

PROIECTAREA CU CIRCUITE LOGICE PROGRAMABILE

1. Scopul lucrării

Lucrarea prezintă principalele tipuri de circuite programabile, etapele din cadrul procesului de proiectare utilizând circuite logice programabile, structura sistemului CAD Xilinx WebPACK și un exemplu simplu de proiectare utilizând acest sistem.

2. Considerații teoretice

2.1. Circuite logice programabile

Circuitele logice programabile, cunoscute și sub forma acronimului PLD (*Programmable Logic Device*), sunt circuite integrate care conțin un număr mare de porți sau celule a căror interconexiune poate fi configurată sau “programată” pentru a implementa orice funcție combinațională sau secvențială dorită. Pentru programarea circuitelor PLD se utilizează două tehnici: programarea prin măști, care se efectuează în timpul procesului de fabricație, sau programarea de către utilizator, pentru care se utilizează echipamente de programare cu costuri reduse. Multe circuite PLD pot fi reprogramate de utilizator de multe ori, motiv pentru care ele sunt avantajoase pentru realizarea prototipurilor unui nou produs.

Conexiunile programabile între elementele logice ale unui circuit PLD conțin comutatoare realizate de obicei cu tranzistoare sau antifuzibile (uneori fuzibile). Porțile logice programabile ale unui circuit PLD pot fi reprezentate în mod simplificat ca în Figura 6.1(b). În locul unor linii de intrare multiple la fiecare din aceste porți, ca în Figura 6.1(a), în reprezentarea simplificată s-a figurat o singură linie. Semnul $\times$ indică o conexiune programabilă a unei linii de intrare la o poartă logică. Absența semnului $\times$ indică faptul că respectiva conexiune a fost programată în starea deconectată.

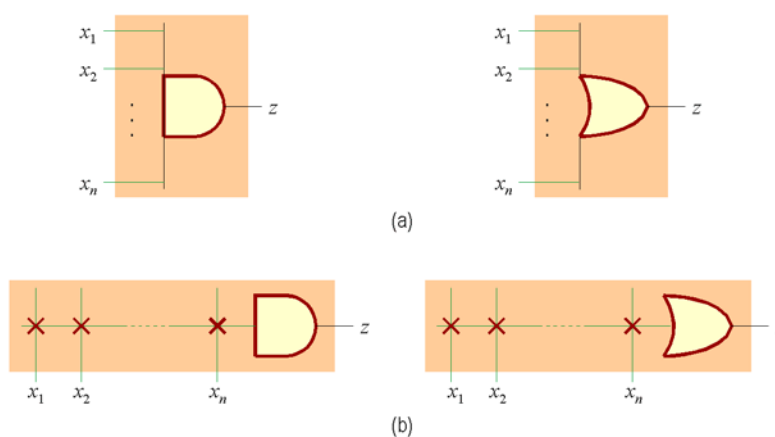


Figura 6.1. Porți ȘI, respectiv SAU: (a) reprezentare obișnuită; (b) reprezentare simplificată pentru circuitele PLD.

Există mai multe tipuri de circuite care sunt denumite în mod generic circuite logice programabile (PLD). Principalele tipuri sunt prezentate în continuare.

2.1.1. Rețele logice programabile

O rețea logică programabilă PLA (*Programmable Logic Array*) este similară ca și concept cu o memorie ROM, cu excepția faptului că nu realizează decodificarea completă a variabilelor și nu generează toți mintermi. Decodificatorul este înlocuit cu o rețea de porți ȘI care poate fi programată pentru a genera termenii produs ai variabilelor de intrare. Termenii produs sunt apoi conectați în mod selectiv cu porți SAU pentru a genera suma termenilor produs pentru funcțiile booleene necesare. Structura de bază a unui circuit PLA este prezentată în Figura 6.2.

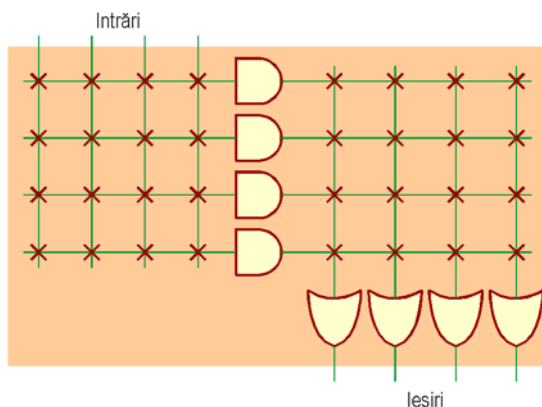


Figura 6.2. Structura generală a unui circuit PLA.

Un circuit PLA poate implementa în mod direct un set de funcții logice exprimate printr-un tabel de adevăr. Fiecare intrare pentru care valoarea funcției este adevărată necesită un termen produs, și acestuia îi corespunde o linie de porți ȘI din primul etaj al circuitului PLA. Fiecare ieșire corespunde la o linie de porți SAU din al doilea etaj al circuitului. Numărul de porți SAU corespunde cu numărul de intrări din tabela de adevăr pentru care ieșirea este adevărată. Dimensiunea totală a circuitului PLA este egală cu suma dintre dimensiunea rețelei de porți ȘI și dimensiunea rețelei de porți SAU. Din Figura 6.2 se observă că dimensiunea rețelei de porți ȘI este egală cu numărul de intrări multiplicat cu numărul diferiților termeni produs, iar dimensiunea rețelei de porți SAU este egală cu numărul de ieșiri multiplicat cu numărul termenilor produs.

Pentru proiectarea unui sistem digital cu un circuit PLA, nu este necesar să se indice conexiunile interne ale circuitului, ci trebuie să se specifice doar tabela de programare. Circuitele PLA pot fi programate prin măști (în timpul fabricației) sau pot fi programate de către utilizator. Circuitele PLA programate de către utilizator se numesc FPLA (*Field Programmable Logic Array*).

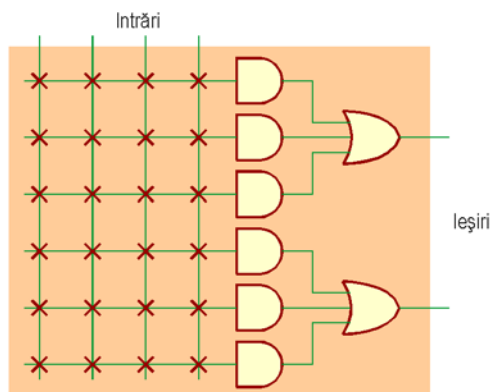


Figura 6.3. Structura generală a unui circuit PAL.

O altă categorie de rețele logice programabile sunt circuitele PAL (*Programmable Array Logic*), care conțin o rețea de porți ȘI programabilă, dar rețeaua de porți SAU are conexiuni fixe (Figura 6.3). Fiecare linie de ieșire este conectată la un set fix de linii ale rețelei de porți ȘI. O asemenea ieșire

a circuitului PAL poate implementa o expresie pe două nivele conținând cel mult opt termeni. Avantajele circuitelor PAL sunt simplitatea utilizării în anumite aplicații și viteza mai ridicată. Aceste circuite sunt însă mai puțin flexibile decât circuitele PLA.

Există și circuite PLA sau PAL care conțin bistabile atașate prin conexiuni programabile la ieșirile rețelei de porți SAU, ceea ce permite implementarea unor circuite secvențiale de dimensiuni medii. Aceste circuite sunt cunoscute și cu denumirea de circuite logice programabile simple (SPLD - *Simple Programmable Logic Device*).

2.1.2. Circuite CPLD

Circuitele logice programabile complexe (CPLD - *Complex Programmable Logic Device*) sunt circuite PLD având o densitate mai ridicată. Ele conțin un număr de blocuri funcționale, asemănătoare unor circuite PLD, fiecare bloc fiind compus din mai multe *macrocelule* (Figura 6.4). Există de asemenea o matrice de rutare pentru interconectarea blocurilor. Funcțiile logice simple pot fi implementate în cadrul unui singur bloc. Funcțiile mai complexe pot necesita mai multe blocuri, care vor fi interconectate prin matricea de rutare.

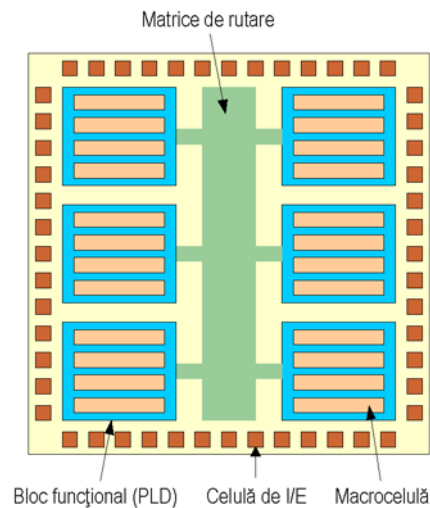


Figura 6.4. Structura generală a unui circuit CPLD.

2.1.3. Circuite FPGA

Circuitele FPGA (*Field Programmable Gate Array*) au fost introduse în anul 1985 de compania Xilinx. De atunci au fost elaborate diferite tipuri de circuite FPGA de un număr de alte companii ca Actel, Altera, Atmel, Texas Instruments etc. Un circuit FPGA constă dintr-o rețea bidimensională de celule sau blocuri logice. De obicei, fiecare bloc logic poate fi programat pentru a implementa orice funcție logică a intrărilor sale. De aceea, aceste blocuri sunt numite de obicei blocuri logice configurabile (*Configurable Logic Block* - CLB). Cele mai multe blocuri logice conțin de asemenea unul sau două bistabile. Canalele și blocurile de comutare dintre aceste blocuri conțin resurse de interconectare, după cum se ilustrează în Figura 6.5. Aceste resurse conțin de obicei segmente de interconectare de diferite lungimi. Interconexiunile conțin comutatoare programabile cu rolul de a conecta blocurile logice la segmentele de interconectare, sau un segment de interconectare la altul. În plus, există celule de I/E la periferia rețelei, care pot fi programate ca intrări sau ieșiri.

