

FLUXUL DE PROIECTARE CU CIRCUITE FPGA

1. Proiectarea cu circuite logice programabile

1.1. Fluxul de proiectare

Pentru proiectarea sistemelor digitale utilizând circuite logice programabile (PLD – *Programmable Logic Device*), cum sunt circuitele FPGA, se utilizează pachete de programe de proiectare asistată de calculator (CAD – *Computer Aided Design*). Aceste pachete de programe asistă proiectantul în toate etapele procesului de proiectare. Astfel, majoritatea pachetelor CAD pentru circuitele programabile asigură următoarele funcții principale:

- *Descrierea* sistemului digital;
- *Sinteza* descrierii, ceea ce înseamnă transformarea descrierii într-o listă de conexiuni conținând porți logice elementare și interconexiunile dintre ele;
- *Simularea funcțională* a sistemului pe baza listei de conexiuni obținute, înainte de implementarea într-un anumit circuit;
- *Implementarea* sistemului într-un circuit, prin adaptarea listei de conexiuni pentru o utilizare eficientă a resurselor disponibile ale circuitului;
- *Configurarea* (programarea) circuitului pentru ca acesta să realizeze funcția dorită.

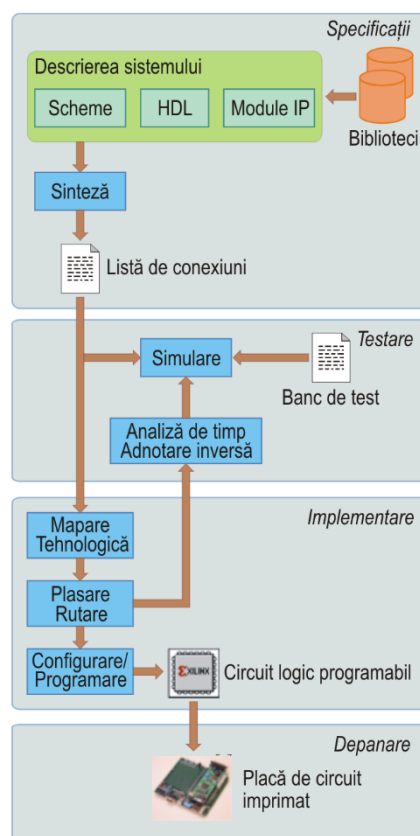


Figura 1. Fluxul de proiectare a sistemelor digitale utilizând circuite logice programabile.

Figura 1 ilustrează etapele din procesul de proiectare a sistemelor digitale utilizând circuite logice programabile. Modulele IP (*Intellectual Property*) reprezintă module hardware complexe care sunt testate în mod extensiv și care pot fi utilizate în diferite proiecte pentru a reduce în mod semnificativ timpul de proiectare.

În continuare sunt descrise mai detaliat principalele etape de proiectare.

1.2. Descrierea sistemului

Există mai multe metode pentru descrierea sistemelor digitale. Principalele metode sunt prin scheme logice, prin limbaje de descriere hardware (HDL – *Hardware Description Language*) și prin diagrame de stare.

În mod tradițional, sistemele digitale sunt descrise prin scheme logice. Pentru această descriere se utilizează un editor schematic, care permite specificarea componentelor care trebuie utilizate și a modului în care acestea trebuie interconectate. Această metodă este ilustrată în figura 2. Circuitul din această figură detectează secvența binară 1010 aplicată la intrarea X. La detectarea acestei secvențe, ieșirea Z va fi setată la 1 logic.

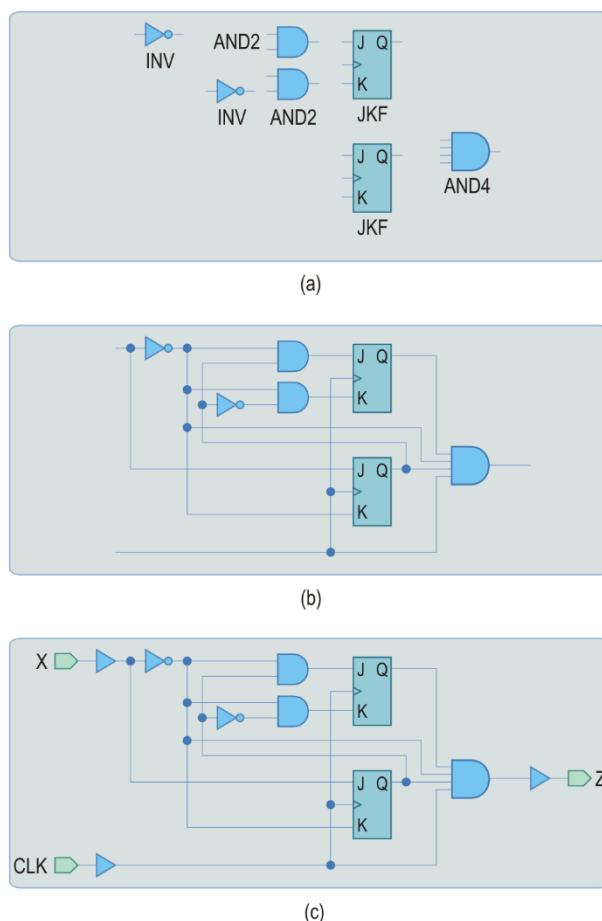


Figura 2. Etapele descrierii unui sistem digital utilizând scheme: (a) selectarea și amplasarea componentelor; (b) conectarea componentelor; (c) adăugarea porturilor de I/E.

Principalele etape la utilizarea schemelor logice pentru proiectarea unui sistem digital sunt următoarele:

1. Într-un editor schematic se selectează componentele necesare dintr-o bibliotecă de componente. Asemenea componente pot fi, de exemplu, porți elementare, multiple-xoare, decodificatoare, numărătoare, unități aritmetice etc. În funcție de circuitul care va utilizat, proiectantul trebuie să selecteze o anumită bibliotecă de componente, deoarece există biblioteci care sunt specifice diferiților producători de circuite logice

programabile și diferitelor familii de circuite. Circuitele dintr-o anumită familie diferă prin capacitatea lor, viteza și capsula utilizată. În această etapă, nu este necesară specificarea exactă a circuitului care va utiliza dintr-o anumită familie.

2. Componentele selectate și plasate în cadrul unei scheme sunt interconectate prin fire de legătură. Proiectantul realizează interconectarea componentelor pentru a obține configurația necesară pentru o anumită aplicație.
3. Se adaugă și se etichetează porturile de I/E. Aceste porturi definesc intrările și ieșirile sistemului, permițând aplicarea semnalelor la pinii de intrare ai sistemului digital și preluarea semnalelor de ieșire generate de sistem la pinii de ieșire. Semnalele de intrare sunt aplicate la intrările sistemului digital prin intermediul unor buffere de intrare, iar semnalele de ieșire sunt preluate de la sistemul digital prin intermediul unor buffere de ieșire. Aceste buffere izolează sistemul digital față de exterior. În unele cazuri, bufferele de I/E sunt adăugate în mod automat de sistemul CAD.

Pe lângă schemele logice, o altă posibilitate pentru descrierea sistemelor digitale este utilizarea unui limbaj de descriere hardware. Aceste limbaje sunt utilizate actualmente pe scară largă, fiind preferate pentru descrierea sistemelor cu complexitate ridicată, datorită următoarelor avantaje:

- Posibilitatea unei descrieri funcționale a sistemelor, aceasta fiind o descriere de nivel mai înalt, fără detalierea structurii la nivelul porților elementare. Astfel, timpul necesar pentru descrierea sistemelor complexe se reduce în mod semnificativ.
- Independența descrierilor HDL față de diferite tipuri de circuite. În timp ce schemele logice sunt realizate cu componente de bibliotecă specifice unei anumite familii de circuite, descrierile HDL sunt complet independente de un anumit circuit, astfel încât aceeași descriere se poate utiliza pentru implementarea sistemului într-un anumit circuit FPGA, dar și într-un alt tip de circuit logic programabil, de exemplu, într-o rețea logică programabilă.
- Posibilitatea modificării mai simple a descrierii HDL a unui sistem, datorită faptului că o asemenea descriere reprezintă în același timp o documentare a sistemului.

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

entity detect is
    Port ( CLK : in STD_LOGIC;
          X  : in STD_LOGIC;
          Z  : out STD_LOGIC);
end detect;

