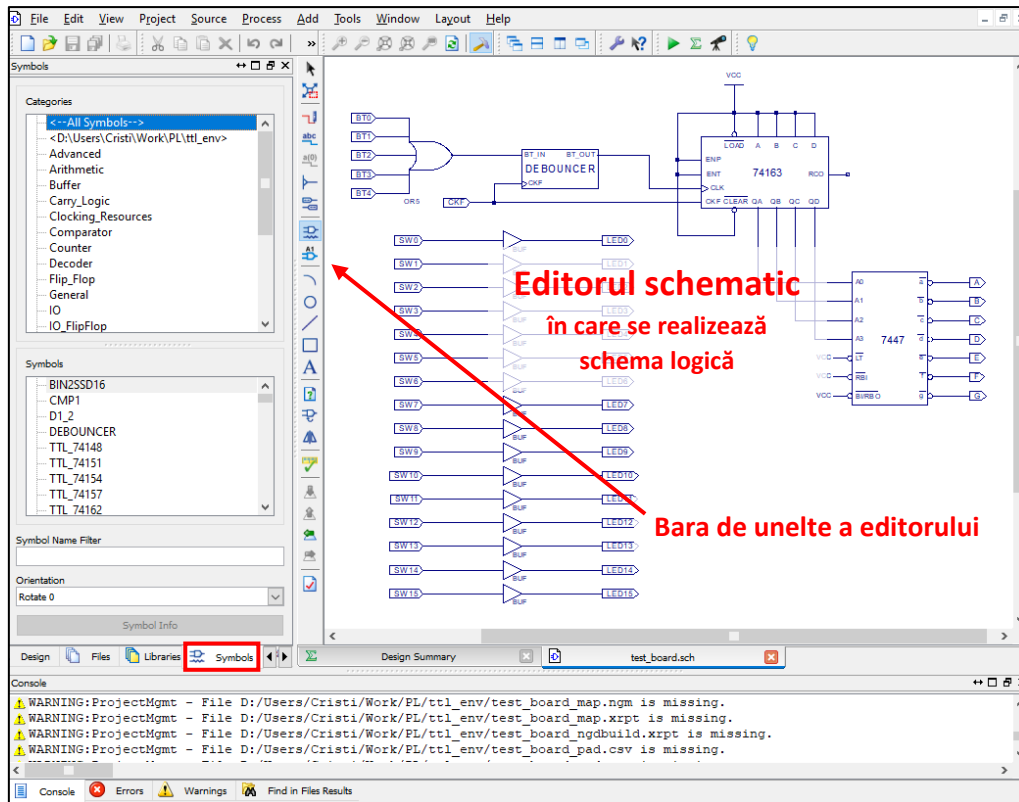


Figura 1. 2 Panoul Symbols și Editorul schematic

1.4.2 Crearea unei noi scheme

Pentru adăugarea unei noi scheme în cadrul proiectului, se revine în bara de unelte a panoului **Design** și se apasă butonul **New Source**. În fereastra următoare se selectează tipul **Schematic** și se introduce numele schemei (fără caractere speciale sau spații): de exemplu **lab01_01** (Figura 1. 3). **Notă:** verificați că opțiunea **Add to project** este bifată. Apoi se apasă **Next** și **Finish**. Pe ecran va apărea editorul schematic în care se poate defini schema circuitului prin adăugare de componente și interconectarea acestora. Aceasta se poate mări sau micșora apăsând pe **Zoom In** sau **Zoom Out** în bara de unelte comună amplasată în partea superioară.