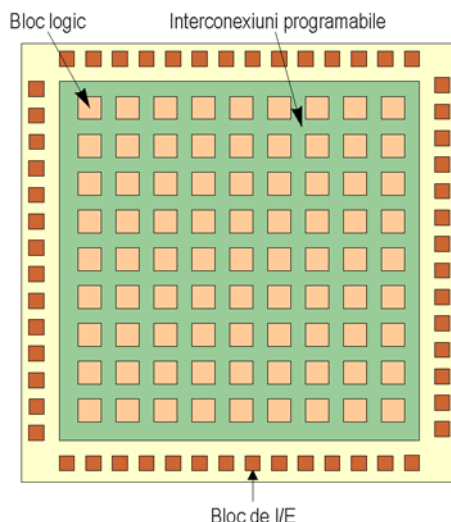


Figura 6.5. Structura unui circuit FPGA tipic.

Din punctul de vedere al tipului conexiunilor programabile, există două categorii principale de circuite FPGA: circuite cu memorii SRAM și circuite cu antifuzibile.

Circuite cu memorii SRAM. Programarea acestor circuite se realizează prin celule de memorie statică. Logica este implementată cu ajutorul unor tabele (*lookup table*) realizate din celulele de memorie, intrările funcțiilor controlând liniile de adresă. Fiecare tabelă de 2^n celule de memorie implementează orice funcție cu n intrări. Una sau mai multe tabele, combinate cu bistabile, formează un bloc logic configurabil. Aceste blocuri sunt aranjate într-un tablou bidimensional, segmentele de interconectare formând canale, similar cu rețelele de porți. Segmentele se conectează la pinii blocurile logice din canale și la alte segmente din blocurile de comutare prin intermediul tranzistoarelor de trecere controlate de celule ale memoriei de configurare.

O secvență de configurare pentru circuitele cu memorii SRAM constă dintr-un singur cuvânt lung de programare. Logica din circuit încarcă cuvântul de programare, pe care îl citește serial dintr-o memorie externă de fiecare dată când circuitul este alimentat. Biții acestui cuvânt setează valorile tuturor celulelor memoriei de configurare din circuit, setând astfel valorile tabelor și selectând segmentele care se vor conecta între ele. Circuitele cu memorii SRAM sunt reprogramabile. Ele pot fi actualizate în sistem, punând la dispoziția proiectanților noi opțiuni și posibilități de proiectare.

Din această categorie de circuite FPGA fac parte cele ale firmelor Xilinx, Altera, AT&T.

Circuite cu antifuzibile. Un antifuzibil este un dispozitiv cu două terminale care în mod normal se află în starea de înaltă impedanță, iar atunci când este expus la o tensiune ridicată, trece în starea cu rezistență redusă (300-500 Ω). Antifuzibilele au dimensiuni reduse, astfel încât o arhitectură bazată pe antifuzibile poate conține sute de mii sau milioane de antifuzibile. Pentru simplificarea arhitecturii și a programării, circuitele FPGA bazate pe antifuzibile constau de obicei din rânduri de elemente logice configurabile cu canale de interconectare între ele, ca și rețelele de porți tradiționale. Un bloc logic poate fi programat prin conectarea pinilor săi de intrare la valori fixe sau la rețele de interconectare. Există antifuzibile la fiecare punct de intersecție între interconexiuni și pini din canal și la toate punctele de intersecție între interconexiuni în locurile în care canalele se intersectează.

Din categoria circuitelor FPGA cu antifuzibile fac parte circuitele firmelor Actel, Quicklogic, Cypress.

2.2. Procesul de proiectare cu circuite programabile

2.2.1. Fluxul de proiectare

Pentru proiectarea sistemelor digitale utilizând circuite programabile, cum sunt circuitele FPGA și CPLD, se utilizează pachete de programe de proiectare asistată de calculator (CAD – *Com-*

puter 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:

- *Specificarea* (descrierea) sistemului digital;
- *Sinteza* descrierii, deci transformarea acesteia într-o listă de conexiuni conținând porți elementare și interconexiunile dintre ele;
- *Simularea* funcționării 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 a se utiliza în mod eficient resursele disponibile ale circuitului;
- *Configurarea* (programarea) circuitului pentru ca acesta să realizeze funcția dorită.

Figura 6.6 ilustrează etapele din cadrul procesului de proiectare a sistemelor digitale utilizând circuite programabile.

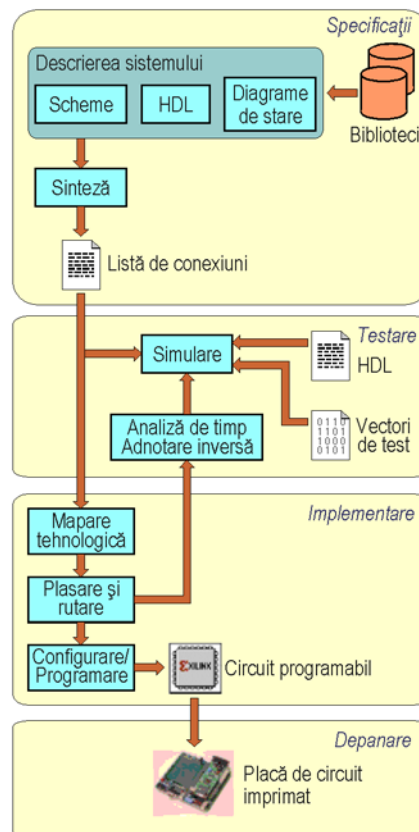


Figura 6.6. Fluxul de proiectare a sistemelor digitale utilizând circuite programabile.

În continuare sunt descrise mai detaliat principalele etape de proiectare.

2.2.1.1. Descrierea sistemului

Există mai multe metode pentru descrierea sistemelor digitale. În figura 6.6 sunt indicate principalele metode: 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 aceasta se utilizează un editor schematic, care permite specificarea componentelor care trebuie utilizate și a modului în care acestea trebuie interconectate. Această metodă este cea care va fi utilizată în primul rând în

lucrările următoare și este ilustrată în Figura 6.7. Circuitul din această figură detectează secvența binară 1010 aplicată la intrarea X. La detectarea acestei secvențe, ieșirea Z va fi setată la 1 logic.

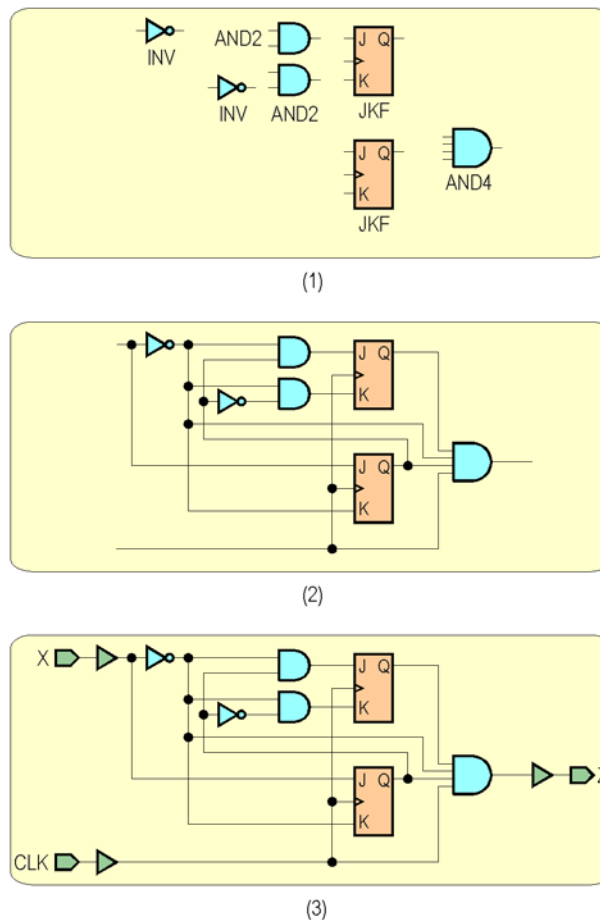


Figura 6.7. Etapele descrierii unui sistem digital utilizând scheme: (1) Selectarea și amplasarea componentelor; (2) Conectarea componentelor; (3) Adăugarea porturilor de I/E.

Există următoarele etape principale la utilizarea schemelor logice pentru proiectarea unui sistem digital:

1. În cadrul unui editor schematic se selectează componentele necesare dintr-o bibliotecă de componente. Asemenea componente pot fi, de exemplu, porți elementare, multiplexoare, decodificatoare, numărătoare, circuite 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 programabile și diferitelor familii de circuite. Circuitele dintr-o anumită familie diferă prin capacitatea lor, viteza și capsula utilizată. În această etapă, nu este însă necesară specificarea exactă a circuitului care va utilizat dintr-o anumită familie.
2. Componentele selectate și plasate în cadrul schemei 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 cu ajutorul limbajelor de descriere hardware. Aceste limbaje sunt din ce în ce mai utilizate, fiind preferate pentru descrierea sistemelor cu complexitate mai ridicată, datorită următoarelor avantaje principale:

- Posibilitatea descrierii funcționale a sistemelor, aceasta fiind o descriere la un nivel mai înalt, fără detalierea structurii la nivelul componentelor simple sau a porților elementare. Astfel, timpul necesar pentru descrierea sistemelor complexe se reduce în mod semnificativ.
- Independența descrierilor HDL față de diferitele 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 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.

```
entity detect is
  port (X, CLK: in STD_LOGIC;
        Z: out STD_LOGIC);
end detect;
architecture automat_st of detect is
  type tip_stare is (A, B, C, D);
  signal stare: tip_stare;
begin
  tranz_st: 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_st;
  Z <= '1' when stare = D else '0';
end automat_st;
```

Figura 6.8. Descrierea în limbajul VHDL a circuitului din Figura 6.7.

Există diferite limbaje de descriere hardware, dar mai utilizat este limbajul VHDL (*VHSIC Hardware Description Language*), VHSIC fiind acronimul pentru *Very High Speed Integrated Circuit*. Pe lângă acest limbaj, pentru proiectarea cu circuite FPGA se mai utilizează limbajul Verilog. Pentru proiectarea cu circuite CPLD, un limbaj utilizat în mod frecvent este ABEL (*Advanced Boolean Expression Language*). Limbajele VHDL și Verilog sunt standardizate de institutul IEEE. Figura 6.8 prezintă o descriere posibilă în limbajul VHDL a circuitului ilustrat în Figura 6.7.