architecture automat_stare of detect is
    type tip_stare is (A, B, C, D);
    signal stare: tip_stare;
begin
    tranz_stare: process (CLK)
    begin
        if rising_edge (CLK) then
            case stare is
                when A =>
                    if (X = '1') then stare <= B; end if;
                when B =>
                    if (X = '0') then stare <= C; end if;
                when C =>
                    if (X = '0') then stare <= A;
                    else stare <= D; end if;
                when D =>
                    if (X = '0') then stare <= C;
                    else stare <= B; end if;
            end case;
        end if;
    end process tranz_stare;
    Z <= '1' when stare = D else '0';
end automat_stare;
```

Figura 3. Descrierea în limbajul VHDL a circuitului ilustrat în figura 2.

Există diferite limbaje de descriere hardware, dar cele mai utilizate sunt limbajele VHDL (*VHSIC Hardware Description Language*) și Verilog. Aceste limbaje sunt standardizate de institutul IEEE. Figura 3 prezintă o descriere posibilă în limbajul VHDL a circuitului ilustrat în figura 2.

Pentru descrierea automatelor cu stări finite se utilizează pe scară largă diagramele de stare. Sistemele CAD pentru proiectarea cu circuite logice programabile pot conține editoare pentru diagramele de stare, care permit specificarea sub formă grafică a stărilor sistemului, a tranzițiilor între stări și a semnalelor de ieșire care trebuie generate în fiecare stare. O diagramă de stare va fi compilată de către sistemul CAD într-o reprezentare internă sau într-o descriere HDL, care poate fi simulată și utilizată apoi pentru implementarea automatului într-un anumit circuit. Figura 4 prezintă o diagramă de stare echivalentă cu circuitul ilustrat în figura 2 și descrierea în limbajul VHDL din figura 3.

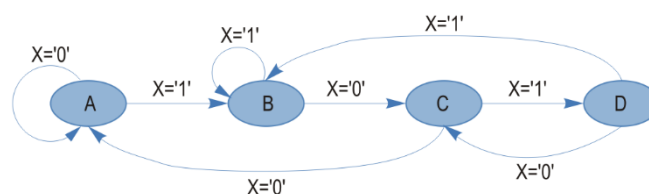


Figura 4. Diagrama de stare echivalentă a circuitului reprezentat prin schema din figura 2 și descrierea în limbajul VHDL din figura 3.

1.3. Sinteza sistemului

După descrierea sistemului digital, etapa următoare din cadrul procesului de proiectare este cea de sinteză a sistemului. Sinteza constă în translatarea schemei logice, a descrierii HDL sau a diagramei de stare într-o *listă de conexiuni*. Această traducere se realizează cu ajutorul unui program de sinteză al sistemului CAD. Lista de conexiuni (*“netlist”*) este o descriere compactă a sistemului digital sub formă textuală, în care sunt specificate componentele sistemului, interconexiunile dintre acestea și pinii de intrare/ieșire. Lista de conexiuni este prelucrată de celelalte componente ale sistemului CAD pentru realizarea etapelor următoare din cadrul procesului de proiectare.

Există diferite formate pentru listele de conexiuni. Formatul EDIF (*Electronic Digital Interchange Format*) este cel mai utilizat și reprezintă un standard industrial. Pe lângă acest format standard, se pot utiliza diferite formate care sunt specifice anumitor producători de circuite. Un exemplu este formatul XNF (*Xilinx Netlist Format*), care este un format proprietar al firmei Xilinx. O altă posibilitate este utilizarea unui limbaj de descriere hardware ca format pentru lista de conexiuni. De exemplu, sistemul CAD poate utiliza o reprezentare structurală a sistemului proiectat într-un limbaj de descriere hardware specificat de proiectant.

Relația dintre schema logică a unui circuit simplu și un format posibil al unei liste de conexiuni este ilustrată în figura 5. În prima parte a listei de conexiuni sunt declarate componentele din cadrul schemei, iar în a doua parte sunt specificate conexiunile dintre componente. Numele componentelor sunt G1..G7, iar numele conexiunilor sunt N1..N10. Aceste nume sunt fie cele specificate de proiectant, fie cele asigurate în mod automat de sistemul CAD.

În circuitul ilustrat în figura 5, există două inversoare (G1 și G2), două porți ȘI cu două intrări (G3 și G4), o poartă ȘI cu patru intrări (G7) și două bistabile JK (G5 și G6). Inversoarele au un pin de intrare IN, un pin de ieșire OUT, un pin de alimentare Vcc și un pin de masă GND. Similar, porțile ȘI cu două intrări au doi pini de intrare IN1 și IN2, un pin de ieșire OUT, un pin de alimentare și un pin de masă. Bistabilele au doi pini pentru intrările de date J și K, un pin pentru intrarea de ceas C și un pin pentru ieșirea Q, pe lângă pinii de alimentare și masă. O conexiune este reprezentată prin listarea tuturor pinilor care sunt conectați împreună. Semnalele de intrare X și CLK sunt conectate la pinii de intrare cu aceleași nume ai circuitului, iar semnalul de ieșire Z este conectat la pinul de ieșire al circuitului.

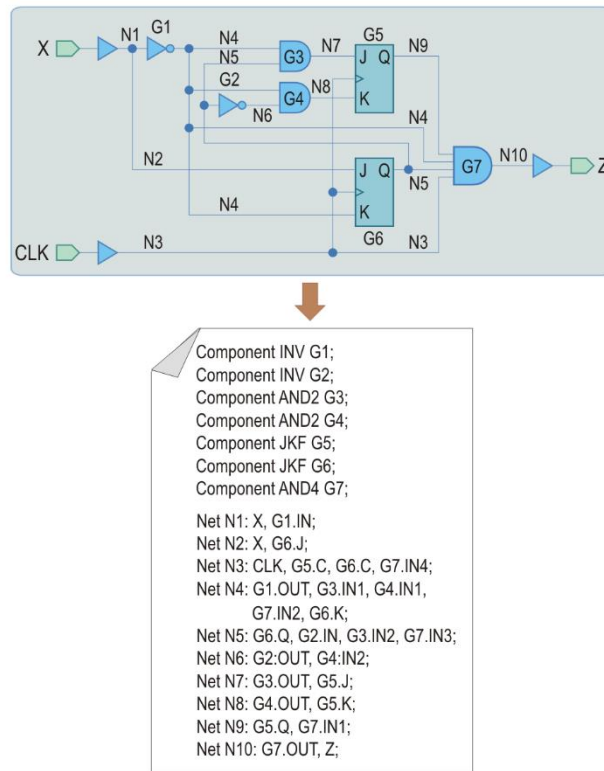


Figura 5. Relația dintre schema logică și lista de conexiuni pentru circuitul din figura 2.

Proiectantul poate specifica diferite criterii de optimizare de care să se țină cont în timpul procesului de sinteză. Exemple de asemenea opțiuni sunt: minimizarea numărului de porți elementare necesare, obținerea vitezei maxime de funcționare a circuitului, sau minimizarea puterii consumate. Proiectantul poate experimenta cu diferite criterii de optimizare pentru a obține soluția cea mai convenabilă pentru aplicația respectivă.

1.4. Simularea funcțională

În această etapă se utilizează un program simulator pentru verificarea funcționării sistemului proiectat, înainte de implementarea acestuia într-un circuit logic programabil. Această verificare se referă doar la aspectele funcționale ale sistemului, fără a se lua în considerare întârzierile semnalelor, care vor fi cunoscute numai după implementare. Pentru verificarea funcțională, proiectantul furnizează simulatorului mai multe combinații ale valorilor semnalelor de intrare; o asemenea combinație este numită *vector de test*. De asemenea, proiectantul poate specifica valorile semnalelor de ieșire care trebuie generate de sistem pentru fiecare vector de test. O secvență de vectori de test care se aplică la intrările unui sistem se numește *banc de test*.

Simulatorul aplică vectorii de test la intrările sistemului unul câte unul, determină semnalele de ieșire care sunt generate de sistem și le compară cu valorile semnalelor care au fost specificate de proiectant. În cazul în care apar diferențe, simulatorul afișează mesaje care indică diferențele apărute. Proiectantul va efectua modificările necesare ale descrierii sistemului pentru a corecta erorile apărute, va efectua sinteza descrierii modificate și va executa din nou simularea funcțională. Aceste etape vor fi repetate până când sistemul va funcționa conform cerințelor.

Figura 6 ilustrează modul în care pot fi vizualizate pe ecranul calculatorului semnalele de intrare și de ieșire ale detectorului de secvență utilizat ca exemplu în secțiunile precedente.

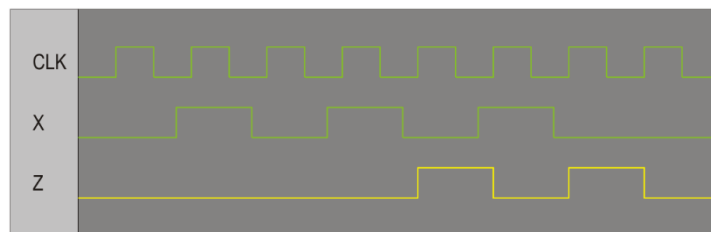


Figura 6. Semnalele de intrare și de ieșire ale detectorului de secvență afișate la simularea funcțională a circuitului.

1.5. Maparea tehnologică

Etapele următoare din cadrul fluxului de proiectare realizează implementarea sistemului proiectat într-un circuit logic programabil. Prima etapă a implementării este cea de *mapare tehnologică*. Această etapă constă dintr-o serie de operații care procesează lista de conexiuni și o adaptează la particularitățile și resursele disponibile ale circuitului utilizat pentru implementare. Operațiile executate în această etapă variază în funcție de sistemul de sinteză. Cele mai obișnuite operații sunt: adaptarea la elementele fizice ale circuitului, optimizarea și verificarea regulilor de proiectare (de exemplu, testarea depășirii numărului pinilor de I/E disponibili în cadrul circuitului). În timpul acestei etape, proiectantul selectează tipul circuitului logic programabil care va fi utilizat, capsula circuitului integrat, viteza și alte opțiuni specifice circuitului respectiv.

În urma execuției operațiilor din etapa de mapare tehnologică se generează un raport detaliat al rezultatelor obținute. Pe lângă mesaje de eroare și de avertizare, de obicei se creează o listă cu resursele utilizate din cadrul circuitului.

Figura 7 ilustrează etapa de mapare tehnologică pentru circuitul utilizat ca exemplu. După cum se observă, schema circuitului a fost modificată pentru a utiliza bistabile D în locul bistabilelor JK, iar porțile ȘI au fost înlocuite cu porți ȘI-NU. Se menționează că aceste transformări sunt efectuate asupra listei de conexiuni care s-a obținut în urma etapei de sinteză, schema din figura 7 fiind doar ilustrativă.

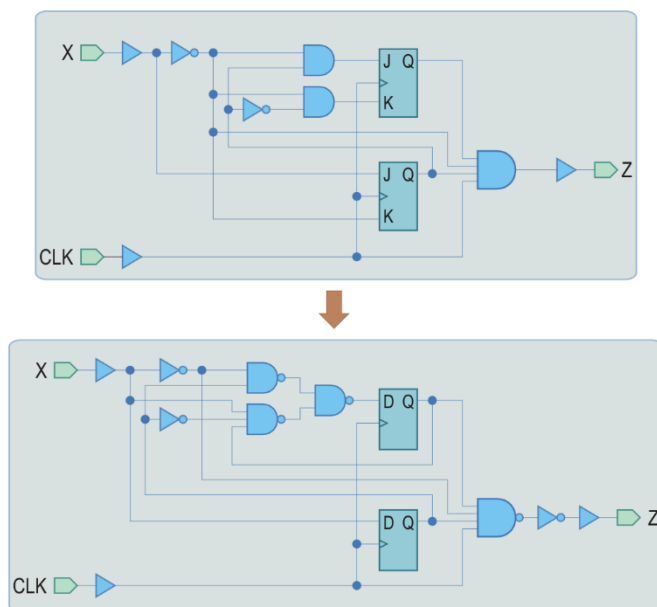


Figura 7. Ilustrarea etapei de mapare tehnologică pentru circuitul din figura 2.

1.6. Plasarea și rutarea

Aceste operații sunt executate atunci când se utilizează un circuit FPGA pentru implementare. Pentru proiectarea cu alte circuite PLD, operația echivalentă este numită adaptare (*“fitting”*). *Plasarea* este procesul de selectare a unor module sau blocuri logice ale circuitului logic programabil care vor fi utilizate pentru implementarea diferitelor funcții ale sistemului digital. *Rutarea* constă în interconectarea acestor blocuri logice utilizând resursele de rutare disponibile ale circuitului.