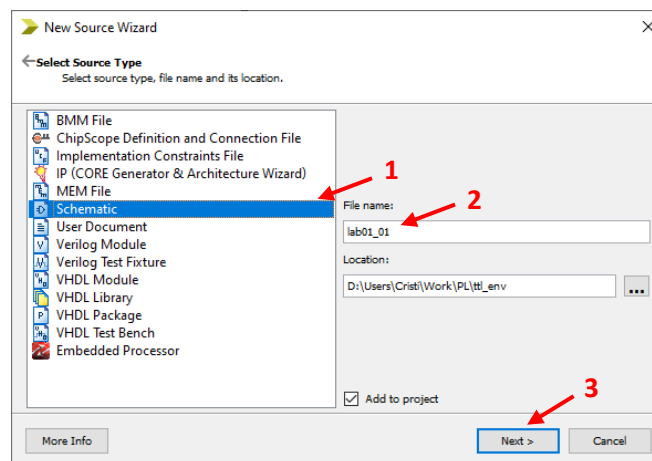



Figura 1. 3 Introducerea datelor necesare creării unui nou circuit

Cum proiectul poate conține mai multe scheme doar una poate să fie cea activă. Setarea schemei **lab01_01**, ca fiind cea activă, se realizează cu click-dreapta pe numele ei în panoul **Design** și alegerea opțiunii **Set as Top Module**. În dreptul numelui schemei active apare întotdeauna simbolul . Schema activă se poate schimba în orice moment. O schemă poate fi eliminată cu click-dreapta pe numele ei și opțiunea **Remove**.

În cadrul schemei se va introduce, pentru început, o poartă ȘI (engl. AND) cu 2 intrări. La panoul **Symbols**, în căsuța de căutare **Symbols Name Filter**, se introduce cuvântul *and2* (sufixul 2 reprezintă numărul de intrări), conform cu Figura 1. 4. Pentru a căuta în lista cu toate circuitele disponibile ne asigurăm că opțiunea <--All symbols--> este selectată în partea superioară. Se selectează simbolul **and2** din lista de simboluri afișate și se plasează pe schemă prin click. Prin apăsări repetate se poate introduce același simbol, în mod repetat. (**Notă:** Orientarea unui simbol poate fi modificată înainte de amplasare de la rubrica **Orientation**)

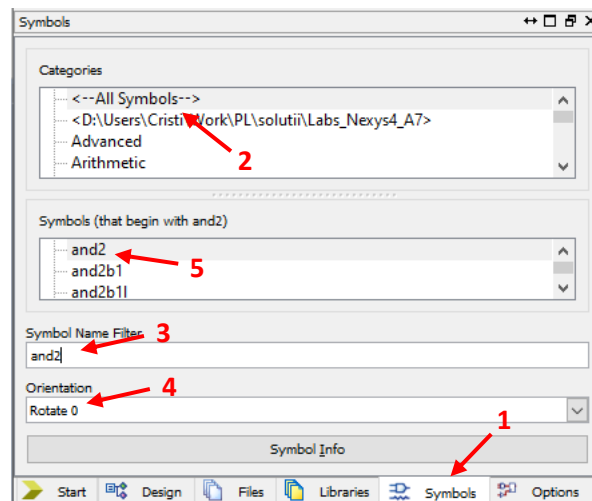





Figura 1. 4 Pașii pentru selectarea simbolului **and2** din lista de componente

Pentru selectarea oricărui element de pe schemă, trebuie să se intre în modul de selectare prin apăsarea butonului **Select** . Un element selectat poate fi deplasat cu mouse-ul sau poate fi șters, cu click-dreapta și **Delete** sau apăsând tasta *Del*.

În continuare, pe intrările și ieșirile componentei AND2 se vor trasa fire de conexiune. Prin apăsarea butonului **Add Wire**  se trece în modul de inserare a firelor. Ulterior, firele se pot trasa apăsând mouse-ul între cele 2 puncte conectate. De remarcat faptul că apare un simbol grafic specific  atunci când mouse-ul este deplasat deasupra punctelor de conexiune și prin simpla apăsare firul se va conecta automat la ele. Capetele rămase neconectate sunt marcate cu un dreptunghi roșu, ca în Figura 1. 5.