Pentru descrierea automatelor cu stări finite se utilizează pe scară largă diagramele de stare. Sistemele CAD pentru proiectarea cu circuite programabile conțin de obicei 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. În Figura 6.9 se prezintă o dia-

gramă de stare echivalentă cu circuitul ilustrat în Figura 6.7 și descrierea în limbajul VHDL din Figura 6.8.

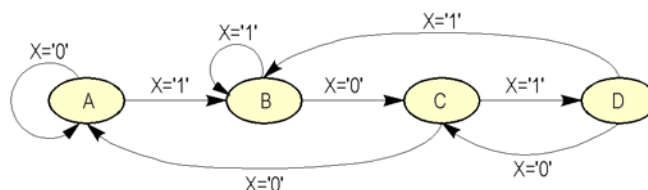


Figura 6.9. Diagrama de stare echivalentă cu circuitul reprezentat prin schema din Figura 6.7 și descrierea în limbajul VHDL din Figura 6.8.

2.2.1.2. 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 din cadrul 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. Această listă este prelucrată de celelalte componente ale sistemului CAD pentru realizarea etapelor următoare din cadrul procesului de proiectare.

Există diferite formate pentru listele de conexiuni, cel mai utilizat fiind formatul EDIF (*Electronic Digital Interchange Format*), acesta reprezentând 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 formatul propriu al firmei Xilinx, cel mai important producător de circuite programabile de tip FPGA și CPLD. 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 6.10. În prima parte a listei de conexiuni sunt declarate componentele din cadrul schemei, iar în a doua parte sunt specificate conexiunile dintre componente. Denumirile componentelor sunt G1..G7, iar denumirile conexiunilor sunt N1..N10. Aceste denumiri sunt fie cele specificate de proiectant, fie cele asignate în mod automat de sistemul CAD.

În circuitul ilustrat în Figura 6.10 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. Pentru simplitate, pinii și semnalele de alimentare și masă au fost omiși în această figură. O conexiune este indicată 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.

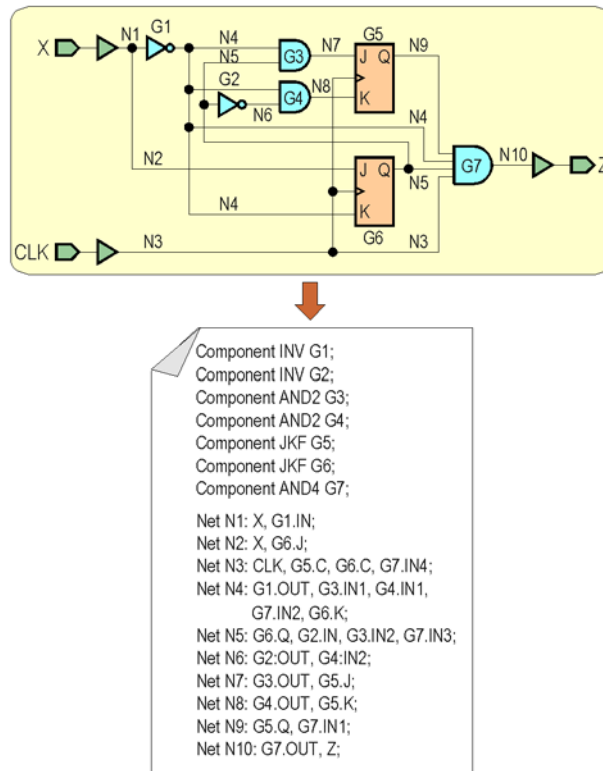


Figura 6.10. Relația dintre schema logică și lista de conexiuni pentru circuitul din Figura 6.7.

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

2.2.1.3. Simularea funcțională

În această etapă se utilizează un program simulator pentru verificarea funcționării sistemului proiectat, înainte de implementarea acestuia într-un circuit 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 fiind numită *vector de test*. De asemenea, proiectantul poate specifica valorile semnalelor de ieșire care trebuie generate de sistem pentru fiecare vector de test.

Simulatorul aplică pe rând câte un vector de test la intrările sistemului, determină semnalele de ieșire care sunt generate de sistem și le compară cu valorile acestor semnale 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 6.11 ilustrează modul în care pot fi vizualizate pe ecranul calculatorului semnalele de intrare și de ieșire ale circuitului detector de secvență utilizat ca exemplu în secțiunile precedente la simularea funcțională a circuitului.

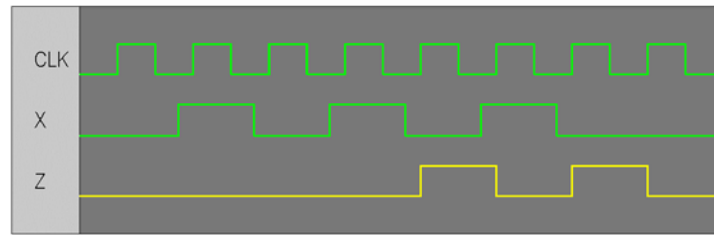


Figura 6.11. Semnalele de intrare și de ieșire ale circuitului din Figura 6.7 vizualizate la simularea funcțională a circuitului.

2.2.1.4. Maparea tehnologică

Etapile următoare din cadrul procesului de sinteză realizează implementarea sistemului proiectat într-un circuit programabil (FPGA sau CPLD). Prima etapă din cadrul implementării este cea de mapare tehnologică. Această etapă constă dintr-o serie de operații care realizează prelucrarea listei de conexiuni și adaptarea acesteia la particularitățile și resursele disponibile ale circuitului utilizat pentru implementare. Operațiile executate în această etapă diferă în funcție de sistemul de proiectare. 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 acesteia etape, proiectantul selectează tipul circuitului 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 tuturor programelor executate. Pe lângă mesaje de eroare și de avertizare, se crează de obicei o listă cu resursele utilizate din cadrul circuitului.

Figura 6.12 ilustrează etapa de mapare tehnologică pentru circuitul utilizat ca exemplu. După cum se observă, schema circuitului a fost modificată pentru a utiliza bistabilele 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 6.12 fiind doar ilustrativă.

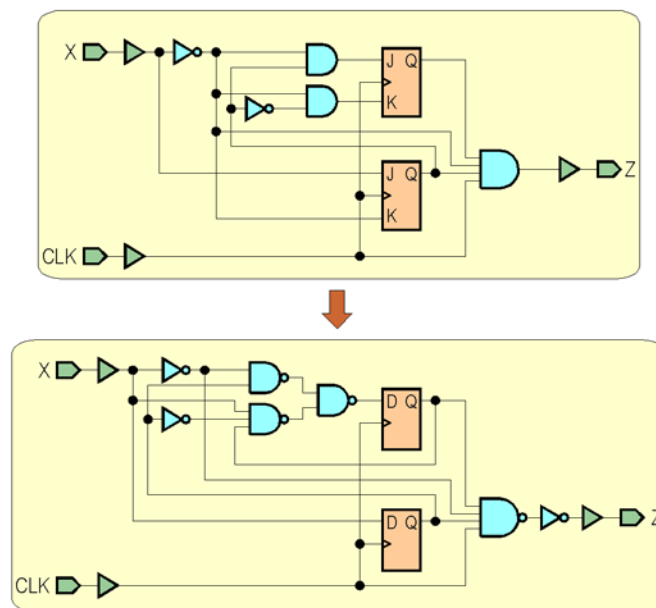


Figura 6.12. Ilustrarea etapei de mapare tehnologică pentru circuitul din Figura 6.7.

2.2.1.5. Plasarea și rutarea

Aceste operații sunt executate în cazul utilizării unui circuit FPGA pentru implementare. Pentru proiectarea cu circuite CPLD, operația echivalentă este numită adaptare ("fitting"). *Plasarea* este procesul de selectare a unor module sau blocuri logice ale circuitului 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 utilizat pentru implementare. Anumite sisteme permit utilizatorilor experți plasarea și rutarea manuală a unor porțiuni critice ale sistemului digital pentru a obține performanțe superioare.

Figura 6.13 ilustrează plasarea și rutarea circuitului rezultat 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 schemei. 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 (indicate în figură prin cercuri), iar liniile de rutare sunt interconectate cu ajutorul unor comutatoare programabile.

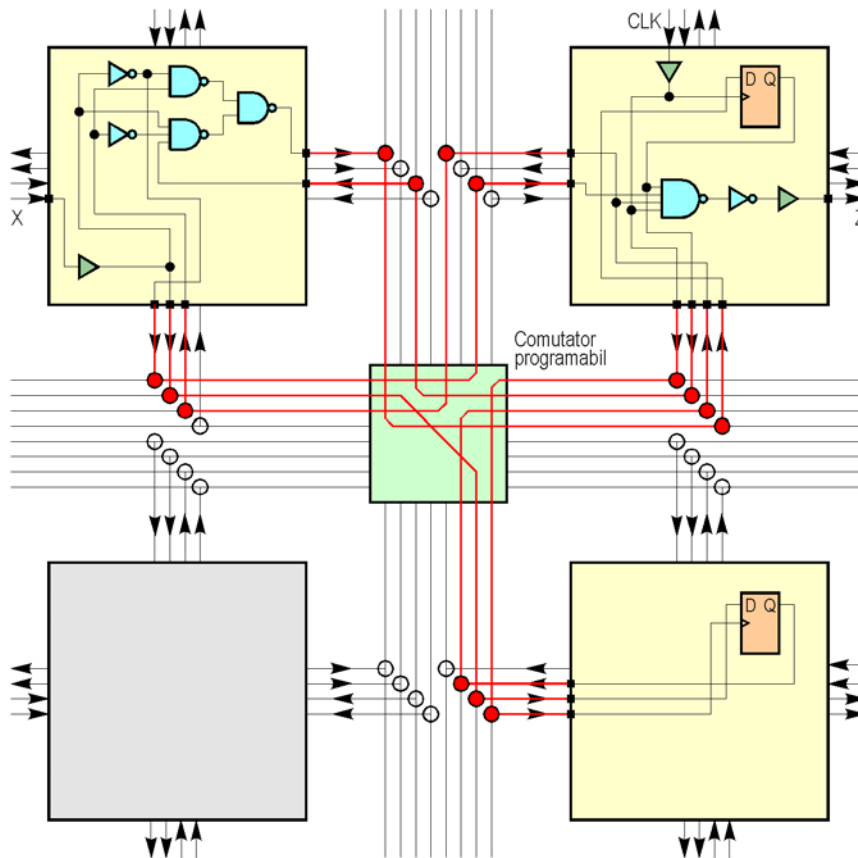


Figura 6.13. Ilustrarea etapelor de plasare și rutare pentru circuitul din Figura 6.12.