Majoritatea sistemelor CAD realizează operațiile de plasare și rutare în mod automat, astfel încât utilizatorul nu trebuie să cunoască detaliile arhitecturii circuitului. Anumite sisteme permit utilizatorilor experți plasarea și rutarea manuală a unor porțiuni critice ale sistemului digital pentru a obține performanțe superioare.

Figura 8 ilustrează plasarea și rutarea listei de conexiuni obținute în urma mapării tehnologice a circuitului utilizat ca exemplu. După selectarea blocurilor logice care vor fi utilizate pentru implementarea circuitului, acestea se configurează pentru implementarea unor porțiuni ale circuitului. Pentru interconectarea semnalelor generate de diferitele blocuri logice se utilizează resursele de rutare disponibile. Aceste resurse sunt indicate în figură prin linii orizontale și verticale. Intrările și ieșirile utilizate ale blocurilor logice se conectează la liniile de rutare prin puncte de conexiune programabile (reprezentate în figură prin cercuri), iar liniile de rutare sunt interconectate prin comutatoare programabile.

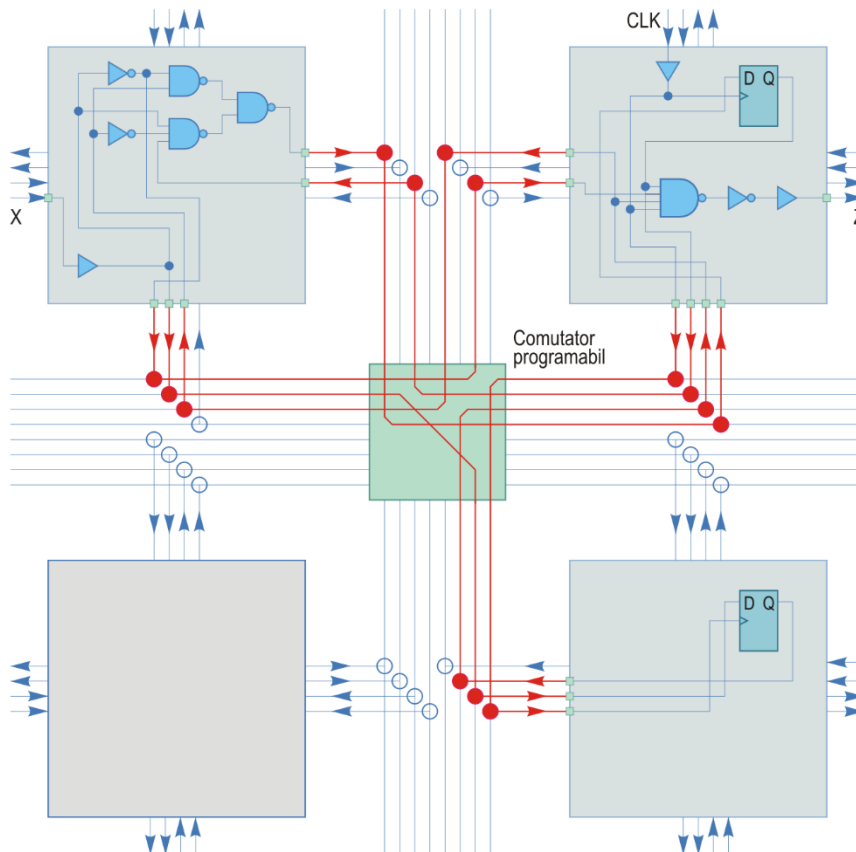


Figura 8. Ilustrarea etapelor de plasare și rutare pentru circuitul din figura 7.

Operațiile de plasare și rutare pot necesita un timp ridicat pentru execuție în cazul sistemelor digitale complexe, deoarece sunt necesare operații complexe pentru determinarea și configurarea blocurilor logice necesare din circuitul programabil, interconectarea corectă a acestora și verificarea faptului că sunt asigurate cerințele de performanță specificate în timpul proiectării.

1.7. Analiza de timp

Pachetele de programe CAD pentru proiectarea sistemelor digitale conțin de obicei un program numit analizor de timp, care poate furniza informații despre întârzierile semnalelor. Aceste informații se referă atât la întârzierile introduse de blocurile logice, cât și la întârzierile datorate interconexiunilor. Analizorul poate afișa aceste informații în diferite moduri, de exemplu, prin ordonarea conexiunilor în ordinea descrescătoare a întârzierilor semnalelor. Proiectantul poate utiliza informațiile despre întârzierile semnalelor pentru a realiza o nouă simulare a sistemului, în care să se țină cont de aceste întârzieri. Această operație prin care se furnizează simulatorului informații detaliate despre întârzierile semnalelor se numește adnotare inversă (“*back-annotation*”).

1.8. Configurarea sau programarea circuitului

Operația de *configurare* se referă la circuitele programabile bazate pe memorii volatile RAM statice și constă din încărcarea informațiilor de configurare în memoria circuitului. Operația de *programare* se referă la circuitele logice programabile bazate pe memorii nevolatile (cum sunt circuitele care conțin antifuzibile). Această operație se execută similar cu cea de configurare, dar informațiile de configurare sunt păstrate și după întreruperea tensiunii de alimentare.

La sfârșitul operațiilor de plasare și rutare, se generează un fișier care conține toate informațiile necesare pentru configurarea circuitului. Aceste informații se referă atât la configurarea blocurilor logice ale circuitului, cât și la specificarea interconexiunilor dintre blocurile logice. Fișierul în care se înscriu aceste informații conține un șir de biți (“*bitstream*”), fiecare bit indicând starea închisă sau deschisă a unui comutator. Circuitele logice programabile conțin un număr mare de asemenea comutatoare. Un comutator poate fi implementat printr-un tranzistor sau o celulă de memorie. Un bit de 1 din șirul de biți va determina închiderea unui comutator și, deci, stabilirea unei conexiuni. Biții din fișierul de configurare sunt aranjați într-un anumit format pentru a realiza o corespondență între un bit și comutatorul corespunzător.

Conținutul fișierului de configurare se transferă la circuitul logic programabil, aflat de obicei pe o placă de circuit imprimat sau o placă de dezvoltare împreună cu alte circuite. Comutatoarele circuitului se închid sau rămân deschise în funcție de valorile biților din șirul de configurare.

Din cauza utilizării unei memorii volatile, circuitul trebuie configurat din nou după fiecare întrerupere a tensiunii de alimentare. Informațiile de configurare pot fi păstrate într-o memorie nevolatilă, iar circuitul poate fi configurat în mod automat din această memorie nevolatilă la aplicarea tensiunii de alimentare.

Configurarea sau programarea se pot realiza de la un calculator utilizând mai multe tipuri de interfețe: o interfață paralelă, cum este portul paralel standard sau îmbunătățit; o interfață serială, cum este SPI (*Serial Peripheral Interface*); interfața USB (*Universal Serial Bus*). Placa cu circuitul logic programabil trebuie să conțină un conector pentru interfața respectivă utilizată.

O altă posibilitate este utilizarea unui cablu special și a unei metodologii de configurare propusă de organizația JTAG (*Joint Test Advisory Group*). Această metodologie, cunoscută și sub numele de “*Boundary-Scan*”, a fost standardizată de institutele IEEE și ANSI ca standardul 1149.1, reprezentând un set de reguli de proiectare care facilitează configurarea sau programarea circuitelor, testarea și depanarea acestora. Un capăt al cablului JTAG se conectează la interfața paralelă sau USB a calculatorului, iar celălalt capăt se conectează la un număr de cinci pini speciali de pe placa de dezvoltare. Informațiile de configurare sunt transferate serial la circuitul logic programabil. Un asemenea cablu permite și testarea sistemului digital implementat prin citirea unor informații (valori ale semnalelor sau conținutul unor locații de memorie) de la circuitul logic programabil în timpul funcționării, transferul acestor informații la calculator și afișarea lor pe ecran.

Figura 9 prezintă conectarea unui cablu JTAG în modul JTAG (sau “*Boundary-Scan*”) la un sistem de dezvoltare conținând unul sau mai multe circuite logice programabile. Firele de legătură se conectează cu un capăt la pinii JTAG ai cablului, iar cu celălalt capăt la pinii JTAG corespunzători ai plăcii de dezvoltare. Un asemenea cablu poate fi utilizat fie pentru configurarea unui singur circuit programabil, fie a mai multor circuite conectate într-un lanț “*Boundary-Scan*”.

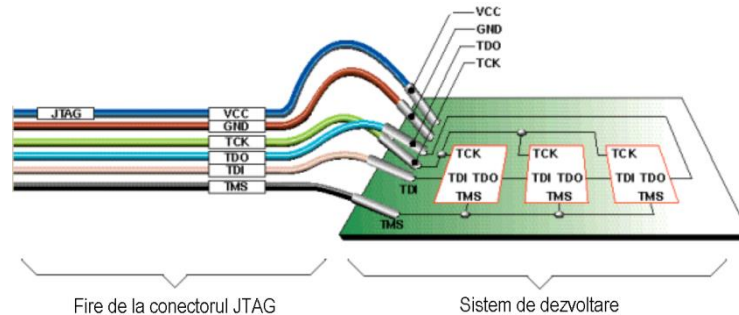


Figura 9. Conectarea unui cablu JTAG la un sistem de dezvoltare în modul “*Boundary-Scan*”.

Tabelul 1 prezintă denumirea și semnificația semnalelor JTAG care sunt utilizate pentru configurarea circuitelor logice programabile.

Tabelul 1. Denumirea și semnificația semnalelor JTAG utilizate pentru configurarea circuitelor logice programabile.

Denumire	Semnificație
VCC	Alimentare - Tensiunea de alimentare (5 V, 3,3 V sau 2,5 V)
GND	Masă - Referința pentru masa electrică
TCK	<i>Test Clock</i> - Semnal de ceas pentru circuitele logice programabile
TDO	<i>Test Data Out</i> - Semnal pentru citirea datelor de la circuitele logice programabile
TDI	<i>Test Data In</i> - Semnal pentru transmiterea instrucțiunilor și datelor la circuitele logice programabile
TMS	<i>Test Mode Select</i> - Semnal decodificat de controlerul JTAG pentru controlul operațiilor

1.9. Verificarea și depanarea sistemului

În această ultimă etapă a procesului de proiectare, se verifică funcționarea sistemului digital în condiții reale. O funcționare necorespunzătoare se poate datora nerespectării specificațiilor de proiectare, a specificațiilor circuitului utilizat pentru implementare, a unor aspecte legate de întârzierile semnalelor etc. Depanarea poate fi simplificată dacă circuitul se configurează astfel încât să conțină unele module speciale care permit citirea valorii unor semnale în timpul funcționării și transferul acestor informații la calculator, utilizând un cablu JTAG și un program special pentru afișarea formei de undă a semnalelor dorite.

2. Mediul de proiectare Xilinx Vivado IDE

Mediul de proiectare Xilinx Vivado Design Suite integrează toate utilitățile necesare pentru proiectarea sistemelor digitale utilizând circuitele FPGA Xilinx din seria 7, UltraScale, UltraScale+, ca și circuitul SoC (System-on-Chip) Zynq-7000. Pentru generațiile precedente de circuite FPGA Xilinx, trebuie să se utilizeze pachetul software Xilinx ISE Design Suite. Mediul Vivado Design Suite pune la dispoziție o interfață grafică numită Vivado Integrated Design Environment (IDE). Interfața grafică Vivado IDE permite invocarea utilităților de proiectare corespunzătoare pentru executarea operațiilor necesare în diferitele etape ale fluxului de proiectare cu circuite FPGA, cum sunt descrierea sistemului, sinteza, implementarea, analiza proiectului, configurarea circuitului, verificarea și depanarea. Facilitățile de analiză a proiectului cuprind simularea logică în diferitele etape ale fluxului de proiectare, definirea

constrângerilor, verificarea regulilor de proiectare, analiza de timp, analiza puterii consumate și vizualizarea logicii proiectului.