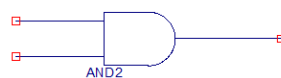


Figura 1. 5 Fire neconectate atașate la pinii componentei AND2

La capetele neconectate ale firelor se vor adăuga terminale de intrare și ieșire. Terminalele de intrare sunt utilizate pentru a introduce valori în circuit (0 sau 1), iar cele


de ieșire sunt necesare pentru vizualizarea rezultatelor în afara circuitului. Modul de inserare terminale se activează prin apăsarea butonului **Add I/O Marker** . Ulterior, se pot introduce terminale apăsând pe capetele neconectate ale firelor. (**Notă:** Terminalele se pot adăuga și în mod direct la pinii simbolului, fără a mai trasa fire.) Tipul de terminal potrivit (intrare sau ieșire) este determinat automat. La inserare, terminalele primesc un nume generat automat. Acesta se poate schimba prin click-dreapta pe terminal și opțiunea **Rename Port**. Numele nu trebuie să conțină caractere speciale sau spațiu. Cele 2 terminale de intrare se vor redenumi la A, respectiv B, iar cel de ieșire la F1 (Figura 1. 6). **Notă:** Firele conectate la terminale primesc automat același nume. **Important:** Firele cu nume identic sunt considerate conectate (se numește *conexiune logică*).



Figura 1. 6 Poarta AND2 conectată la terminale de intrare și ieșire

Se va adăuga în schema curentă o nouă funcție de ieșire F2, care realizează un ȘI extins la 3 intrări. Pentru aceasta se va introduce o nouă componentă AND care primește pe o intrare rezultatul porții AND existente și pe cealaltă un al 3-lea terminal de intrare C. Ieșirea acestei porți AND va fi legată la terminalul de ieșire F2. Pașii necesari sunt următorii:

1. Se introduce în schemă o nouă poartă AND2 din panoul **Symbols**.
2. Se trasează o conexiune dinspre ieșirea primului AND2 la una din intrările celui de-al doilea AND2. În consecință, va apărea o ramificație pe acest fir marcată printr-un dreptunghi. **Notă:** Toate ramificațiile sunt considerate unul și același fir.
3. Se trasează fire pe pinii neconectați ai noului AND2.
4. Se introduce un terminal C pe intrarea rămasă liberă a lui AND2 și un terminal F2 pe ieșire. Schema rezultată este prezentată în Figura 1. 7.
5. Se salvează schema în forma finală cu combinația de taste **Ctrl+S** sau cu comanda **File > Save**.

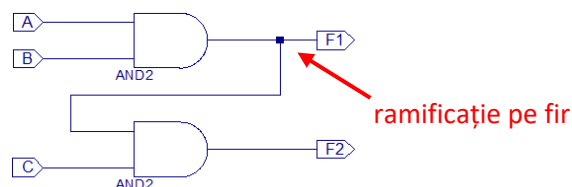


Figura 1. 7 Circuit care implementează funcțiile $F1=A \cdot B$ și $F2=(A \cdot B) \cdot C$

Ulterior, se va conecta terminalul de intrare C la o altă ieșire F3. **Notă:** Este permisă conectarea directă a unui terminal de intrare la un terminal de ieșire numai printr-un element buffer, denumit BUF, care se găsește în lista de simboluri la categoria *General*. După adăugarea acestei conexiuni prin intermediul unui BUF, circuitul devine cel din Figura 1. 8, în care se observă apariția unei noi ramificații de la intrarea C la buffer-ul BUF.

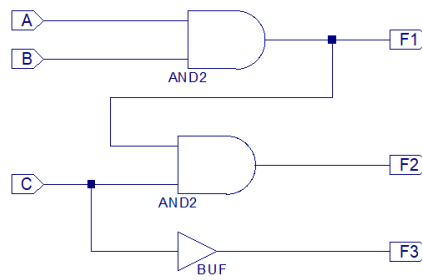


Figura 1. 8 Circuitul cu funcțiile $F1=A \cdot B$, $F2=(A \cdot B) \cdot C$ și $F3=C$

1.4.3 Definirea constrângerilor de implementare

Testarea circuitelor definite prin schema logică se realizează pe placa de dezvoltare din Figura 1. 9, care este dotată cu **16 comutatoare** și **5 butoane** pentru generarea valorilor pe terminalele de intrare. Vizualizarea rezultatelor înregistrate pe terminalele de ieșire se poate realiza pe cele **16 led-uri**.

La apăsarea unui **buton** acesta generează valoarea logică 1, altfel generează valoarea 0. Un **comutator** poziționat către led-uri generează valoarea 1 și valoarea 0 în poziția opusă. Un **led** care primește 0 este stins, iar dacă primește 1 luminează. Comutatoarele și led-urile sunt numerotate de la 0 la 15, de la dreapta spre stânga.