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 cadrul circuitului programabil, interconectarea corectă a acestora și verificarea faptului că sunt asigurate cerințele de performanță specificate în timpul proiectării.

2.2.1.6. 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*”).

2.2.1.7. Configurarea sau programarea circuitului

Operația de configurare se referă la circuitele programabile bazate pe memorii volatile SRAM (*Static Random Access Memory*) și constă din încărcarea informațiilor de configurare în memoria circuitului. Operația de programare se referă la circuitele 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, în principiu, un șir de biți (“*bitstream*”), fiecare bit indicând starea închisă sau deschisă a unui comutator. Circuitele programabile conțin un număr mare de asemenea comutatoare, un comutator fiind realizat sub forma unui tranzistor sau a unei celule de memorie. Un bit de 1 din șirul de biți va determina închiderea unui comutator și, deci, stabilirea unei conexiuni. Biții din acest fișier 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 programabil, aflat de obicei pe o placă de circuit imprimat împreună cu alte circuite. Comutatoarele circuitului se închid sau rămân deschise în funcție de valorile biților din șirul de configurare. După terminarea configurării, circuitul va funcționa conform descrierii sistemului digital care a fost implementat.

Din cauza memoriei volatile, circuitul trebuie configurat din nou după fiecare întrerupere a tensiunii de alimentare. Informațiile de configurare pot fi păstrate într-o memorie nevolatilă PROM (*Programmable Read Only Memory*), existând posibilitatea configurării automate a circuitului din această memorie nevolatilă la aplicarea tensiunii de alimentare.

Configurarea sau programarea se pot realiza utilizând interfața paralelă a calculatorului. Pentru aceasta, este necesar ca placa cu circuitul programabil să conțină un conector pentru interfața paralelă, pentru transfer utilizându-se un cablu paralel. Figura 6.14 ilustrează un cablu paralel obișnuit, care conține conectori cu 25 de contacte DB25.

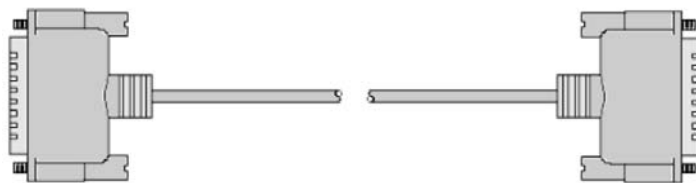


Figura 6.14. Cablu paralel care poate fi utilizat pentru configurarea circuitelor programabile.

O altă posibilitate este utilizarea unui cablu special și a unei metodologii de configurare propuse de organizația JTAG (*Joint Test Advisory Group*). Această metodologie, cunoscută și sub numele de “*Boundary-Scan*”, a fost standardizată de institutele IEEE (*Institute of Electrical and Electronic Engineers*) și ANSI (*American National Standards Institute*) 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 a

calculatorului, iar celălalt capăt se conectează la un număr de 5 pini speciali de pe placa circuitului programabil. Informațiile de configurare sunt preluate în paralel de la calculator și sunt transferate serial (bit cu bit) la circuitul 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 programabil în timpul funcționării, transferul acestora la calculator și vizualizarea lor pe ecran.

Figura 6.15 ilustrează un cablu JTAG al firmei Xilinx (*Parallel Cable IV*). Există mai multe variante de cabluri JTAG produse de această firmă. Cablul *MultiLIX* poate fi conectat fie la interfața serială RS232 a calculatorului, fie la interfața USB, prin intermediul unui cablu serial sau al unui cablu USB. Dispozitivul *MultiPRO* poate fi utilizat atât ca și cablu de configurare, cât și ca programator pentru memorii PROM și circuite CPLD CoolRunner II cu ajutorul unor adaptoare. Cablurile *Parallel Cable III* și *Parallel Cable IV* se conectează la interfața paralelă a calculatorului. Cablul *Parallel Cable IV* permite o rată de transfer superioară comparativ cu cablul *Parallel Cable III* (de până la 5 MB/s față de 500 KB/s).

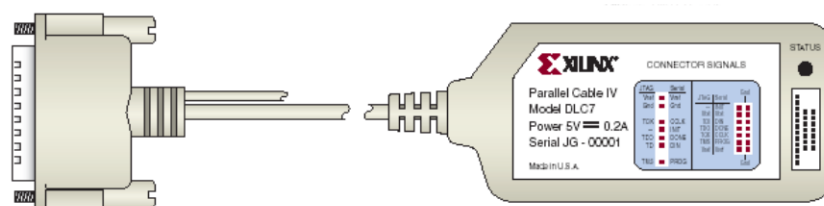


Figura 6.15. Cablul paralel JTAG *Parallel Cable IV* al firmei Xilinx.

Figura 6.16 ilustrează conectarea unui cablu JTAG în modul JTAG (sau “*Boundary-Scan*”) la un sistem de dezvoltare conținând unul sau mai multe circuite 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*”. De menționat că un cablu JTAG poate fi utilizat de obicei și pentru configurarea circuitelor în alte moduri decât modul JTAG, cum sunt modurile “*Slave Serial*” sau “*Slave Parallel*”.

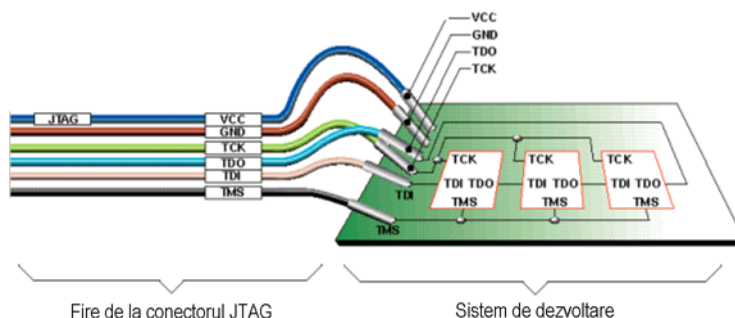


Figura 6.16. Conectarea unui cablu JTAG la un sistem de dezvoltare în modul “*Boundary-Scan*”.

Tabelul 6.1 indică denumirea și semnificația semnalelor JTAG care sunt utilizate pentru configurarea circuitelor programabile.

Tabelul 6.1. Denumirea și semnificația semnalelor JTAG utilizate pentru configurarea circuitelor.

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 programabile
TDO	<i>Test Data Out</i> - Semnal pentru citirea datelor de la circuitele programabile
TDI	<i>Test Data In</i> - Semnal pentru transmiterea instrucțiunilor și datelor la circuitele programabile
TMS	<i>Test Mode Select</i> - Semnal decodificat de controlerul JTAG pentru controlul operațiilor

2.2.1.8. Depanarea sistemului

În această ultimă etapă a procesului de proiectare se verifică funcționarea sistemului digital proiectat în condiții reale. O funcționare necorespunzătoare se poate datora nerespectării specificațiilor de proiectare, a specificațiilor circuitului utilizat pentru implementare, a unor aspecte legate de întârzierea 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 vizualizarea semnalelor dorite.

2.3. Sistemul de proiectare Xilinx WebPACK ISE

Pachetul de programe WebPACK ISE al firmei Xilinx reprezintă un mediu integrat pentru proiectarea sistemelor digitale utilizând circuite FPGA sau CPLD produse de această firmă. Acest pachet conține un subset al sistemului de proiectare Xilinx Foundation ISE. Pentru descrierea sistemelor digitale, proiectantul poate utiliza scheme, limbaje de descriere (VHDL sau Verilog pentru circuitele FPGA, ABEL pentru circuitele CPLD), sau diagrame de stare. Este posibilă utilizarea combinată a acestor metode pentru descrierea diferitelor componente ale aceluiași sistem digital.

Pachetul de programe WebPACK este disponibil gratuit de pe paginile Web ale firmei Xilinx. Pentru transferul pachetului este necesară înregistrarea prealabilă la adresa:

http://www.xilinx.com/xlnx/xil_entry2.jsp?sMode=login&group=webpack

Se alege un identificator și o parolă, după care va fi permis accesul la pagina de pe care se poate transfera pachetul, adresa acestei pagini fiind:

<http://www.xilinx.com/webpack/index.html>

Pentru simularea descrierilor, sistemul WebPACK conține o versiune a simulatorului ModelSim al firmei Model Technology (<http://www.model.com>), denumită MXE (*ModelSim Xilinx Edition*). Acest simulator permite simularea funcțională a descrierilor înainte de sinteză, sau simularea după procesul de implementare pentru verificarea întârzierii semnalelor. Sistemul WebPACK conține de asemenea o interfață grafică pentru specificarea unor vectori de test. Pe baza acestora se generează un banc de test sub forma unui fișier HDL, care va fi compilat împreună cu descrierea sistemului digital pentru simularea funcționării acestuia.

Sistemul WebPACK permite utilizarea următoarelor tipuri de circuite FPGA și CPLD Xilinx:

- Circuite FPGA din familia Spartan: Spartan-2, Spartan-2E;
- Circuite FPGA din familia Virtex: Virtex, Virtex-E, Virtex-2, Virtex-2 Pro;
- Circuite CPLD din familia XC9500: XC9500, XC9500XL, XC9500XV;
- Circuite CPLD din familia CoolRunner: CoolRunner, CoolRunner-2.

2.3.1. Structura sistemului WebPACK

Figura 6.17 prezintă principalele module software ale sistemului de proiectare WebPACK.