Există două moduri care se pot utiliza pentru invocarea utilităților de proiectare individuale integrate în mediul Vivado Design Suite. Primul mod constă în invocarea acestor utilități din interfața grafică Vivado IDE; aceasta presupune crearea de către utilizator a unui proiect cu ajutorul interfeței grafice Vivado IDE și includerea fișierelor sursă în proiect. Interfața grafică Vivado IDE gestionează apoi fișierele sursă incluse în proiect, memorează rezultatele rulării, generează rapoarte ale proiectului și salvează în mod automat starea proiectului. Al doilea mod constă în utilizarea unor comenzi Tcl (*Tool Command Language*) sau a unor fișiere script Tcl. În acest mod, utilizatorul are un control total asupra fluxului de proiectare, dar diferitele utilități de proiectare nu gestionează în mod automat fișierele sursă și nu salvează în mod automat starea proiectului. Comenzile Tcl individuale pot fi introduse în panoul *Tcl Console* din interfața grafică Vivado IDE, iar fișierele script Tcl conținând secvențe de comenzi Tcl pot fi lansate în execuție din același panou. Este posibil să se utilizeze și interpretorul de comenzi Vivado Tcl, care se deschide în afara interfeței grafice Vivado IDE, pentru a introduce comenzi Tcl sau pentru a rula fișiere script Tcl. Aceste fișiere se pot utiliza pentru a executa întregul flux de proiectare sau pentru a executa părți ale fluxului de proiectare.

Figura 10 ilustrează fluxul de proiectare utilizând mediul de proiectare Xilinx Vivado Design Suite.

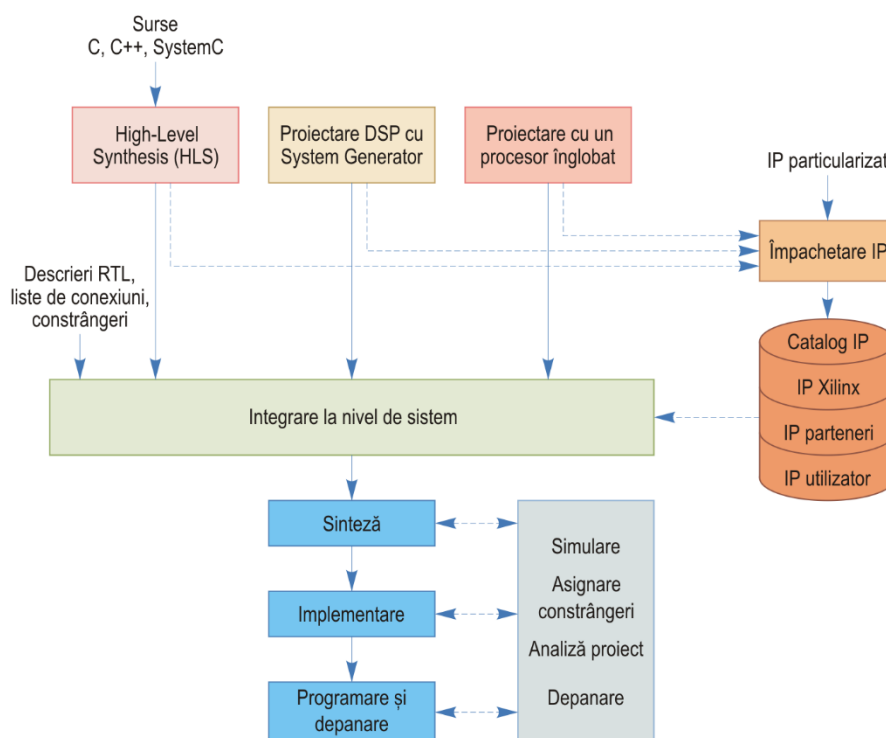


Figura 10. Fluxul de proiectare utilizând mediul de proiectare Xilinx Vivado Design Suite.

Mediul de proiectare Vivado Design Suite permite utilizarea unor fișiere sursă de diferite tipuri conținând descrierea sistemului digital care trebuie implementat. Un tip de fișier sursă conține descrierea sistemului digital sau a unei părți a acestuia la nivelul transferurilor între registre (RTL – *Register Transfer Level*). La acest nivel, sistemul constă din elemente interconectate cum sunt multiplexoare, decodificatoare, sumatoare, bistabile, registre, numărătoare și memorii. Descrierile RTL pot fi specificate în limbajele de descriere hardware VHDL, Verilog sau SystemVerilog.

Un alt tip de fișier sursă este reprezentat de o listă de conexiuni sintetizată, specificată fie în formatul standard EDIF, fie ca o descriere structurală Verilog. Un asemenea fișier poate fi creat cu un utilitar de sinteză al unui producător terț sau cu un alt utilitar de sinteză Xilinx, cum este utilitarul XST (*Xilinx Synthesis Technology*) al mediului de proiectare Xilinx ISE

Design Suite. Fișiere de constrângeri pentru specificarea cerințelor și a restricțiilor de proiectare pot fi, de asemenea, incluse ca fișiere sursă într-un proiect Vivado. Aceste fișiere pot fi specificate în formatul SDC (*Synopsys Design Constraints*), care este un standard industrial, și în formatul propriu al Xilinx, XDC (*Xilinx Design Constraints*), care este o extensie a formatului SDC.

Mediul de proiectare Vivado Design Suite permite adăugarea unor module IP (*Intellectual Property*) la proiect. Aceste module pot fi adăugate din catalogul IP pus la dispoziție de mediul Vivado Design Suite. Acest catalog conține module IP furnizate de Xilinx și partenerii săi și poate fi extins cu module IP particularizate create de utilizator și împachetate într-un format standard cu ajutorul utilitarului Vivado *IP Packager*. Se pot utiliza și module create cu ajutorul utilitarului CORE Generator al mediului de proiectare Xilinx ISE Design Suite. Modulele IP ale Xilinx, ale partenerilor și ale utilizatorului pot fi particularizate cu un utilitar de configurare. Un alt utilitar Vivado pus la dispoziție pentru a lucra cu module IP este *IP Integrator*, care permite crearea unor subsisteme complexe, numite proiecte bloc. Aceste proiecte bloc sunt de fapt subsisteme IP conținând module IP configurate de utilizator și conexiunile dintre aceste module IP.

În mediul de proiectare Vivado Design Suite sunt incluse și utilitare de proiectare de nivel mai înalt. Unul dintre acestea permite crearea unor sisteme înglobate bazate pe circuitul SoC Zynq-7000 sau pe procesorul Xilinx MicroBlaze. Se poate folosi utilitarul *IP Integrator* pentru instanțierea și configurarea modului procesorului, selectarea dispozitivelor periferice, configurarea setărilor hardware și conectarea acestor componente pentru a crea un sistem înglobat complex. Pentru interconectarea modulelor IP se utilizează magistrala AMBA AXI (*Advanced eXtensible Interface*). Magistrala AXI face parte din familia de magistrale ARM AMBA (*Advanced Microcontroller Bus Architecture*) dezvoltată de ARM Ltd. pentru microcontrolere. Această familie de magistrale este utilizată și pentru interconectarea blocurilor funcționale din circuitele SoC. Versiunea magistralei AXI care este utilizată este AXI4, care este inclusă în specificațiile AMBA 4.0. După implementarea proiectului hardware, acesta este exportat în mediul Xilinx Software Development Kit (SDK) pentru dezvoltarea și depanarea aplicației software care rulează pe procesorul înglobat.

Un alt utilitar care este integrat în mediul de proiectare Vivado Design Suite este *System Generator*, care permite utilizarea mediului de modelare și simulare Simulink al firmei MathWorks pentru proiectarea unor module de procesare digitală a semnalelor (DSP – *Digital Signal Processing*) implementate în circuite FPGA. Pentru aceste module DSP se poate utiliza limbajul de programare MATLAB (*Matrix Laboratory*), care este deosebit de util pentru calcule numerice. Fișiere existente conținând modele de proiecte *System Generator* se pot adăuga într-un proiect Vivado ca și fișiere sursă, iar noi module DSP pot fi create din interfața grafică Vivado IDE, care va lansa în execuție utilitarul *System Generator*. Un modul DSP poate fi împachetat ca și modul IP și poate fi adăugat la catalogul IP, putând fi integrat ulterior în proiect sau utilizat în proiecte viitoare.

Utilitarul *High-Level Synthesis* (HLS) permite descrierea diferiților algoritmi folosind limbajele C, C++ sau SystemC. După descrierea unui algoritm în unul din aceste limbaje, acesta poate fi compilat, poate fi executat pentru validarea faptului că algoritmul este corect din punct de vedere funcțional, iar apoi poate fi sintetizat într-un modul RTL. Modulul implementat poate fi analizat și depanat, iar apoi poate fi instanțiat în proiect sau poate fi împachetat într-un modul IP pentru a fi utilizat în proiect sau în proiecte viitoare. Utilitarul HLS pune la dispoziție mai multe facilități pentru generarea unei implementări optime a algoritmului.

3. Exemplu de utilizare a mediului Xilinx Vivado IDE

Această secțiune prezintă un exemplu de proiectare care este bazat pe limbajul VHDL și utilizează mediul de proiectare Xilinx Vivado IDE (*Integrated Design Environment*) pentru implementare. Exemplul ilustrează etapele fluxului de proiectare cu circuite FPGA și demonstrează principalele funcții oferite de mediul Xilinx Vivado IDE.

3.1. Descrierea proiectului

Proiectul implementează un ceas de timp real care păstrează și afișează timpul în ore, minute și secunde, având posibilitatea de setare a orei, a minutului și a secunde. Proiectul este destinat plăcii Nexys4 DDR și utilizează afișajul cu șapte segmente al acestei plăci. În exemplul de proiectare al ceasului de timp real, ceasul sistem este generat de oscilatorul de 100 MHz al plăcii. Schema bloc este ilustrată în figura 11.

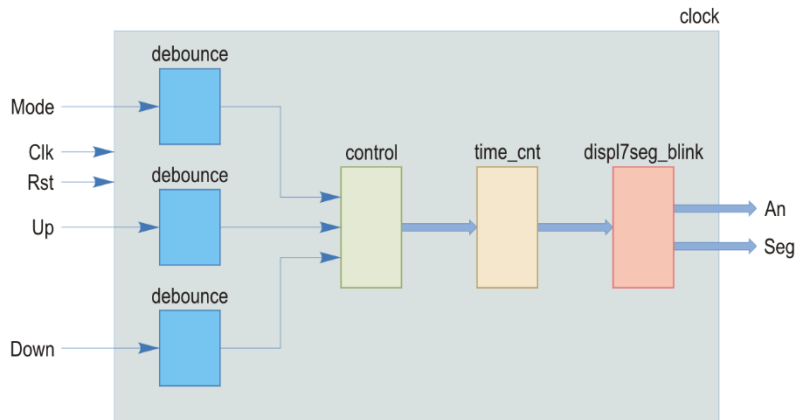


Figura 11. Schema bloc a ceasului de timp real.

Semnalele de intrare ale ceasului de timp real sunt următoarele:

- *Clk*: Ceasul sistem cu frecvența de 100 MHz.
- *Rst*: Semnal de resetare; inițializează ceasul și resetează ora la 12:00:00.
- *Mode*: Semnal de mod; trece ceasul în modul de setare a orei. Prin apăsarea repetată a butonului *Mode*, este posibilă setarea orei, a minutului și a secunde.
- *Up*: Semnal de incrementare; incrementează ora, minutul sau secunda în modul de setare a orei.
- *Down*: Semnal de decrementare; decrementează ora, minutul sau secunda în modul de setare a orei.

Semnalele de ieșire ale ceasului de timp real sunt următoarele:


- *An(7:0)*: Semnale de selecție a anodului activ al afișajului.
- *Seg(7:0)*: Semnale de selecție a catodilor (segmentelor) cifrei active a afișajului.

Proiectul constă în următoarele blocuri funcționale:

- **debounce**: Implementează un circuit simplu pentru eliminarea oscilațiilor semnalelor de intrare *Mode*, *Up* și *Down*.
- **control**: Implementează un automat de stare pentru generarea semnalelor de comandă necesare funcționării ceasului de timp real.
- **time_cnt**: Actualizează ceasul în timpul funcționării normale și în modul de setare a orei. Acest modul are șase ieșiri de câte 4 biți, reprezentând cifrele orei, minutului și secunde.
- **displ7seg_blink**: Reprezintă un controler pentru afișajul cu șapte segmente.

3.2. Crearea unui nou proiect