Asocierea terminalelor la butoane, comutatoare, și led-uri se realizează prin definirea unui fișier de constrângeri (nu este case sensitive) cu extensia .ucf (User Constraints File). Fiecare schemă trebuie să aibă asociat un astfel de fișier. Pentru simplificare, în directorul proiectului există un șablon predefinit de fișier de constrângeri denumit `_template.ucf`, care conține toate elementele plăcii, în comentarii (rânduri prefixate cu simbolul #). Se vor de-comenta doar rândurile asociate cu resursele necesare.

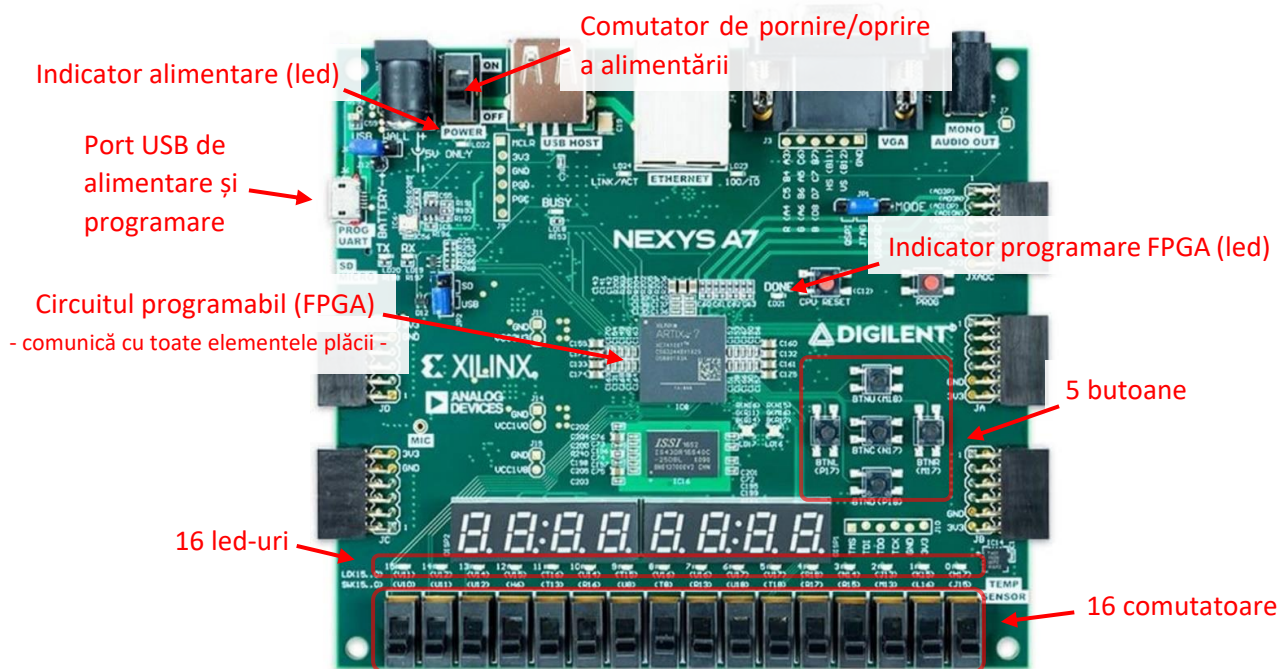


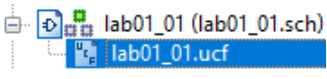


Figura 1. 9 Placa de dezvoltare

Înainte de asocierea unui fișier de constrângeri la o schemă este indicată crearea unei copii a fișierului `_template.ucf` cu numele schemei și extensia `ucf`. În cazul de față copia se va redenumi la `lab01_01.ucf`. Pentru ca asocierea să fie făcută la schema curentă aceasta trebuie să fie desemnată top-module, lucru indicat de simbolul  în dreptul ei, în panoul **Design**. În bara de unelte a panoului **Design** se va apăsa pe butonul **Add Source**  și se va selecta fișierul `lab01_01.ucf` de pe disc. Ulterior încărcării acesta va apărea sub schema `lab01_01` în structura proiectului  și se va deschide pentru editare cu dublu-click pe numele său.