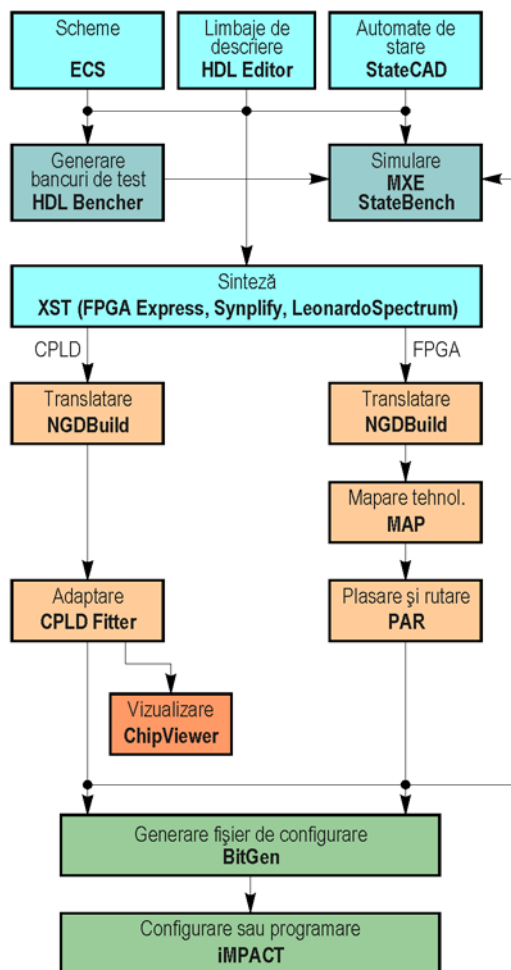


Figura 6.17. Structura sistemului de proiectare Xilinx WebPACK.

2.3.2. Prezentarea modulelor

Pentru proiectarea unui sistem digital cu ajutorul pachetului WebPACK, trebuie să se creeze mai întâi un proiect. Acestuia i se asociază un fișier care va conține informațiile principale despre sistemul proiectat: fișierele care conțin descrierea sistemului digital, tipul circuitului care va fi utilizat pentru implementare etc. Toate fișierele care conțin descrierea sistemului digital și cele care vor fi generate în diferitele etape ale procesului de proiectare se păstrează într-un director separat pe disc.

2.3.2.1. Interfața grafică *Project Navigator*

Programele care compun sistemul de proiectare WebPACK pot fi lansate din fereastra unei interfețe grafice, numită *Project Navigator*, care coordonează procesele și fișierele asociate cu un anumit proiect. Acest “navigator” vizualizează fișierele de intrare ale proiectului, fișierele intermediare sau de ieșire generate pe parcursul etapelor de proiectare, procesele care pot fi executate asupra diferitelor fișiere și mesajele generate de programele executate.

Ecranul interfeței grafice *Project Navigator* este împărțit în patru ferestre principale, după cum se ilustrează în Figura 6.18. În *fereastra fișierelor sursă* se afișează toate fișierele sursă care au fost incluse în proiectul curent. Prin execuția unui clic dublu pe numele unui fișier din această fereas-

tră se va deschide fișierul respectiv de către modulul CAD corespunzător. *Fereastra proceselor* indică toate procesele care sunt disponibile pentru un anumit tip de fișier sursă. Execuția unui clic dublu pe numele unui proces va lansa în execuție procesul respectiv. *Fereastra editorului HDL* afișează conținutul fișierului sursă selectat, conținând descrierea unei componente digitale într-un limbaj de descriere hardware. Conținutul fișierului poate fi editat în această fereastră. În *fereastra consolei* se afișează toate mesajele care indică starea execuției unui proces, inclusiv mesajele de avertisment sau de eroare care sunt generate în urma execuției.

Pe lângă aceste ferestre principale, interfața *Project Navigator* poate deschide și alte ferestre necesare pentru unele aplicații.

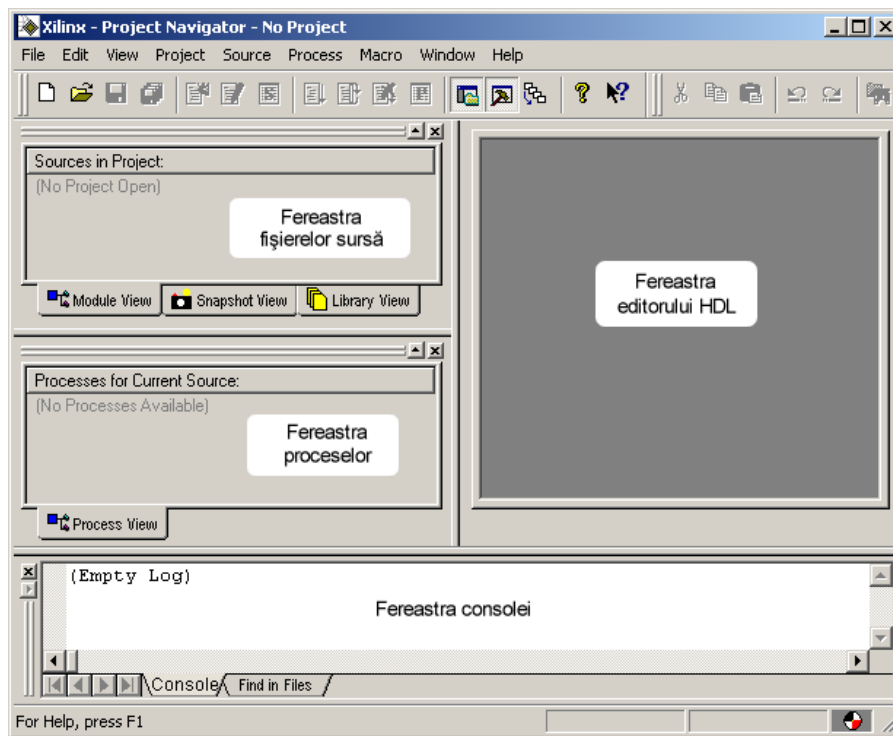


Figura 6.18. Ecranul principal al interfeței grafice *Project Navigator*.

2.3.2.2. Modulele pentru descrierea sistemului

Modulele pentru descrierea sistemului digital cuprind editorul schematic ECS (*Engineering Capture System*), editorul HDL pentru limbaje de descriere hardware și editorul pentru diagrame de stare StateCAD. Editorul schematic utilizează un modul care conține primitivele de bibliotecă pentru circuitele FPGA și CPLD care pot fi utilizate pentru implementarea sistemului digital. Editorul HDL permite editarea fișierelor sursă care conțin descrierea sistemului digital în limbajul VHDL sau Verilog. Editorul StateCAD permite descrierea unor automate cu stări finite utilizând diagrame de stare. Utilizatorul poate desena stările automatului, tranzițiile între stări și poate specifica semnalele de ieșire care trebuie generate în fiecare stare.

Sistemul digital poate fi descris utilizând mai multe metode, de exemplu, prin scheme și limbaje de descriere hardware. Schemele și diagramele de stare vor fi compilate în descrieri HDL care pot fi simulate pentru verificarea funcțională a sistemului proiectat. Aceste descrieri vor fi utilizate în următoarele etape de proiectare.

2.3.2.3. Modulul HDL Bench

Modulul HDL Bencher permite testarea mai simplă a sistemului digital. Cu ajutorul acestui modul, utilizatorul poate introduce semnalele de intrare sub formă grafică, ca diagrame de timp. Pe baza acestor semnale se va genera un banc de test ("*testbench*") sub forma unei descrieri HDL, descri-

ere care poate fi utilizată pentru simularea funcționării sistemului digital. Utilizatorul poate specifica și rezultatele așteptate ale simulării, ceea ce permite simulatorului să afișeze mesaje de avertisment dacă rezultatele obținute prin simulare nu coincid cu cele așteptate.

2.3.2.4. Simulatoarele MXE și StateBench

Simulatorul *ModelSim Xilinx Edition* (MXE) permite atât o simulare funcțională a sistemului digital proiectat, cât și o simulare care ține cont de întârzierile semnalelor. Acest simulator utilizează biblioteci pentru circuitele programabile care pot fi utilizate pentru implementare, aceste biblioteci fiind precompilate. Descrierea sistemului digital care trebuie testat și bancul de test generat pe baza vectorilor de test sunt compilate, rezultatul compilării fiind utilizat pentru simularea sistemului.

Simularea funcțională a sistemului digital este executată înainte de etapa de sinteză. După implementarea într-un circuit programabil se poate realiza o simulare temporală, adăugând informațiile despre întârzierile introduse de către blocurile logice și cele datorate interconexiunilor.

Pentru simularea funcționării automatelor de stare descrise cu ajutorul editorului StateCAD se poate utiliza simulatorul StateBench. Acest simulator permite și crearea unor bancuri de test, fie în mod interactiv, fie în mod automat. După crearea unui banc de test, se poate simula funcționarea automatului utilizând bancul de test creat.

2.3.2.5. Modulele pentru sinteză

Aceste module realizează sinteza descrierii sistemului digital, generând o reprezentare internă a sistemului sub forma unei liste de conexiuni. Rezultatele sintezei sunt controlate prin specificarea diferitelor proprietăți (opțiuni). Una din proprietățile specificate în mod obișnuit este cea care indică modul în care se dorește optimizarea rezultatelor sintezei, din punct de vedere al resurselor utilizate sau al vitezei de funcționare. O altă proprietate este cea care indică efortul de optimizare, afectând timpul care va fi necesar pentru execuția procesului de sinteză.

Modulul de sinteză care este integrat în cadrul sistemului WebPACK este XST (*Xilinx Synthesis Technology*). Pe lângă acest modul, se pot utiliza următoarele module de sinteză pentru care există interfețe integrate în cadrul sistemului: *FPGA Express* (Synopsys, Inc.), *Synplify/Synplify Pro* (Synplicity, Inc.) sau *LeonardoSpectrum* (Exemplar Logic, Inc.). Aceste module nu fac parte însă din sistemul WebPACK și trebuie instalate separat.

2.3.2.6. Modulele pentru implementare

O parte din modulele utilizate pentru implementarea sistemului digital sunt diferite în funcție de tipul circuitului programabil utilizat: CPLD sau FPGA. Pentru ambele tipuri de circuite, prima etapă de implementare este cea de *translatare*. În această etapă se rulează programul NGDBuild, care prelucrează lista de conexiuni obținută în etapa de sinteză (sau listele de conexiuni în cazul în care descrierea sistemului digital este conținută în mai multe fișiere sursă). Aceste liste de conexiuni, care pot fi în format XNF sau EDIF, sunt convertite în formatul Xilinx *Native Generic Database* (NGD). Pe lângă această conversie, programul NGDBuild realizează verificarea constrângerilor de temporizare specificate de proiectant și verificarea regulilor de proiectare (DRC – *Design Rule Check*). Programul generează un raport care conține eventualele mesaje de eroare sau de avertismente despre rezultatele acestor verificări.