Lansați în execuție programul *Vivado IDE* selectând *Start* → *All Programs* → *Xilinx Design Tools* → *Vivado 2017.4* → *Vivado 2017.4*. Executați următoarele operații pentru a crea un nou proiect.


1. În fereastra *Vivado 2017.4*, secțiunea *Quick Start*, selectați **Create Project**. Se afișează fereastra de dialog *Create a New Vivado Project*.
2. Selectați butonul **Next** în fereastra *Create a New Vivado Project*. Se afișează fereastra de dialog *Project Name*.
3. În câmpul *Project name* introduceți numele proiectului (de exemplu, **clock**).
4. În câmpul *Project location* selectați butonul  pentru a naviga la directorul în care trebuie creat proiectul (acesta trebuie să fie un subdirector al directorului D:\Student\). Selectați directorul propriu creat pentru lucrările de laborator, după care selectați butonul **Select**. Verificați ca opțiunea **Create project subdirectory** să fie bifată, iar apoi selectați butonul **Next**. Se afișează fereastra de dialog *Project Type*.
5. Selectați tipul proiectului **RTL Project**, bifați opțiunea **Do not specify sources at this time**, iar apoi selectați butonul **Next**. Se afișează fereastra de dialog *Default Part*.
6. Selectați butonul **Boards**, iar în tabelul afișat selectați linia **Nexys4 DDR**. Prin aceasta, implementarea se va realiza pentru circuitul FPGA al plăcii Nexy4 DDR (circuitul Artix-7 xc7a100tcs324-1). Selectați butonul **Next**.
7. În fereastra *New Project Summary*, revizuiți setările cu care se va crea noul proiect și selectați butonul **Finish** pentru crearea proiectului.

3.3. Descrierea proiectului

În această etapă a fluxului de proiectare, se vor adăuga la proiect fișierele sursă ale exemplului de proiectare, se va crea un nou modul VHDL și acesta se va conecta la celelalte module.

3.3.1. Adăugarea fișierelor sursă și a fișierului de constrângeri

Fișierele sursă ale exemplului de proiectare sunt disponibile în arhiva clock.zip de pe pagina laboratorului. Adăugați fișierele sursă și fișierul de constrângeri la proiect după cum urmează:

1. Extrageți fișierele din arhiva clock.zip într-un director temporar.
2. În panoul *Flow Navigator* din partea stângă a ferestrei *Vivado 2017.4* selectați opțiunea **Add Sources** sau selectați iconița **Add Sources**  din panoul *Sources*. Se afișează fereastra de dialog *Add Sources*.
3. Selectați opțiunea **Add or create design sources** și selectați butonul **Next**. Se afișează fereastra de dialog *Add or Create Design Sources*.
4. Selectați butonul **Add Files**, navigați la directorul temporar, selectați toate fișierele cu extensia .vhd și apoi selectați butonul **OK**.
5. În aceeași fereastră de dialog *Add or Create Design Sources*, verificați ca opțiunea **Copy sources into project** să fie bifată și selectați butonul **Finish**. Fișierele selectate vor fi adăugate în proiect și se va actualiza ierarhia fișierelor sursă.
6. Selectați din nou opțiunea **Add Sources** din panoul *Flow Navigator* pentru a adăuga fișierul de constrângeri la proiect. În fereastra de dialog *Add Sources*, selectați opțiunea **Add or create constraints** și selectați butonul **Next**. Se afișează fereastra de dialog *Add or Create Constraints*.

7. Selectați butonul **Add Files**, selectați fișierul cu extensia .xdc din același director temporar și apoi selectați butonul **OK**.
8. În aceeași fereastră de dialog *Add or Create Constraints*, verificați ca opțiunea **Copy constraints files into project** să fie bifată și selectați butonul **Finish**. Fișierul selectat va fi adăugat în proiect și se va actualiza ierarhia fișierelor de constrângeri.
9. În panoul *Sources*, expandați ierarhia fișierelor sursă și a fișierelor de constrângeri din secțiunile *Design Sources*, respectiv *Constraints*, pentru a vizualiza fișierele care au fost adăugate în proiect. În acest panou se afișează și unitățile de proiectare care sunt adăugate în acest moment în proiect, împreună cu numele entității asociate. După cum se poate observa în figura 12, fiecare unitate de proiectare este reprezentată sub forma *nume_instanțiere : nume_entitate (nume_arhitectură) (nume_fișier)*.

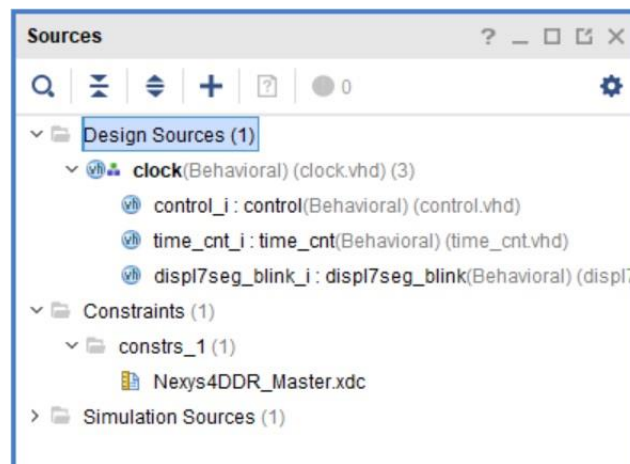


Figura 12. Ierarhia proiectului după adăugarea fișierelor sursă și a fișierului de constrângeri.

10. Folosiți utilitarul *File Explorer* pentru a vizualiza conținutul directorului în care a fost creat proiectul. Se poate observa că în acest director au fost create mai multe subdirectoare, printre care subdirectorul *clock.srscs*. Fișierele sursă adăugate în proiect pot fi regăsite în subdirectorul *clock.srscs\sources_1\imports\clock*, iar fișierul de constrângeri adăugat poate fi regăsit în subdirectorul *clock.srscs\constrs_1\imports\clock* (figura 13).

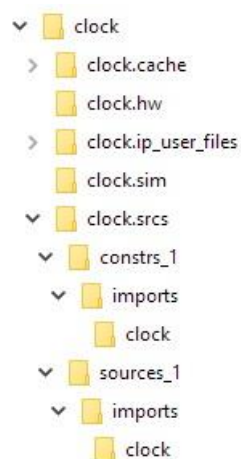


Figura 13. Structura de directoare generată la crearea proiectului și adăugarea fișierelor.

3.3.2. Crearea unui nou modul VHDL



În proiectul inițial al ceasului de timp real, semnalele de intrare *Mode*, *Up* și *Down* sunt conectate direct de la butoanele de pe placa Nexys4 DDR la modulul *control*. Pentru

eliminarea oscilațiilor semnalelor de intrare *Mode*, *Up* și *Down* generate de butoanele plăcii, se va crea un nou modul VHDL. Pentru a crea acest modul, parcurgeți etapele următoare.

1. Selectați opțiunea **Add Sources** din panoul *Flow Navigator*. În fereastra de dialog *Add Sources*, selectați opțiunea **Add or create design sources** și selectați butonul **Next**. Se afișează fereastra de dialog *Add or Create Design Sources*.
2. Selectați butonul **Create File**; se va afișa caseta de dialog *Create Source File*. În câmpul *File type* selectați opțiunea **VHDL**, în câmpul *File name* introduceți numele fișierului (de exemplu, *debounce*), iar apoi selectați butonul **OK**.
3. În fereastra de dialog *Add or Create Design Sources* selectați butonul **Finish**. Se va afișa fereastra de dialog *Define Module*.
4. În câmpul de sub linia *Port Name* introduceți numele portului de intrare **Clk**. Introduceți în mod similar numele porturilor **Rst**, **D_in** și **Q_out**. Modificați direcția portului *Q_out* la **out** și selectați butonul **OK**. Se va crea fișierul sursă *debounce.vhd*, care conține declarația de entitate și de arhitectură a modulului *debounce*.
5. Deschideți fișierul sursă *debounce.vhd* în fereastra de editare executând un clic dublu pe numele acestui fișier în panoul *Hierarchy* din fereastra *Sources*.

3.3.3. Adăugarea unui șablon al limbajului

Editorul de texte al mediului de proiectare Vivado IDE pune la dispoziție construcții ale limbajelor VHDL și Verilog pentru simulare și sinteză, reprezentând componente logice utilizate pe scară largă. În acest exemplu de proiectare se va utiliza construcția pentru sinteză *Debounce circuit*. Executați următoarele operații pentru a selecta șablonul necesar și pentru a adăuga acest șablon în fișierul sursă al modulului *debounce*.

1. Selectați iconița *Language Templates*  din meniul ferestrei de editare. Se va afișa fereastra *Language Templates*.
2. Expandați intrările *VHDL*, *Synthesis Constructs*, *Coding Examples* și *Misc* din fereastra *Language Templates*, iar apoi selectați șablonul numit *Debounce circuit*. Conținutul șablonului este afișat în panoul *Preview* al ferestrei *Language Templates*.
3. Selectați întregul text afișat în panoul *Preview*, executați un clic dreapta pe textul selectat și selectați opțiunea **Copy**.
4. Închideți fereastra *Language Templates*.
5. În fereastra de editare a fișierului *debounce.vhd*, inserați textul copiat anterior sub linia *begin*.
6. În fișierul *debounce.vhd* mutați linia cu definițiile semnalelor *Q1*, *Q2*, *Q3* astfel încât aceasta să fie plasată între cuvintele cheie *architecture* și *begin*.
7. Înlocuiți *<clock>* (inclusiv parantezele *<>*) cu *Clk* în lista de sensibilitate a procesului și în instrucțiunea *if*. Înlocuiți *<reset>* cu *Rst* în instrucțiunea *if*.
8. Salvați fișierul sursă selectând iconița *Save File*  din meniul ferestrei de editare. Fișierul va fi salvat în subdirectorul *clock.srcs/sources_1/new*.

3.3.4. Instanțierea modulului VHDL

În continuare, modulul VHDL *debounce* creat anterior va fi instanțiat în modulul de nivel superior și va fi conectat la modulele existente ale proiectului. Pentru instanțierea și conectarea modulului *debounce*, executați operațiile descrise în continuare.

1. În panoul *Hierarchy* al ferestrei *Sources* executați un clic dublu pe linia în care apare numele fișierului `clock.vhd` pentru a deschide fișierul în fereastra de editare.
2. În fișierul `clock.vhd` declarați trei noi semnale de tip `STD_LOGIC` după ultima declarație `signal` și denumiți aceste semnale `ModeD`, `UpD` și `DownD`. Inițializați fiecare din aceste semnale cu '0'. Semnalele vor reprezenta ieșirile modulelor pentru eliminarea oscilațiilor care vor fi instanțiate în modulul de nivel superior.
3. Instanțiați entitatea `debounce` de trei ori la sfârșitul fișierului (înainte de linia `end Behavioral`) și conectați porturile în modul ilustrat în figura 14.