Se poate observa că în cadrul fișierului toate terminalele au nume predefinite între ghilimele, după cuvântul cheie **NET**. Se vor de-comenta (prin ștergerea simbolului `#`) numai rândurile asociate resurselor necesare de pe placă și numele terminalelor se va actualiza în conformitate cu cele de pe schemă. Presupunând că terminalele A, B, C vor fi asociate la comutatoarele 0, 1, respectiv 2, vor trebui de-comentate rândurile 13-15 și modificate astfel (vezi marcajul roșu):

Switches

```
NET "A" LOC=J15 | IOSTANDARD=LVC MOS33 | CLOCK_DEDICATED_ROUTE=FALSE; # sw0
NET "B" LOC=L16 | IOSTANDARD=LVC MOS33 | CLOCK_DEDICATED_ROUTE=FALSE; # sw1
NET "C" LOC=M13 | IOSTANDARD=LVC MOS33 | CLOCK_DEDICATED_ROUTE=FALSE; # sw2
```

Notă: Valoarea asociată atributului **LOC** (de la locație) coincide cu codul înscris pe placă în dreptul comutatoarelor. Acest lucru este valabil și pentru butoane și led-uri.

Presupunând că ieșirile F1, F2 și F3 vor fi conectate la led-urile 0, 1 și 2 se vor de-comenta și ajusta rândurile 40-42 în felul următor:

LEDs

```
NET "F1" LOC=H17 | IOSTANDARD=LVC MOS33; # led0
NET "F2" LOC=K15 | IOSTANDARD=LVC MOS33; # led1
NET "F3" LOC=J13 | IOSTANDARD=LVC MOS33; # led2
```

Se salvează fișierul de constrângeri cu **Ctrl+S** sau cu comanda **File > Save**.

1.4.4 Generarea fișierului de programare

Fișierul de programare se poate genera de la rubrica **Generate Programming File**, situată în partea inferioară a panoului **Design** (Figura 1. 10), cu click-dreapta pe ea și una din comenzile **Run**, **ReRun** sau **Rerun All**. Pe măsură ce procesarea avansează pot să apară diverse erori sau avertismente. Dacă apar erori, în funcție de gravitatea acestora, întreg procesul se poate opri automat. Problemele sesizate trebuie eliminate și se va relua procesul de generare cu comanda **Rerun All**.

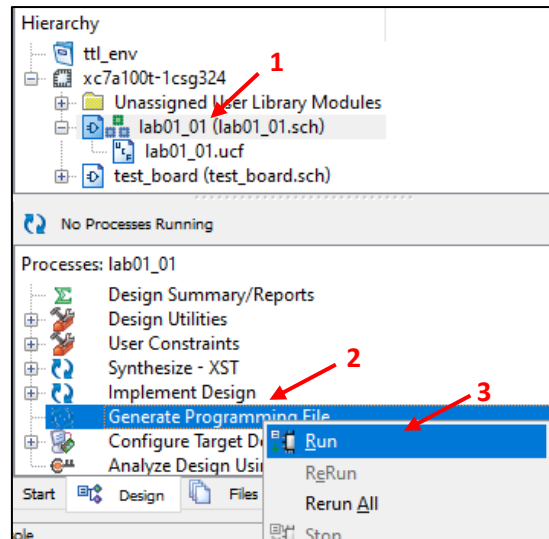


Figura 1. 10 Generarea fișierului de programare

Dacă generarea se încheie cu succes se va crea fișierul lab01_01.bit în directorul proiectului și va apărea iconița verde: **Generate Programming File**.

1.4.5 Programarea circuitului FPGA și testarea pe placa de dezvoltare

Important: Înainte de programarea circuitului, placa (Figura 1. 9) se conectează la stația de lucru folosind un cablu USB și se pornește alimentarea. Se va activa indicatorul led aflat sub comutatorul de alimentare.

Pentru programare, se lansează în execuție utilitarul ISE iMPACT cu click-dreapta pe pasul **Configure Target Device** și comanda **Run** (Figura 1. 11).