În cazul proiectării cu circuite CPLD, următoarea etapă de implementare este cea de *adaptare* (“*Fitting*”), executată de modulul CPLD Fitter. Acest modul minimizează logica utilizată pentru descrierea sistemului și realizează adaptarea acesteia la particularitățile circuitului CPLD din familia XC9500 sau CoolRunner, astfel încât să se utilizeze numărul minim de macrocelule și termeni produs. Modulul CPLD Fitter poate genera și un raport conținând caracteristici de timp ale semnalelor, raport care indică și frecvența maximă a semnalului de ceas la care poate funcționa sistemul proiectat.

În cazul proiectării cu circuite FPGA, următoarea etapă de implementare după cea de *translatare* este *maparea tehnologică*, executată de programul MAP. Acest program execută mai întâi verificarea regulilor de proiectare pentru descrierea din fișierul în format NGD, iar apoi realizează

adaptarea logicii la resursele disponibile în circuitul FPGA din familia Spartan sau Virtex utilizat pentru implementare. Rezultatul mapării tehnologice este un fișier în format NCD (*Native Circuit Description*), care specifică elementele fizice care vor fi utilizate din circuitul FPGA. Programul MAP generează un raport cu rezultatele operațiilor efectuate. Raportul conține mesajele de eroare și de avertisment care au fost afișate, logica eliminată în urma optimizărilor efectuate, numărul și procentul de utilizare a blocurilor logice, a blocurilor de I/E și a bistabilelor.

După maparea tehnologică, se rulează programul PAR, care execută operațiile de *plasare* și *rutare* a circuitului FPGA. Acest program prelucrează fișierul în format NCD obținut în urma mapării tehnologice și generează un fișier în același format care conține și informațiile de rutare. Programul PAR generează mai multe rapoarte care conțin: numărul semnalelor care nu au putut fi rutate complet, caracteristici de timp ale semnalelor, pinii circuitului sortați mai întâi după numele semnalelor, iar apoi după număr, frecvența maximă de funcționare a circuitului.

După implementare, se poate genera un model al sistemului proiectat conținând și întârzierile semnalelor, aceste informații fiind adăugate prin operația de *adnotare inversă*. Modelul generat poate fi utilizat pentru simularea temporală a sistemului. Pentru generarea acestui model, se rulează (în mod automat sau la comanda utilizatorului) programul NGDAnno și unul din programele NGD2VHDL, NGD2VER sau NGD2EDIF, în funcție de fluxul de proiectare utilizat (XST VHDL, XST Verilog, respectiv EDIF).

2.3.2.7. Modulul pentru generarea fișierului de configurare

După executarea etapelor de implementare, se poate genera un fișier care va fi utilizat pentru configurarea circuitului programabil astfel încât acesta să execute funcțiile dorite ale sistemului digital proiectat. Fișierul de configurare este sub forma unui șir de biți ("*bitstream*"), conținând toate informațiile din fișierul NCD care definesc logica internă și interconexiunile circuitului utilizat pentru implementare, plus alte informații din unele fișiere asociate cu acest circuit. Asemenea informații sunt, de exemplu, cele care indică asignarea semnalelor la pinii circuitului. Fișierul de configurare este generat de programul BitGen. Acest fișier poate fi utilizat apoi pentru configurarea circuitului sau pentru crearea unui fișier necesar programării unei memorii PROM cu informațiile de configurare ale circuitului.

Pentru circuitele CPLD, fișierul de configurare este creat în formatul JEDEC (*Joint Electron Device Engineering Council*) și are extensia **.jed**. Ulterior, se poate crea un fișier în formatul SVF (*Serial Vector Format*) cu ajutorul modulului iMPACT. Pentru circuitele FPGA, fișierul de configurare creat este un fișier binar și are extensia **.bit**.

2.3.2.8. Modulul pentru configurare sau programare iMPACT

Modulul iMPACT permite configurarea sau programarea circuitului CPLD sau FPGA utilizat pentru implementare. Pentru aceasta trebuie conectat un cablu paralel sau un cablu JTAG la portul paralel al calculatorului (în unele cazuri, se poate utiliza un cablu serial sau un cablu USB). Fișierul care conține șirul de biți pentru configurare va fi transferat la circuitul programabil.

Modulul iMPACT conține și un program (*PROM File Formatter*) pentru generarea fișierului necesar programării unei memorii PROM. O asemenea memorie nevolatilă poate fi utilizată pentru păstrarea informațiilor de configurare necesare circuitelor CPLD sau FPGA care se bazează pe tehnologia SRAM și necesită configurarea după fiecare întrerupere a tensiunii de alimentare. Fișierul generat de modulul iMPACT poate avea formatul MCS-86 (Intel), EXORMACS (Motorola) sau TEKHEX (Tektronix). Se poate genera și un fișier HEX, care conține reprezentarea hexazecimală a șirului de configurare. Fișierul generat poate fi utilizat pentru programarea unei memorii PROM cu ajutorul unui echipament de programare.

2.3.2.9. Modulul de vizualizare

Pentru circuitele CPLD, modulul *ChipViewer* poate fi utilizat pentru vizualizarea în mod grafic a structurii macrocelulelor după configurare, a semnalelor de intrare și de ieșire, a ecuațiilor sem-

nalelor a modului de asignare a pinilor de I/E și a întârzierii semnalelor între doi pini. De asemenea, modulul *ChipViewer* permite asignarea semnalelor la anumiți pini de I/E înaintea etapelor de implementare.

2.4. Exemplu de proiectare

În continuare se prezintă un exemplu simplu de proiectare utilizând pachetul de programe Xilinx WebPACK, versiunea 4.2.

2.4.1. Prezentarea circuitului proiectat

Pentru exemplul de proiectare se va utiliza editorul schematic ECS. Circuitul proiectat constă din două numărătoare de câte 16 biți, care sunt conectate pentru a forma un numărător de 32 de biți. Acest numărător va fi implementat pe o placă de dezvoltare XSA-50 a firmei XESS, care conține un circuit FPGA Xilinx Spartan2 XC2S50, cu o capacitate de 50.000 porți echivalente. Numărătorul primește semnalul de ceas de 100 MHz de la generatorul de ceas al plăcii XSA, realizat cu un oscilator programabil. Două din liniile de ieșire ale numărătorului vor fi conectate la segmentul de sus și cel de jos al afișajului cu 7 segmente al plăcii. Segmentul de sus va fi comandat de bitul 25 al numărătorului, bit a cărui valoare se modifică din 0 în 1 sau din 1 în 0 cu o frecvență de $100.000.000 / 2^{26} = 1,49$ Hz, astfel că acest segment se va aprinde sau se va stinge de 1,49 ori pe secundă (aproximativ de 3 ori în 2 secunde). Segmentul de jos va fi comandat de bitul 24 al numărătorului, iar acest segment se va aprinde sau se va stinge de aproximativ 3 ori pe secundă.

Schema circuitului este prezentată în Figura 6.19.

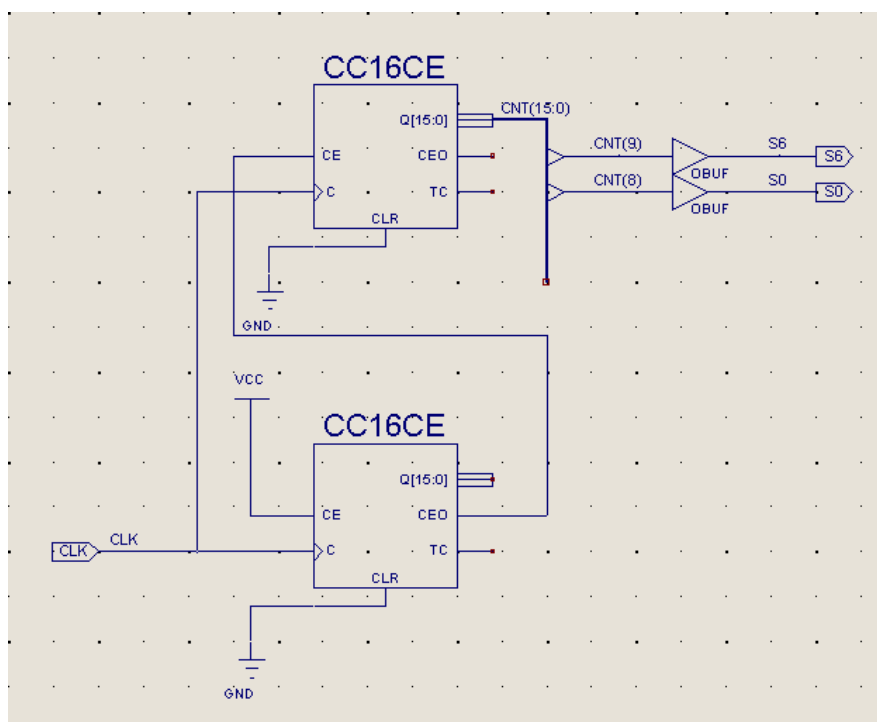


Figura 6.19. Schema circuitului utilizat ca exemplu de proiectare.

2.4.2. Lansarea programului WebPACK

Pentru lansarea în execuție a programului WebPACK se execută un dublu clic pe icoana interfeței grafice *Project Navigator* (Figura 6.20). Programul poate fi lansat în execuție și selectând în meniul *Start* → *Programs* → *Xilinx WebPACK 4.2* → *WebPACK Project Navigator*. În funcție de

configurarea interfeței *Project Navigator*, se va încărca ultimul proiect care a fost utilizat, sau nu se va încărca nici un proiect.



Figura 6.20. Icoana interfeței grafice *WebPACK Project Navigator*.