```

mode_i: entity WORK.debounce port map (
    Clk => Clk,
    Rst => Rstp,
    D_in => Mode,
    Q_out => ModeD);


up_i: entity WORK.debounce port map (
    Clk => Clk,
    Rst => Rstp,
    D_in => Up,
    Q_out => UpD);

down_i: entity WORK.debounce port map (
    Clk => Clk,
    Rst => Rstp,
    D_in => Down,
    Q_out => DownD);

end Behavioral;

```

Figura 14. Instanțierea entității `debounce`.

4. În același fișier `clock.vhd`, în declarația de instanțiere a entității `control`, modificați semnalele `Mode`, `Up` și `Down` (care apar după indicatorul `=>`) în `ModeD`, `UpD`, respectiv `DownD` (nu modificați numele porturilor `ModeIn`, `UpIn` și `DownIn` care apar la stânga indicatorului `=>`). Prin aceste modificări, vor fi eliminate oscilațiile semnalelor de intrare înaintea conectării lor la modulul `control`.
5. Salvați fișierul `clock.vhd` selectând iconița *Save File* . În panoul *Hierarchy* al ferestrei *Sources* observați că instanțierile modulului `debounce` sunt adăugate în ierarhia proiectului (figura 15).

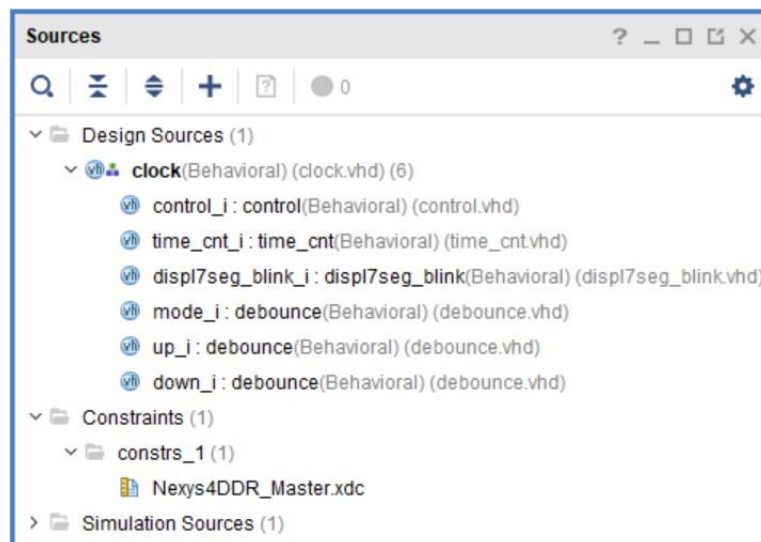




Figura 15. Ierarhia proiectului după adăugarea și instanțierea modulului `debounce`.

3.4. Elaborarea proiectului

În etapa de elaborare a proiectului, mediul de proiectare Vivado IDE compilează fișierele sursă ale proiectului, execută optimizări de nivel înalt, generează lista de conexiuni la nivelul RTL (*Register Transfer Level*) a proiectului, construiește structurile de date necesare etapelor următoare de proiectare și, în mod opțional, aplică unele constrângeri de proiectare. Elaborarea se realizează în mod automat în timpul sintezei proiectului, dar există posibilitatea invocării explicite a acestei etape pentru compilarea și verificarea sintactică a fișierelor sursă, explorarea listei de conexiuni și a schemei la nivelul RTL, verificarea unor reguli de proiectare și specificarea unor constrângeri.

Executați operațiile următoare pentru compilarea fișierelor sursă, vizualizarea schemei RTL și specificarea constrângerilor de plasare a porturilor de I/E.

1. În panoul *Flow Navigator*, selectați opțiunea **Open Elaborated Design** din secțiunea RTL ANALYSIS.
2. Dacă se afișează caseta de dialog *Elaborate Design*, selectați butonul **OK**. După terminarea elaborării proiectului, dacă nu sunt erori sintactice în fișierele sursă, se va afișa schema bloc a modulului de nivel superior al proiectului. Schema se poate afișa și dacă se selectează opțiunea *Tools* → *Schematic* din meniul principal sau se apasă tasta **F4**.
3. În fereastra *Schematic* puteți selecta iconița *Zoom In*  pentru a se putea distinge detaliile schemei. Se pot observa modulele care au fost instanțiate în modulul de nivel superior `clock` și unele componente suplimentare care au fost adăugate. Executați un clic dublu pe unul din modulele `debounce` pentru afișarea schemei interne a acestui modul.
4. Selectați iconița *Previous*  pentru a reveni la schema modulului de nivel superior. Selectați unul din module (de culoare albastră), executați un clic cu butonul din dreapta în interiorul modulului și selectați opțiunea **Go to Source** (sau apăsați tasta **F7**) pentru a deschide fișierul sursă în care este instanțiat modulul respectiv. Reveniți în fereastra *Schematic*, selectați unul din modulele RTL_ROM (de culoare galbenă) și apăsați tasta **F7** pentru a deschide fișierul sursă din care s-a generat acest modul.
5. Selectați panoul *Messages* din partea de jos a ecranului și parcurgeți mesajele generate în urma analizei fișierelor sursă și a elaborării proiectului. În acest panou se afișează și eventualele mesaje de eroare în urma compilării fișierelor sursă. Se poate selecta oricare mesaj de avertisment sau de eroare pentru a deschide fișierul sursă corespunzător în fereastra de editare.
6. Închideți fereastra *Schematic*.
7. În panoul *Flow Navigator*, secțiunea RTL ANALYSIS, selectați opțiunea **Report Noise**.
8. În caseta de dialog *Report Noise*, selectați butonul **OK** pentru generarea raportului `ssn_1`. Raportul va fi afișat în partea de jos a ecranului, secțiunea *I/O Bank Details* a raportului fiind selectată în mod automat.
9. În secțiunea *I/O Bank Details*, textul **Unplaced Ports** este afișat cu culoarea roșie (figura 16), motivul fiind faptul că nu s-a specificat asignarea porturilor la pinii circuitului. Se observă că sunt raportate doar porturile de ieșire `An` și `Seg`, deoarece analiza se realizează doar pentru porturile de ieșire. Cu toate acestea, este necesară și specificarea asignării porturilor de intrare la pinii circuitului.
10. În meniul principal Vivado 2017.4 selectați opțiunea *Window* → *Sources*, expandați secțiunea *Constraints* din fereastra *Sources* și deschideți pentru editare fișierul de constrângeri `Nexys4DDR_Master.xdc`.

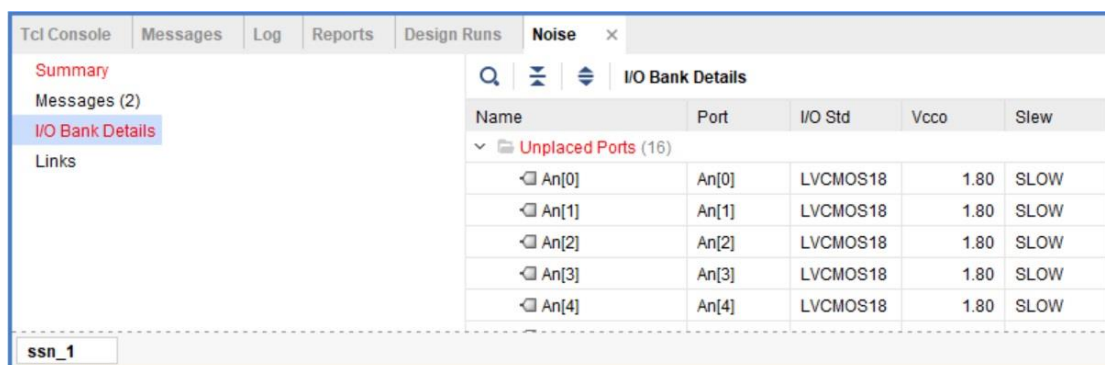


Figura 16. Panoul Noise.


11. În fișierul deschis se observă că toate liniile sunt comentate (fiecare linie începe cu caracterul #), acesta fiind un fișier general de constrângeri pentru placa Nexys4 DDR. Pentru specificarea asignării porturilor, trebuie șterse caracterele # de la începutul liniilor corespunzătoare porturilor utilizate în proiect și trebuie modificate numele porturilor astfel încât acestea să corespundă cu numele utilizate în proiect. Mai întâi, ștergeți caracterele # de la începutul celor două linii care se referă la semnalul de ceas și înlocuiți numele portului CLK100MHZ (care apare după comanda `get_ports`) cu Clk. Apoi, ștergeți caracterele # de la începutul liniilor corespunzătoare porturilor indicate în tabelul 2 și modificați numele acestor porturi conform acestui tabel.

Observație

- În fișierul de constrângeri se ține cont de tipul literelor, mari sau mici, din numele porturilor.

Tabelul 2. Numele porturilor care trebuie modificate în fișierul de constrângeri.

Nume inițial al portului	Nume modificat al portului
CLK100MHZ	Clk
CA	Seg[0]
CB	Seg[1]
CC	Seg[2]
CD	Seg[3]
CE	Seg[4]
CF	Seg[5]
CG	Seg[6]
DP	Seg[7]
AN[0] .. AN[7]	An[0] .. An[7]
CPU_RESETN	Rst
BTNU	Up
BTNL	Mode
BTND	Down

12. Salvați fișierul de constrângeri selectând iconița *Save File* , iar apoi închideți acest fișier. Se observă că sub bara de titlu a ferestrei *Elaborated Design* se afișează un mesaj indicând faptul că proiectul elaborat nu mai este la zi deoarece constrângerile au fost modificate. Selectați legătura **Reload** pentru procesarea constrângerilor introduse și reîncărcarea proiectului elaborat.
13. În panoul *Flow Navigator* selectați din nou opțiunea **Report Noise**, iar în caseta de dialog *Report Noise* selectați butonul **OK** pentru regenerarea raportului ssn_1. În raportul afișat nu ar mai trebui să apară erori (texte afișate cu culoarea roșie).

14. Închideți proiectul elaborat selectând opțiunea *File* → *Close Elaborated Design* din meniul principal și confirmați închiderea selectând butonul **OK** în caseta de dialog *Confirm Close*.

3.5. Setarea opțiunilor pentru sinteză și implementare

Opțiunile pentru sinteză și implementare permit modificarea comportamentului utilitatelor care execută sinteza și implementarea proiectului pentru a efectua optimizări conform cu anumite criterii. Exemple de criterii sunt timpul de execuție redus, performanța crescută a sistemului implementat sau spațiul redus ocupat în circuit. Aceste opțiuni pot fi specificate sub forma unor strategii de rulare pentru sinteză și implementare. O *strategie de rulare* reprezintă un set memorat de parametri, set care va fi utilizat în diferitele etape ale procesului de sinteză sau de implementare. Există un număr de strategii predefinite, separat pentru sinteză și pentru implementare, dar utilizatorul poate defini strategii proprii care vor fi utilizate la rulările ulterioare.

Procese de sinteză și de implementare vor fi executate pe baza unei *rulări de sinteză* și a unei *rulări de implementare*, care sunt create automat de mediul Vivado IDE și care utilizează strategiile de rulare selectate anterior. Aceste rulări create automat sunt afișate în fereastra *Design Runs* din partea de jos a ecranului. Utilizatorul poate crea rulări suplimentare de sinteză și de implementare, care utilizează strategii de rulare diferite cu scopul de a crea mai multe versiuni ale proiectului implementat. Rulările de sinteză și de implementare pot fi lansate în execuție în mod secvențial sau, dacă sunt disponibile mai multe nuclee de procesare, în mod concurrent.

Executați operațiile următoare pentru specificarea opțiunilor care vor fi utilizate pentru sinteza și implementarea proiectului.

1. În panoul *Flow Navigator*, selectați opțiunea **Settings**. Se va afișa panoul *General* din zona *Project Settings* a ferestrei de dialog *Settings*.
2. În panoul *General*, în dreptul liniei **Target language** selectați opțiunea **VHDL** în loc de *Verilog*.
3. În partea stângă a ferestrei de dialog *Settings*, selectați linia **Synthesis** din zona *Project Settings*.
4. În zona *Options*, vizualizați opțiunile disponibile pentru categoria *Strategy*. După ce parcurgeți aceste opțiuni, selectați strategia **Flow_RuntimeOptimized**. Se poate observa că o parte a parametrilor afișați în zona *Synth Design* sunt modificați în mod automat; aceștia sunt indicați printr-un asterisc (figura 17). Strategia de rulare selectată permite reducerea timpului de execuție a procesului de sinteză, cu toate că vor fi executate mai puține optimizări.
5. Revizuiți parametrii afișați în zona *Synth Design*. Parametrul *-flatten_hierarchy* indică modul în care este controlată ierarhia diferitelor module ale proiectului în timpul sintezei. Observați că opțiunile disponibile pentru acest parametru sunt *full*, *none* și *rebuilt*. Opțiunea *full* indică eliminarea completă a ierarhiei modulelor, păstrând doar modulul de nivel superior. Opțiunea *none* indică păstrarea ierarhiei proiectului; astfel, optimizarea va fi efectuată asupra modulelor separate, ceea ce va reduce complexitatea și va reduce timpul de procesare necesar utilitarului de sinteză. Opțiunea *rebuilt* indică utilitarului de sinteză să elimine ierarhia proiectului, să execute sinteza, iar apoi să reconstruiască ierarhia inițială a modulelor. Această opțiune permite efectuarea unor optimizări, având și avantajul că ierarhia inițială este păstrată, ceea ce permite o analiză mai simplă a proiectului. Pentru reducerea timpului de execuție, păstrați neschimbată opțiunea selectată pentru parametrul *-flatten_hierarchy* (**none**).