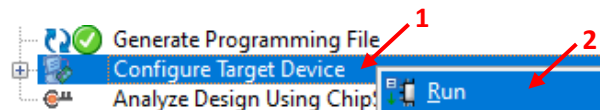


Figura 1. 11 Lansarea utilitarului ISE iMPACT

În cadrul utilitarului se deschide o sesiune de lucru cu dublu-click pe **Boundary Scan** (Figura 1. 12).

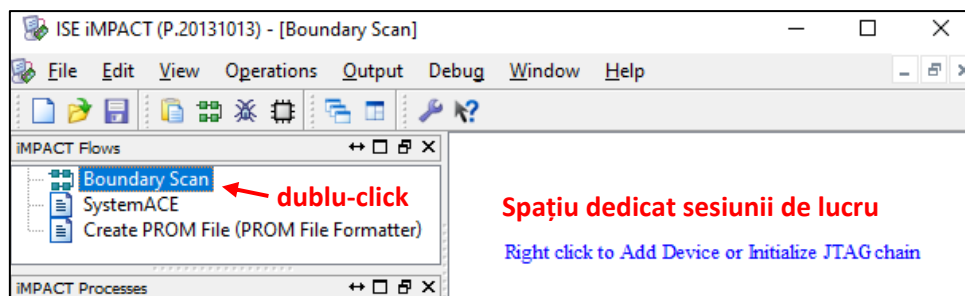
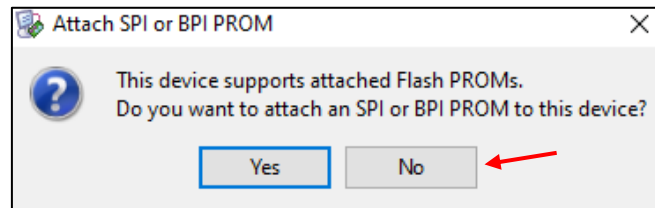


Figura 1. 12 Crearea unei sesiuni de lucru în ISE iMPACT

Pașii de realizare a unei conexiuni cu placa de dezvoltare:

1. Click-dreapta în spațiul dedicat sesiunii și se alege comanda **Initialize Chain**.

2. Se confirmă asocierea unui fișier de configurare și se selectează lab01_01.bit din directorul proiectului.
3. **Important:** Se va refuza atașarea unui modul SPI sau BPI PROM ori de câte ori apare această fereastră.



4. Se apasă **OK** în fereastra următoare de afișare a proprietăților de programare.

O conexiune funcțională va afișa în spațiul de lucru simbolul circuitului FPGA de pe placă, modelul acestuia și numele fișierului .bit de configurare curent, ca în Figura 1. 13.

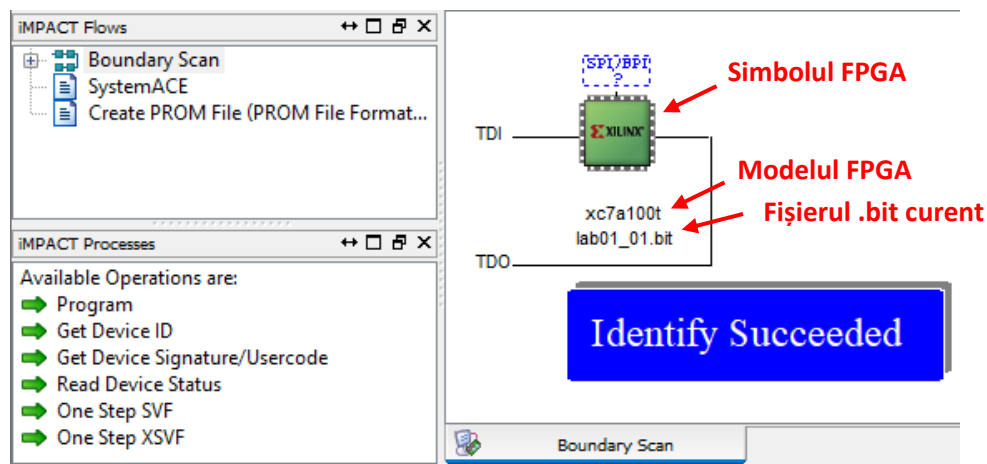


Figura 1. 13 ISE iMPACT recunoaște placa de dezvoltare

Urmează pasul final, de programare a circuitului FPGA, cu click-dreapta pe simbolul circuitului și se lansează comanda **Program** (Figura 1. 14). Încheierea acestei ultime etape este marcată de activarea pe placă a indicatorului led pentru programare. Din acest moment placa funcționează conform circuitului de pe schemă. Cu ajutorul comutatoarelor i se pot transmite valori de 0 sau 1 pe intrări, iar pe led-urile alocate se vor vizualiza rezultatele: F1=1, dacă A=B=1, F2=1, dacă A=B=C=1 și F3=C.