2.4.3. Crearea proiectului

Pentru crearea unui nou proiect, se execută următoarele etape:

1. În ecranul *Project Navigator* se selectează *File* → *New Project*. Se va deschide fereastra de dialog *New Project*.
2. În câmpul *Project Location* din fereastra de dialog *New Project* se introduce directorul în care se va crea proiectul, de exemplu, **C:\Student\Tcalc2\AC**. În acest director, programul va crea un nou director pentru fișierele proiectului. Se poate utiliza butonul “...” pentru a selecta un director existent pe disc.
3. În câmpul *Project Name* se introduce numele proiectului, de exemplu, **lab6_1**.
4. În câmpul *Device Family* se selectează familia de circuite FPGA: **Spartan2**.
5. În câmpul *Device* se selectează circuitul FPGA: **xc2s50-5tq144**. În această denumire, **xc2s50** indică tipul și capacitatea circuitului, **-5** este un indicator al vitezei circuitului, iar **tq144** reprezintă tipul capsulei circuitului și numărul de pini.
6. În câmpul *Design Flow* se selectează fluxul de proiectare: **XST VHDL**.
7. Se execută un clic pe butonul *OK* pentru a crea proiectul.

2.4.4. Lansarea editorului schematic

Pentru lansarea în execuție a editorului schematic ECS, se execută următoarele operații:

1. În ecranul *Project Navigator* se selectează *Project* → *New Source*. Se va deschide fereastra de dialog *New*.
2. Se selectează opțiunea *Schematic* din meniu.
3. În câmpul *File Name* se introduce numele fișierului care va conține schema numărătorului proiectat, de exemplu, **num32**.
4. Se execută un clic pe butonul *Next*, iar apoi pe butonul *Finish*. Editorul schematic va fi lansat într-o nouă fereastră.

2.4.5. Adăugarea componentelor

În etapa următoare se vor adăuga componentele necesare pentru realizarea numărătorului de 32 de biți. Componentele disponibile pentru diferitele familii de circuite programabile se păstrează în câte o bibliotecă de componente, această bibliotecă fiind specifică pentru o anumită familie de circuite. Categoriile de componente sunt afișate în fereastra *Categories* a editorului schematic, iar componentele dintr-o categorie selectată sunt afișate în fereastra *Symbols*. Deoarece nu este disponibil un numărător de 32 de biți, acesta va fi realizat prin interconectarea a două numărătoare de 16 biți.

Pentru adăugarea componentelor necesare, se procedează astfel:

1. În bara de meniuri se selectează comanda *Add* → *Symbol*, sau se execută un clic pe butonul *Add Symbol* .
2. În zona din dreapta ecranului, în câmpul *Categories* se selectează **Counter**, iar în câmpul *Symbols* se selectează componenta **cc16ce**.
3. Se deplasează cursorul în fereastra principală a editorului. Se observă că s-a atașat cursorului simbolul unui numărător. Se deplasează cursorul și se plasează numărătorul prin execuția unui clic în poziția dorită (conform figurii 6.19). Se observă că simbolul este etichetat cu numele componentei, iar terminalele sunt etichetate cu numele lor.
4. Se plasează al doilea numărător prin deplasarea cursorului și execuția unui clic în poziția dorită.
5. Se apasă tasta **Esc** pentru terminarea plasării componentelor.

Componentele adăugate pot fi deplasate ulterior în pozițiile convenabile. Ștergerea unei componente se poate realiza prin selectarea acesteia și apăsarea tastei **Delete**.

Se pot utiliza butoanele *Zoom In*  și *Zoom Out*  pentru vizualizarea componentelor la dimensiunile convenabile.

2.4.6. Interconectarea componentelor

În continuare se vor interconecta componentele din schemă prin fire de conexiuni. Pentru aceasta se poate selecta în bara de meniuri comanda *Add* → *Wire*, sau se poate utiliza butonul *Add Wire* . Pentru interconectarea a doi pini se execută un clic pe unul din pini, se deplasează cursorul, iar apoi se execută un clic dublu pe al doilea pin. Pinul selectat trebuie să apară marcat prin patru puncte de culoare roșie plasate în colțurile unui mic pătrat. Similar se poate interconecta un pin cu un fir de conexiune existent, executând un clic pe conexiunea respectivă. Se pot interconecta mai mulți pini, dar atunci se execută un clic dublu doar pe ultimul pin. Terminarea interconectării se poate realiza și prin apăsarea tastei **Esc**.

Este posibilă și interconectarea logică a două semnale dacă se denumesc în mod identic două sau mai multe segmente de interconectare. În acest caz, firele de conexiune nu trebuie conectate fizic în cadrul schemei.

Se execută următoarele etape pentru interconectarea componentelor din schemă:

1. Pentru conectarea semnalului de ceas la numărătorul de jos, se execută un clic pe butonul *Add Wire* ; forma cursorului devine cea a unui creion. Se execută un clic pe terminalul *C* al numărătorului de jos, iar apoi se trasează o linie orizontală spre stânga și se execută un clic dublu în poziția dorită. La această conexiune se va atașa ulterior un terminal de intrare, care se va plasa pe pinul circuitului FPGA care este conectat la generatorul de ceas de pe placa de dezvoltare.
2. Se conectează terminalul de ceas *C* al numărătorului de sus la conexiunea trasată anterior, executând un clic pe acest terminal, iar apoi un clic pe firul de conexiune. Se apasă apoi tasta **Esc**.
3. Se conectează în mod similar terminalul de ieșire *CEO* (*Count Enable Out*) al numărătorului de jos la terminalul de validare al ceasului *CE* (*Clock Enable*) al numărătorului de sus. Pentru a obține forma dorită a conexiunii, se poate executa un clic în punctele intermediare în care se dorește schimbarea direcției liniei de conexiune.

Prin această conectare a terminalelor *CEO* și *CE*, numărătorul de sus se va incrementa numai atunci când numărătorul de jos trece de la valoarea FFFFh la 0000h, deci o dată la fiecare 65.536 impulsuri de ceas. Aceasta deoarece ieșirea *CEO* a numărătorului de jos va trece în 1 logic numai atunci când valoarea numărătorului este FFFFh. Prin conectarea ieșirii *CEO* la intrarea de validare a ceasului

(CE) numărătorului de sus, acest numărător va reacționa la un impuls de ceas numai atunci când numărătorul de jos ajunge la valoarea FFFFh.

Numărătorul de jos, care va conține biții mai puțin semnificativi ai contorului, trebuie să se incrementeze la fiecare impuls de ceas. De aceea, intrarea CE a acestui numărător trebuie conectată în permanență la 1 logic. Valoarea 1 logic se poate obține prin plasarea și conectarea simbolului VCC. Fiecare numărător are o intrare de ștergere CLR (Clear) care re setează numărătorul la 0 atunci când această intrare trece în 1 logic. Pentru a preveni resetarea numărătoarelor în timpul funcționării, se va aplica în permanență valoarea logică 0 la ambele intrări CLR. Valoarea 0 logic se poate obține prin plasarea și conectarea simbolului GND.

Pentru conectarea intrării CE la 1 logic și a intrărilor CLR la 0 logic se procedează astfel:

1. Se execută un clic pe butonul *Add Symbol* . În fereastra *Categories* se selectează **General**, iar în fereastra *Symbols* se selectează simbolul **vcc**.
2. Se deplasează cursorul în fereastra principală și se verifică forma simbolului atașat cursorului. Dacă această formă nu este cea a literei T, se execută clic de două ori pe butonul *Rotate*  pentru rotirea simbolului. Se plasează apoi simbolul **vcc** deasupra terminalului CE a numărătorului de jos și la stânga acestuia, asigurând un spațiu suficient pentru a putea conecta simbolul la terminal printr-o linie de conexiune. Se apasă apoi tasta **Esc**.
3. Se trasează o linie de conexiune între simbolul **vcc** și terminalul CE al numărătorului de jos, în modul descris anterior.
4. În fereastra *Symbols* se selectează simbolul **gnd**. Se plasează câte un simbol **gnd** sub terminalele CLR ale numărătoarelor și la stânga acestora, asigurând un spațiu suficient pentru a putea conecta simbolurile la terminale prin linii de conexiune. Se apasă apoi tasta **Esc**.
5. Se trasează linii de conexiune între simbolurile **gnd** și terminalele CLR ale numărătoarelor.

Se salvează schema, selectând *File* → *Save* sau executând un clic pe butonul *Save* .

2.4.7. Adăugarea și conectarea unei magistrale

În cazul editorului ECS, o magistrală este ca și o linie de conexiune, dar căreia i s-a atribuit un nume indicând numărul de linii ale magistralei, de exemplu, **bus(7:0)**. Pentru adăugarea unei magistrale, se procedează ca și pentru adăugarea unei linii de conexiune, iar apoi se atribuie acesteia numele corespunzător. După adăugarea magistralei, există posibilitatea utilizării individuale a semnalelor magistralei.

Doi biți ai numărătorului de sus trebuie conectați la două segmente ale afișajului de pe placa XSA-50. Pentru a avea acces la acești biți, mai întâi se prelungește magistrala de ieșire Q a numărătorului de sus și se atribuie un nume acestei magistrale, astfel:

1. Se execută un clic pe butonul *Add Wire* .
2. Se trasează o linie de conexiune de la ieșirea Q a numărătorului de sus, mai întâi la dreapta, iar apoi în jos. Se apasă tasta **Esc**.
3. Se selectează *Add* → *Net Name* sau se execută un clic pe butonul *Add Net Name* .
4. Se introduce de la tastatură numele **CNT(15:0)** și se apasă tasta **Enter**. Prin aceasta se atașează cursorului numele magistralei.
5. Se execută un clic pe capătul de sus al magistralei pentru a se atașa numele acesteia. Se apasă apoi tasta **Esc**.

Pentru comanda segmentului de sus și a celui de jos al afișajului cu 7 segmente, este necesar accesul la doi biți ai numărătorului. Se vor utiliza biții 25 și 24 ai numărătorului de 32 de biți, aceștia

fiind reprezentați de biții 9, respectiv 8, ai număratorului de sus. Pentru accesul la acești biți, se procedează astfel:

1. Se selectează *Add* → *Bus Tap* sau se execută un clic pe butonul *Add Bus Tap* . Se observă modificarea formei cursorului.
2. Dacă este necesar, se modifică orientarea în fereastra de opțiuni (cea în care este afișat *Rotate 0*), astfel încât baza triunghiului să fie paralelă cu porțiunea verticală a magistralei. Se execută un clic pe porțiunea verticală a magistralei pentru adăugarea primului conector la magistrală, iar apoi un nou clic pentru adăugarea celui de-al doilea conector. Se apasă apoi tasta **Esc**.