6. Parametrul *-fanout_limit* indică valoarea factorului de încărcare (*fanout*) maxim al semnalelor. Utilitarul de sinteză va limita factorul de încărcare la valoarea maximă specificată prin duplicarea porților cu factor de încărcare mare sau prin inserarea unor

buffere. Pentru acest exemplu de proiectare, păstrați neschimbată valoarea maximă a factorului de încărcare (10.000).

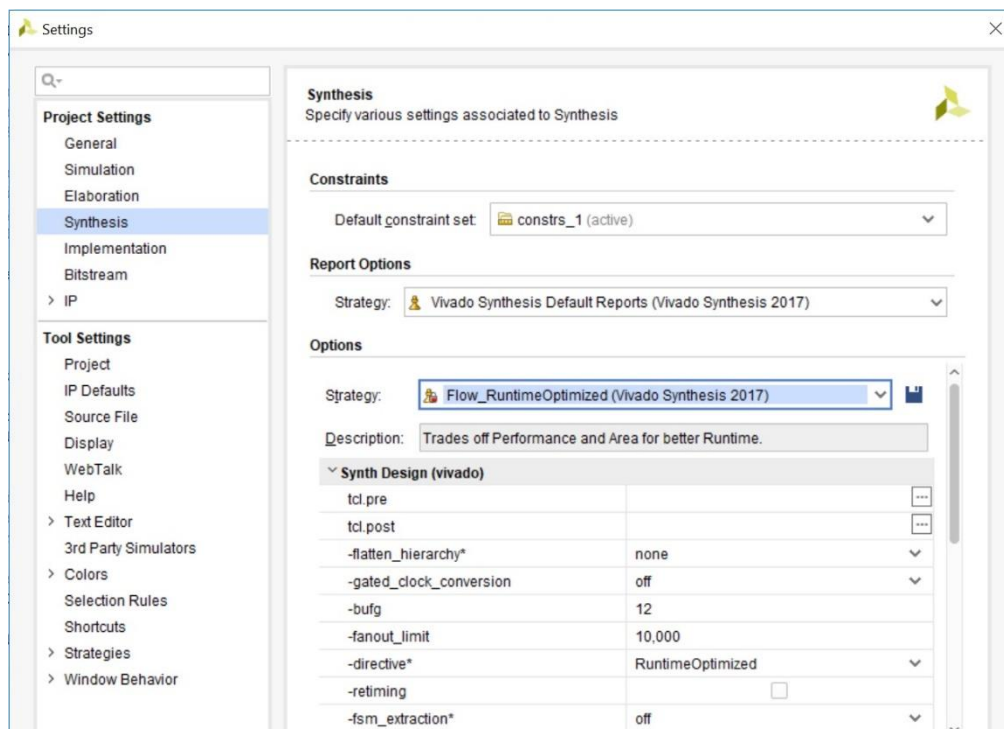


Figura 17. Setarea opțiunilor pentru sinteză.

7. Parametrul *-directive* permite rularea procesului de sinteză ținând cont de diferite optimizări. Acest parametru a fost modificat mod automat în *RuntimeOptimized* atunci când s-a selectat strategia de rulare **Flow_RuntimeOptimized**. Păstrați neschimbat acest parametru.
8. În partea stângă a ferestrei de dialog *Settings*, selectați linia **Implementation** din zona *Project Settings*. Se vor afișa opțiunile disponibile pentru implementarea proiectului.
9. În zona *Options*, vizualizați opțiunile disponibile pentru categoria *Strategy*. După ce parcurgeți aceste opțiuni, selectați strategia **Flow_Quick**; observați că o parte a parametrilor afișați pentru diferitele etape ale procesului de implementare sunt modificați în mod automat. Strategia de rulare *Flow_Quick* permite reducerea timpului de execuție a procesului de implementare, dar nu ține cont de constrângerile temporale.
10. Parcurgeți parametrii care sunt setați pentru diferitele etape de implementare, dar nu modificați acești parametri. Selectați butonul **OK** pentru a închide fereastra de dialog *Settings*.

3.6. Sinteza proiectului

După setarea opțiunilor pentru sinteză și implementare, se poate lansa în execuție una din comenzile următoare:

- *Run Synthesis*, prin care se va executa doar procesul de sinteză;
- *Run Implementation*, prin care se va executa mai întâi procesul de sinteză dacă acesta nu s-a executat în prealabil, după care se va executa procesul de implementare;
- *Generate Bitstream*, prin care se vor executa mai întâi procesele de sinteză și de implementare dacă acestea nu s-au executat în prealabil, după care se va genera fișierul conținând șirul de biți pentru configurarea circuitului FPGA.

În acest exemplu de proiectare, cele trei procese vor fi lansate în execuție separat.


În etapa de sinteză a proiectului, se execută mai întâi elaborarea proiectului, chiar dacă etapa de elaborare a fost executată anterior (deoarece rezultatele elaborării proiectului nu sunt salvate). În continuare, lista de conexiuni RTL generată la elaborarea proiectului, care este o listă de conexiuni generică și nu este specifică unei anumite arhitecturi, este transformată într-o listă de conexiuni care este optimizată pentru arhitectura circuitului FPGA care va fi utilizat pentru implementare.

3.6.1. Lansarea în execuție a sintezei proiectului

Pentru lansarea în execuție a procesului de sinteză a proiectului, în panoul *Flow Navigator* selectați opțiunea **Run Synthesis** din secțiunea SYNTHESIS (sau selectați opțiunea *Flow* → *Run Synthesis* din meniul principal). Dacă se afișează fereastra de dialog *Launch Runs*, selectați butonul **OK**. La terminarea sintezei, se afișează caseta de dialog *Synthesis Completed*, cu trei opțiuni disponibile. Selectați opțiunea *Open Synthesized Design*, iar apoi selectați butonul **OK**.

3.6.2. Analiza proiectului sintetizat

După deschiderea proiectului sintetizat, se afișează ierarhia listei de conexiuni generate în fereastra *Netlist* și structura circuitului FPGA în fereastra *Device*. Executați operațiile următoare pentru analiza proiectului sintetizat.

1. Folosiți utilitarul *File Explorer* pentru a verifica faptul că a fost creat subdirectorul `clock.runs` în directorul proiectului, iar în acest subdirector a fost creat subdirectorul `synth_1` conținând fișierele generate în urma sintezei.
2. În panoul *Flow Navigator*, selectați opțiunea **Schematic**, aflată în secțiunea *Open Synthesized Design*. Se va afișa schema circuitului sintetizat în fereastra *Schematic*.
3. În fereastra *Schematic*, selectați iconița *Zoom In*  pentru a putea distinge detaliile schemei. În schema generată se pot regăsi modulele care au fost instanțiate în modulul principal `clock`. Se poate observa însă că schema conține resurse disponibile în circuitul FPGA și nu module generice RTL. De exemplu, memoriile `RTL_ROM` din schema generată la elaborarea proiectului au fost înlocuite cu blocuri `LUT` (*Look-Up Table*) din circuitul FPGA (`LUT4` reprezintă blocuri `LUT` cu câte patru intrări). De asemenea, au fost adăugate în mod automat buffere de intrare `IBUF`, un buffer pentru semnalul de ceas `BUFG` și buffere de ieșire `OBUF`.
4. După vizualizarea schemei, închideți fereastra *Schematic*.
5. Selectați panoul *Reports* din partea de jos a ecranului. Dacă acest panou nu este vizibil, selectați opțiunea *Window* → *Reports* din meniul principal.
6. În panoul *Reports*, executați un clic dublu pe linia care conține textul **Vivado Synthesis Report** în coloana *Report Type* pentru deschiderea raportului de sinteză.
7. Parcurgeți raportul de sinteză și urmăriți care sunt componentele RTL generate pentru fiecare modul sintetizat. Închideți apoi raportul de sinteză.
8. În panoul *Flow Navigator*, selectați opțiunea **Report Timing Summary** din secțiunea *Open Synthesized Design*. În fereastra de dialog *Report Timing Summary*, selectați butonul **OK** pentru generarea raportului de timp `timing_1`. Se va deschide panoul *Timing* în partea de jos a ecranului.
9. În partea din stânga a panoului *Timing*, expandați intrarea *Check Timing*. Se poate observa că sunt afișate avertismente referitoare la faptul că nu sunt specificate întârzieri pentru semnalele aplicate la intrările circuitului FPGA și pentru semnalele de ieșire generate de circuitul FPGA. Aceste avertismente pot fi ignorate pentru prezentul exemplu de proiectare.

10. Închideți panoul *Timing*.
11. În panoul *Flow Navigator*, selectați opțiunea **Report Power** din secțiunea *Open Synthesized Design*. În fereastra de dialog *Report Power*, păstrați opțiunile implicite și selectați butonul **OK** pentru generarea raportului power_1.

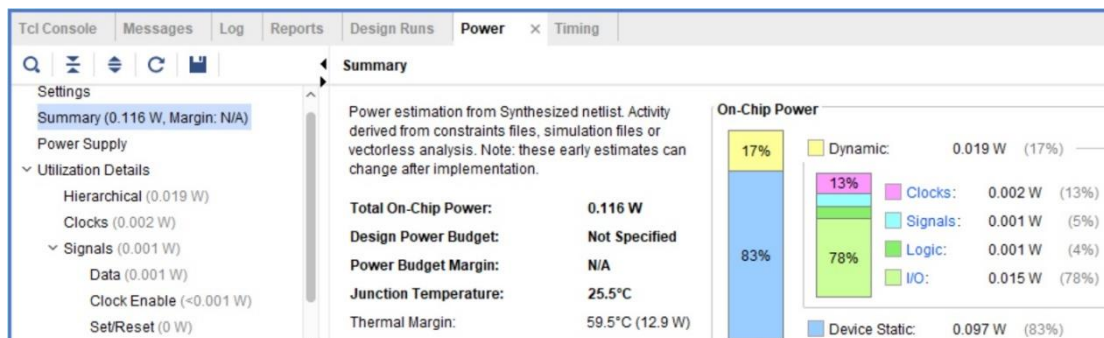


Figura 18. Panoul *Power*.

12. În panoul *Power* din partea de jos a ecranului se va afișa un raport cu puterea consumată estimată (figura 18). În secțiunea *Summary* a raportului se afișează un sumar cu puterea consumată de diferite elemente ale circuitului: semnale de ceas, alte semnale, blocuri logice și blocuri de I/E. De notat că valorile afișate reprezintă doar estimări realizate după sinteză și ele se pot modifica după implementare. Selectați unele categorii din partea stângă a panoului *Power* pentru a vizualiza detalii despre acestea.
13. Închideți panoul *Power*.

3.6.3. Modificarea opțiunilor de sinteză și resintetizarea proiectului

Rezultatele sintezei pot fi salvate într-un fișier al punctului de reper (*checkpoint*), fișier care poate fi încărcat ulterior pentru analiza proiectului, fără a fi necesară execuția din nou a sintezei. În continuare, se va salva rezultatul sintezei, se va modifica o opțiune de sinteză și se va executa din nou sinteza proiectului cu opțiunea modificată. Pentru aceasta, executați operațiile descrise în continuare.

1. În meniul principal, selectați opțiunea *File* → *Write Checkpoint* pentru salvarea într-un fișier a rezultatelor sintezei.
2. În caseta de dialog *Write Checkpoint*, se afișează numele implicit al fișierului cu extensia *.dcp* care va fi creat în directorul proiectului. Selectați butonul **OK** pentru generarea fișierului.
3. În panoul *Flow Navigator*, selectați opțiunea **Settings**.
4. În fereastra de dialog *Settings*, selectați linia **Synthesis** din zona *Project Settings*.
5. Pentru parametrul *-flatten_hierarchy* din zona *Synth Design* selectați opțiunea **full** pentru a elimina ierarhia proiectului, după care selectați butonul **OK**.
6. Se afișează caseta de dialog *Create New Run*, în care utilizatorul este interogat dacă dorește să se creeze o nouă rulare, deoarece proprietățile proiectului sintetizat au fost modificate. Selectați butonul **Yes** pentru confirmare.
7. În caseta de dialog *Create Run*, păstrați numele afișat în mod implicit și selectați butonul **OK**. În panoul *Design Runs* se poate observa că s-a creat o nouă rulare de sinteză (*synth_2*) și o nouă rulare de implementare (*impl_2*), iar aceste rulări sunt active.
8. Selectați opțiunea **Run Synthesis** din panoul *Flow Navigator* pentru lansarea în execuție a operației de sinteză. Dacă se afișează fereastra de dialog *Launch Runs*, selectați butonul **OK**.

9. La terminarea sintezei, se afișează caseta de dialog *Synthesis Completed*. Selectați opțiunea **Open Synthesized Design** dacă nu este selectată deja, iar apoi selectați butonul **OK** pentru reîncărcarea proiectului sintetizat.
10. În panoul *Flow Navigator*, selectați opțiunea **Schematic** din secțiunea *Open Synthesized Design*. Se va afișa schema circuitului sintetizat în fereastra *Schematic*.
11. Se poate observa că în schema afișată ierarhia a fost eliminată complet și nu se mai regăsesc modulele care au fost instanțiate în modulul principal.
12. În meniul principal, selectați opțiunea *File* → *Write Checkpoint* pentru salvarea rezultatelor sintezei.
13. În caseta de dialog *Write Checkpoint*, selectați butonul **OK** pentru generarea fișierului cu numele implicit și extensia *.dcp*.
14. Închideți proiectul sintetizat selectând opțiunea *File* → *Close Synthesized Design* din meniul principal și selectați butonul **OK** în caseta de dialog *Confirm Close*.
15. În meniul principal, selectați opțiunea *File* → *Open Checkpoint* pentru încărcarea primei versiuni a proiectului sintetizat din fișierul punctului de reper.