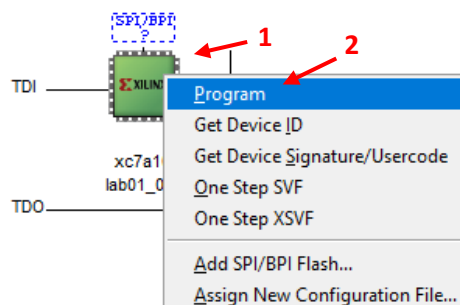


Figura 1. 14 Lansarea pasului de programare după atașarea unui fișier .bit

Configurarea cu circuitul curent se menține cât timp placa este alimentată. Sesiunea ISE iMPACT poate fi menținută pornită chiar dacă se aduc modificări la schemă.

Pentru a economisi timp, nu se recomandă închiderea ISE iMPACT după fiecare testare. În cazul unor modificări cu utilitarul Project Navigator, se va regenera fișierul de programare, se va reveni în ISE iMPACT și se va relua programarea cu pasul final, comanda **Program**, fără a parcurge pașii anteriori pentru realizarea conexiunii.

În cazul în care se dorește programarea cu un fișier .bit, având nume diferit de cel curent, se poate folosi comanda **Assign New Configuration File...** (Figura 1. 14), care va solicita alegerea de pe disc a noului fișier. Ulterior, se poate realiza programarea cu comanda **Program**. Nici în acest caz nu este necesară refacerea conexiunii.

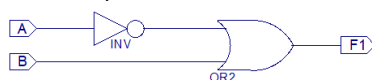
Notă: La închiderea utilitarului ISE iMPACT (la final de laborator) nu este recomandată salvarea proiectului din cadrul acestuia, dacă se solicită acest lucru.

Observație: Proiectul *ttl_env* poate fi curățat de fișierele generate pe parcurs, prin executarea scriptului **clean_win.bat** în Windows sau **clean_lnx.sh** în Linux, ambele aflate în folder-ul proiectului. Scripturile păstrează schemele .sch și fișierele de programare .ucf.

1.5 Activități practice

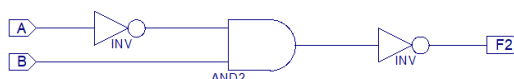
1. Adăugați în cadrul proiectului *ttl_env* următoarele circuite (în scheme diferite) și testați funcționalitatea acestora pe placă, folosind tabelele de adevăr alăturate. Denumirea schemelor și alegerea comutatoarelor și a led-urilor utilizate este la alegere.

a) $F1 = \bar{A} + B$



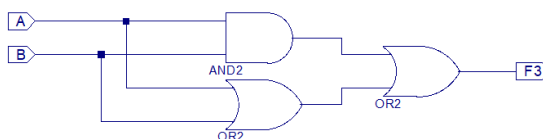
A	B	F1
0	0	1
0	1	1
1	0	0
1	1	1

b) $F2 = \overline{A \cdot B}$



A	B	F2
0	0	1
0	1	0
1	0	1
1	1	1

c) $F3 = (A \cdot B) + (A + B)$



A	B	F3
0	0	0
0	1	1
1	0	1
1	1	1

1.6 Bibliografie

[1] Digilent, Nexys A7 Reference Manual, [Online]. Available:

<https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual>

[2] AMD Xilinx, ISE WebPACK Design Software, [Online]. Available:

<https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html>

[3] Wikipedia, George Boole, [Online]. Available:

https://en.wikipedia.org/wiki/George_Boole

[4] Wikipedia, Transistor–transistor logic, [Online]. Available:

https://en.wikipedia.org/wiki/Transistor-transistor_logic