2.4.8. Adăugarea și conectarea bufferelor de ieșire

Semnalele de ieșire se vor conecta la pinii de ieșire prin intermediul unor buffere de ieșire. Pentru adăugarea și conectarea acestora se procedează astfel:

1. Se execută un clic pe butonul *Add Symbol* . În fereastra *Categories* se selectează **IO**, iar în fereastra *Symbols* se selectează simbolul **obuf**.
2. Se plasează două componente **obuf** la dreapta celor doi conectori la magistrală, la o oarecare distanță de aceștia. Se apasă apoi tasta **Esc**.
3. Se trasează câte o linie de conexiune de la fiecare conector al magistralei la bufferul de ieșire corespunzător.
4. Se prelungesc la dreapta liniile de la ieșirile componentelor **obuf**.

2.4.9. Atașarea numelor la conexiuni

În continuare se vor atașa nume unor conexiuni. Pentru aceasta se procedează astfel:

1. Se selectează *Add* → *Net Name* sau se execută un clic pe butonul *Add Net Name* .
2. Se introduce de la tastatură numele **CLK** și se apasă tasta **Enter**.
3. Se execută un clic în partea stângă a liniei conectate la intrarea de ceas *C* a număratorului de jos și se verifică atașarea numelui **CLK** la această linie. Se apasă tasta **Esc**.
4. Se procedează similar pentru atașarea numelui **CNT(9)** la linia dintre conectorul de sus al magistralei și bufferul de ieșire corespunzător.
5. Se atașează numele **CNT(8)** la linia dintre conectorul de jos al magistralei și bufferul de ieșire corespunzător.
6. Se atașează numele **S6** la linia din dreapta bufferului de ieșire conectat la semnalul *CNT(9)*.
7. Se atașează numele **S0** la linia din dreapta bufferului de ieșire conectat la semnalul *CNT(8)*.

2.4.10. Adăugarea porturilor de I/E

Porturile de I/E permit utilizarea unei componente proiectate sau a unui subsistem într-o schemă mai complexă. Pentru numărator proiectat, singurul port de intrare este cel de ceas. Porturile de ieșire sunt liniile **CNT(9)** și **CNT(8)** ale număratorului. Pentru adăugarea acestor porturi, se procedează astfel:

1. Se selectează *Add* → *I/O Marker* sau se execută un clic pe butonul *Add I/O Marker* .
2. Se selectează direcția *Input* în partea de sus a ecranului.

3. Se apasă butonul din stânga al mouse-ului și, cu butonul apăsat, se trasează un dreptunghi în jurul semnalului *CLK*. Se observă adăugarea simbolului pentru portul de intrare.
4. Se selectează direcția *Output*.
5. Se apasă butonul din stânga al mouse-ului și, cu butonul apăsat, se trasează un dreptunghi în jurul semnalelor *S6* și *S0*. Se observă adăugarea a două simboluri pentru porturile de ieșire. Se apasă tasta **Esc**.

Se salvează schema utilizând butonul *Save* .

2.4.11. Verificarea schemei

Pentru verificarea corectitudinii conexiunilor, se selectează comanda *Tools* → *Check Schematic* sau se execută un clic pe butonul *Check Schematic* . Eventualele erori se afișează într-o nouă fereastră, *Schematic Check Errors*. Se verifică dacă sunt erori, iar apoi se închide fereastra cu butonul *Close*. În cazul unor erori, se verifică modul în care sunt realizate interconexiunile pe schemă și se repetă testul.

2.4.12. Asignarea pinilor la semnalele de intrare și ieșire

În acest moment, schema este terminată, dar semnalele de intrare și ieșire trebuie conectate la pinii corespunzători ai circuitului FPGA, în funcție de modul în care sunt conectați acești pini pe placa XSA-50. Conexiunile necesare sunt ilustrate în Figura 6.21.

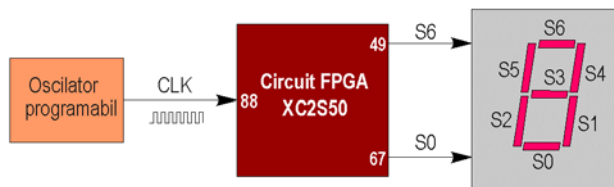


Figura 6.21. Conectarea oscilatorului programabil și a afișajului cu 7 segmente la pinii circuitului FPGA de pe placa XSA-50.

Pinii la care se vor conecta semnalele de intrare și de ieșire se pot specifica prin atașarea unui parametru, denumit *LOC*, semnalelor respective. Pentru aceasta se procedează astfel:

1. Se execută un clic cu butonul din dreapta pe portul de intrare al semnalului *CLK* și se selectează opțiunea *Object Properties*. Se va deschide o fereastră cu atributele acestui semnal.
2. Se execută un clic pe butonul *New* pentru adăugarea unui nou atribut.
3. În fereastra *Attribute Name* se introduce **LOC**, iar în fereastra *Attribute Value* se introduce **P88**. Aceasta înseamnă că semnalul *CLK* va fi conectat la pinul 88 al circuitului FPGA, pin care este conectat la oscilatorul programabil al plăcii XSA-50.
4. Se execută un clic pe butonul *OK*, iar apoi un nou clic pe butonul *OK* pentru închiderea ferestrei *Object Properties*.
5. Se procedează similar pentru conectarea semnalului *S6* la pinul 49 al circuitului FPGA (segmentul *S6* al afișajului cu 7 segmente).
6. Se procedează similar pentru conectarea semnalului *S0* la pinul 67 al circuitului FPGA (segmentul *S0* al afișajului).

Se salvează schema utilizând butonul *Save* .

2.4.13. Sinteza și implementarea proiectului

După terminarea editării schemei, se revine în fereastra *Project Manager*. În fereastra fișierelor sursă se selectează fișierul care conține schema (fișierul cu extensia *.sch*), dacă fișierul nu este selectat. În fereastra proceselor se vor afișa procesele disponibile pentru acest fișier.

1. Pentru lansarea procesului de sinteză, se execută un clic dublu pe numele procesului *Synthesize* din fereastra proceselor. În timpul sintezei, se afișează diferite mesaje în fereastra consolei. După terminarea sintezei, se verifică dacă s-au afișat mesaje de eroare (marcate cu culoarea roșie). În cazul unor erori, se revine la editorul schematic și se verifică schema. Mesajele de avertisment (marcate cu culoarea galbenă) pot fi ignorate.
2. Se execută un clic pe semnul + din stânga numelui *Synthesis*, iar apoi se execută un clic dublu pe linia *View Synthesis Report*. Prin aceasta se va afișa conținutul fișierului de raport generat în urma sintezei. Se parcurge fișierul, iar în secțiunea "*Timing Summary*" se verifică frecvența maximă de funcționare a circuitului. Se închide apoi fișierul de raport.
3. Pentru lansarea procesului de implementare, se execută un clic dublu pe numele procesului *Implement Design* din fereastra proceselor.
4. După terminarea implementării, se execută un clic pe semnul + din stânga numelui *Implement Design*, iar apoi un clic pe semnul + din stânga numelui *Map*. Se execută un clic dublu pe linia *Map Report*. În fișierul afișat, se determină din secțiunea "*Design Summary*" care este procentul resurselor utilizate din circuit: numărul de blocuri logice (*Number of Slices*), numărul de bistabile (*Number of Slice Flip-Flops*) și numărul de porți echivalente (*Total equivalent gate count for design*). Se închide apoi fișierul de raport.
5. Se execută un clic pe semnul + din stânga numelui *Place & Route* din cadrul procesului *Implement Design*. Se vizualizează fișierul de raport care conține asignarea pinilor de I/E (*Pad Report*), generat în urma proceselor de plasare și rutare (*Place & Route*). Se verifică dacă semnalele sunt asignate în mod corect la pinii de I/E. Se închide apoi fișierul de raport.
6. Pentru generarea fișierului de configurare, se execută un clic dublu pe numele procesului *Generate Programming File* din fereastra proceselor. Se verifică în directorul proiectului existența unui fișier cu extensia *.bit*, care este fișierul de configurare.

2.4.14. Configurarea circuitului FPGA

Pentru configurarea circuitului FPGA se procedează astfel:

1. Se lansează în execuție programul GXSLOAD, cu icoana din Figura 6.22, care permite transferul fișierului de configurare la circuitul FPGA.
2. Se verifică tipul plăcii selectate în fereastra *Board Type*; acest tip trebuie să fie XSA-50.
3. Se alimentează placa XSA-50 cu o tensiune de 7,5 V.
4. Se utilizează programul *Windows Explorer* pentru vizualizarea fișierelor din directorul proiectului. Se reduce dimensiunea ferestrei acestui program pentru a fi vizibilă și fereastra programului GXSLOAD.
5. Se selectează fișierul cu extensia *.bit* și se plasează acest fișier în fereastra FPGA/CPLD a programului GXSLOAD.
6. Se execută un clic pe butonul *Load*.

Fișierul de configurare se va transfera la circuitul FPGA, după care circuitul va începe să funcționeze. Dacă oscilatorul plăcii este programat la 100 MHz, segmentul de sus al afișajului cu 7 segmente va clipi de aproximativ 3 ori în 2 secunde, iar segmentul de jos de 3 ori pe secundă.



Figura 6.22. Icoana programului GXSLOAD.

3. Desfășurarea lucrării

3.1. Executați etapele descrise în secțiunea 2.4 pentru proiectarea și verificarea funcționării numărătorului.

3.2. Modificați schema pentru a conecta alți doi biți ai numărătorului de sus la cele două segmente ale afișajului. Repetați etapele necesare pentru implementare, configurați din nou circuitul și verificați funcționarea.

3.3. Conectați alți biți ai numărătorului de sus la alte segmente ale afișajului. Conectarea segmentelor afișajului la pinii circuitului FPGA este indicată în Tabelul 6.2.

Tabelul 6.2. Conexiunile segmentelor afișajului cu 7 segmente la pinii circuitului FPGA.

Segment afișaj	Pin FPGA
S0	67
S1	39
S2	62
S3	60
S4	46
S5	57
S6	49