16. În fereastra de dialog *Open Checkpoint*, navigați la directorul proiectului, selectați fișierul *checkpoint_1.dcp* și selectați butonul **OK**.
17. Se afișează caseta de dialog *Open a Checkpoint*, în care se indică faptul că proiectul *clock* este deschis. Selectați butonul **Yes** pentru închiderea proiectului.
18. După deschiderea fișierului, se afișează ierarhia listei de conexiuni a proiectului în fereastra *Netlist* și structura circuitului FPGA în fereastra *Device*. În fereastra *Netlist*, selectați lista de conexiuni de nivel superior *clock* și apăsați tasta **F4** pentru afișarea schemei modulului corespunzător. Se observă că schema afișată corespunde primei versiuni a proiectului, în care este păstrată ierarhia modulelor.
19. Închideți proiectul încărcat din fișierul punctului de reper selectând *File* → *Close Checkpoint Design* din meniul principal și selectați butonul **OK** în caseta de dialog *Confirm Close*.
20. În fereastra de start Vivado 2017.4, deschideți proiectul *clock* selectând numele proiectului afișat în zona *Recent Projects*.
21. În panoul *Design Runs* a proiectului deschis, executați un clic cu butonul din dreapta pe linia **synth_1** și selectați opțiunea *Make Active*. Astfel, prima versiune a proiectului sintetizat va fi marcată ca fiind activă și se va putea continua implementarea acestei versiuni.

3.7. Implementarea proiectului

Procesul de implementare realizează plasarea și rutarea proiectului pe baza listei de conexiuni generate în urma sintezei. Mai întâi, se execută un sub-proces de optimizare a proiectului, după care se execută sub-procesele de plasare și de rutare. Opțional, după optimizarea proiectului se poate executa o optimizare a elementelor de proiectare pentru a se reduce puterea consumată de circuitul FPGA. De asemenea, în mod opțional, după plasare se poate executa o optimizare suplimentară pentru a se reduce puterea consumată și o optimizare a logicii și a plasării ținând cont de întârzierile estimate pe baza plasării anterioare. După rutare se poate executa și o optimizare a logicii, a plasării și a rutării ținând cont de întârzierile efective de rutare.

3.7.1. Lansarea în execuție a implementării proiectului

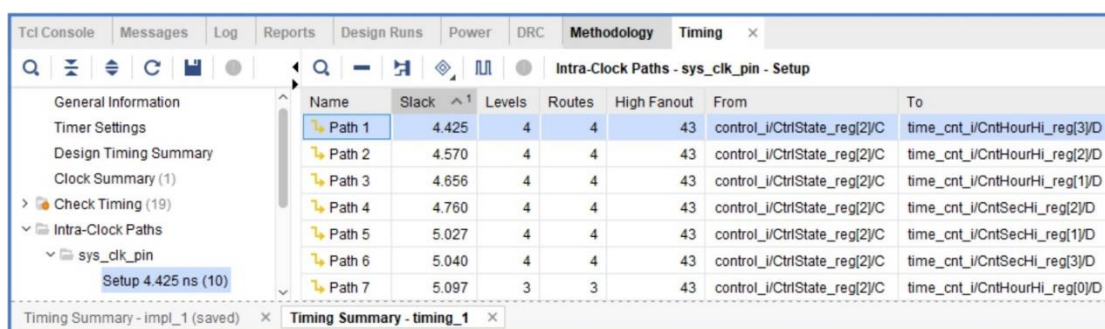
Pentru lansarea în execuție a procesului de implementare, în panoul *Flow Navigator* selectați opțiunea **Run Implementation** (sau selectați opțiunea *Flow* → *Run Implementation*

din meniul principal). Dacă se afișează fereastra de dialog *Launch Runs*, selectați butonul **OK**. În timpul procesului de implementare, se pot urmări mesajele afișate selectând panoul *Log* din partea de jos a ecranului. La terminarea implementării, se afișează caseta de dialog *Implementation Completed*, cu trei opțiuni disponibile. Selectați opțiunea *Open Implemented Design*, iar apoi selectați butonul **OK**.

3.7.2. Analiza proiectului implementat

După deschiderea proiectului implementat, se afișează ierarhia listei de conexiuni în fereastra *Netlist* și structura circuitului FPGA în fereastra *Device*. Executați operațiile următoare pentru analiza proiectului implementat.

1. Folosiți utilitarul *File Explorer* pentru a verifica faptul că în subdirectorul *clock.runs* a fost creat subdirectorul *impl_1* conținând fișierele generate în urma implementării.
2. În panoul *Flow Navigator*, selectați opțiunea **Report Utilization** din secțiunea *Open Implemented Design*. În caseta de dialog *Report Utilization*, selectați butonul **OK** pentru generarea raportului *utilization_1*. Se va deschide panoul *Utilization* în partea de jos a ecranului.
3. Selectați categoria *Slice LUTs* din partea stângă a panoului *Utilization* pentru a vizualiza numărul total de blocuri LUT utilizate și numărul de blocuri LUT utilizate de diferitele module ale proiectului. Selectați categoria *Slice Registers* pentru a vizualiza numărul total de registre utilizate și numărul de registre utilizate de diferitele module.
4. Închideți panoul *Utilization*.
5. În panoul *Flow Navigator*, selectați opțiunea **Report Timing Summary** din secțiunea *Open Implemented Design*. În fereastra de dialog *Report Timing Summary*, selectați butonul **OK** pentru generarea raportului de timp *timing_1*. Se va deschide panoul *Timing* în partea de jos a ecranului.
6. În partea stângă a panoului *Timing*, expandați intrarea *Intra-Clock Paths*, apoi expandați intrarea *sys_clk_pin* și selectați linia *Setup*. În partea dreaptă a panoului *Timing*, selectați una din căile afișate, de exemplu, **Path1** (figura 19). Calea selectată va fi evidențiată în fereastra *Device*.



Name	Slack	Levels	Routes	High Fanout	From	To
Path 1	4.425	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntHourHi_reg[3]/D
Path 2	4.570	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntHourHi_reg[2]/D
Path 3	4.656	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntHourHi_reg[1]/D
Path 4	4.760	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntSecHi_reg[2]/D
Path 5	5.027	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntSecHi_reg[1]/D
Path 6	5.040	4	4	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntSecHi_reg[3]/D
Path 7	5.097	3	3	43	control_i/CtrlState_reg[2]/C	time_cnt_i/CntHourHi_reg[0]/D

Figura 19. Panoul *Timing* cu calea de date *Path1* selectată.

7. Selectați iconița *Routing Resources* din meniul ferestrei *Device*, dacă nu este selectată deja. Astfel se vor afișa și conexiunile de rutare în fereastra *Device*. Urmăriți rutarea căii selectate în această fereastră.
8. În timp ce una din căile din panoul *Timing* este selectată, executați un clic cu butonul din dreapta în fereastra *Device* și selectați opțiunea *Schematic*. Se va deschide fereastra *Schematic*, în care se afișează schema căii selectate (figura 20).
9. În fereastra *Schematic*, fără a deselecta calea selectată, executați un clic cu butonul din dreapta într-o zonă liberă și selectați opțiunile *Expand Cone* → *To Flops or I/Os*

pentru a se afișa un context mai larg al căii selectate. Urmăriți blocurile logice din calea selectată.

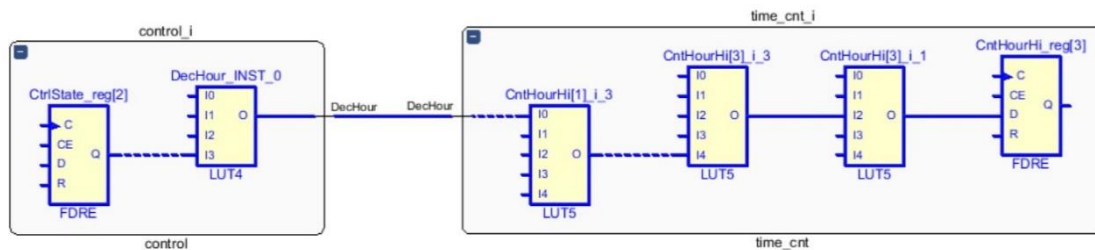


Figura 20. Schema căii de date Path1 selectate în panoul Timing.

10. Închideți proiectul implementat selectând opțiunea *File → Close Implemented Design* din meniul principal și selectați butonul **OK** în caseta de dialog *Confirm Close*.


3.8. Generarea fișierului de configurare

Pentru a genera fișierul cu șirul de biți necesar pentru configurarea circuitului FPGA executați operațiile descrise în continuare.

1. În panoul *Flow Navigator*, selectați opțiunea **Generate Bitstream** din secțiunea *PROGRAM AND DEBUG*. Dacă se afișează fereastra de dialog *Launch Runs*, selectați butonul **OK**.
2. În timp ce se execută generarea fișierului, selectați panoul *Tcl Console* din partea de jos a ecranului. Se poate observa că a fost lansată în execuție comanda `launch_runs` pentru rularea de implementare `impl_1` și comanda `write_bitstream`.
3. La terminarea generării fișierului, se afișează caseta de dialog *Bitstream Generation Completed*. Selectați butonul **Cancel** pentru închiderea casetei de dialog.

3.9. Configurarea circuitului și testarea funcționării

Pentru configurarea circuitului FPGA și testarea funcționării ceasului de timp real executați operațiile descrise în continuare.

1. Conectați o placă de dezvoltare Nexys4 DDR la un port USB al calculatorului.
2. Mutați comutatorul de alimentare POWER de pe placă în poziția ON.
3. În panoul *Flow Navigator*, selectați opțiunea **Open Hardware Manager** din secțiunea *PROGRAM AND DEBUG*. Se deschide fereastra *HARDWARE MANAGER* indicând starea neconectată a plăcii.
4. Selectați iconița *Auto Connect*  din panoul *Hardware*. Starea afișată în partea de sus a panoului ar trebui să se modifice în *Connected*.
5. Selectați legătura *Program device* din partea de sus a ferestrei *HARDWARE MANAGER*.
6. În caseta de dialog *Program Device*, verificați ca în câmpul *Bitstream file* să fie afișat fișierul cu extensia `.bit` din directorul proiectului, iar apoi selectați butonul **Program**. Fișierul de configurare va fi transmis la circuitul FPGA.
7. La terminarea configurării, se va aprinde dioda LED DONE de pe placă. Ceasul ar trebui să afișeze ora 12:00:00.
8. Apăsăți butonul BTNL pentru a trece ceasul în modul de setare. Cifrele orei trebuie să pâlpâie. Setați ora utilizând butonul BTNU (pentru incrementare) sau BTND (pentru decrementare).

9. Apăsați din nou butonul BTNL. Setări minutul utilizând butonul BTNU sau BTND. Procedați în mod similar pentru setarea secunde.
10. Apăsați butonul BTNL pentru a reveni în modul de funcționare normală.
11. După terminarea testării, închideți fereastra HARDWARE MANAGER și confirmați închiderea în caseta de dialog *Confirm Close*. Închideți apoi fereastra *Vivado 2017.4* și confirmați închiderea în caseta de dialog *Exit Vivado*.
12. Mutați comutatorul de alimentare al plăcii în poziția OFF și deconectați placa de la calculator.

4. Aplicații

4.1. Răspundeți la următoarele întrebări:

- a. Care sunt avantajele utilizării limbajelor de descriere hardware pentru proiectarea sistemelor digitale?
- b. Care este operația principală executată în etapa de sinteză a proiectului?
- c. Ce reprezintă maparea tehnologică?
- d. Care sunt operațiile executate la plasarea și rutarea proiectului?
- e. Care sunt utilitățile de proiectare de nivel înalt incluse în mediul de proiectare Xilinx Vivado Design Suite?

4.2. Executați etapele descrise în secțiunea 3 pentru exemplul de proiectare al ceasului de timp real.

4.3. Creați un nou proiect pentru afișarea unor caractere hexazecimale la afișajul cu șapte segmente al plăcii Nexys4 DDR. Utilizați modulul `displ7seg` disponibil pe pagina laboratorului în arhiva [displ7seg.zip](#) ca și multiplexor pentru afișajul cu șapte segmente. Creați un modul principal în care instanțiați entitatea `displ7seg` și aplicați la intrarea de date a acestuia un număr de 8 caractere hexazecimale. Folosiți ca și fișier de constrângeri același fișier utilizat în exemplul de proiectare al ceasului de timp real, în care comentați liniile corespunzătoare butoanelor, cu excepția butonului de resetare.