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 şi un exemplu simplu de proiectare utilizând acest sistem.

2. Considerații teoretice

2.1. Circuite logice programabile

Circuitele logice programabile, cunoscute şi 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 măşti, 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 realizare 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 × indică o conexiune programabilă a unei linii de intrare la o poartă logică. Absența semnului × indică faptul că respectiva conexiune a fost programată în starea deconectată.

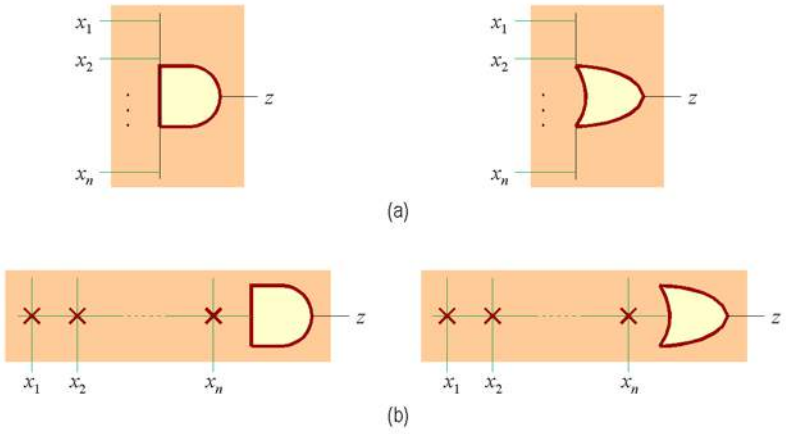

Figura 6.1. Porți ŞI, respectiv SAU: (*a*) reprezentare obişnuită; (*b*) reprezentare simplificată pentru circuitele PLD.

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 şi concept cu o memorie ROM, cu excepția faptului că nu realizează decodificarea completă a variabilelor şi nu generează toți mintermii. Decodificatorul este înlocuit cu o rețea de porți ŞI 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.

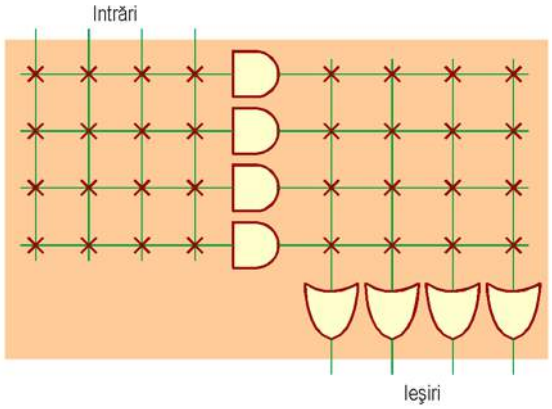

Figura 6.2. Structura generală a unui circuit PLA.

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, şi acestuia îi corespunde o linie de porți ŞI din primul etaj al circuitului PLA. Fiecare ieşire 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 ieşirea este adevărată. Dimensiunea totală a circuitului PLA este egală cu suma dintre dimensiunea rețelei de porți ŞI şi dimensiunea rețelei de porți SAU. Din Figura 6.2 se observă că dimensiunea rețelei de porți ŞI este egală cu numărul de intrări multiplicat cu numărul diferiților termeni produs, iar dimensiunea rețelei de porți SAU este egală cu numărul de ieşiri 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 măşti (în timpul fabricației) sau pot fi programate de către utilizator. Circuitele PLA programate de către utilizator se numesc FPLA (*Field Programmable Logic Array*).

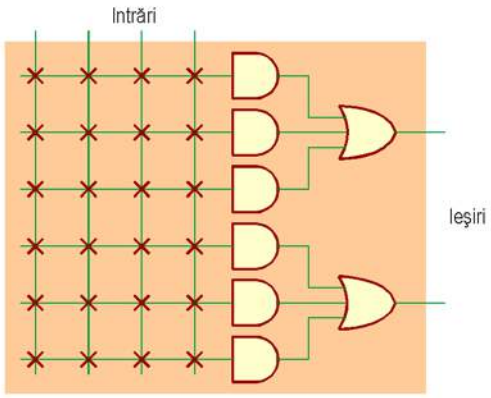

Figura 6.3. Structura generală a unui circuit PAL.

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

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

Există şi circuite PLA sau PAL care conțin bistabile atașate prin conexiuni programabile la ieşirile rețelei de porți SAU, ceea ce permite implementarea unor circuite secvențiale de dimensiuni medii. Aceste circuite sunt cunoscute şi 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 uni singur bloc. Funcțiile mai complexe pot necesita mai multe blocuri, care vor fi interconectate prin matricea de rutare.

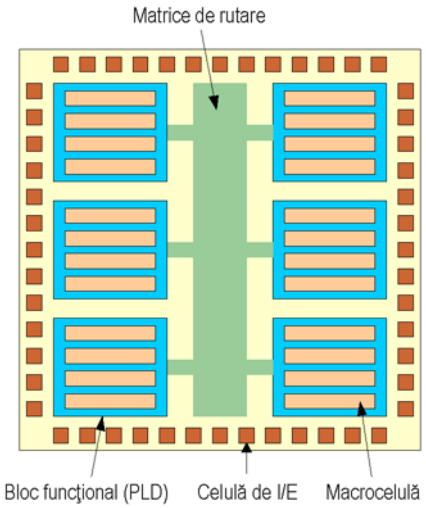

Figura 6.4. Structura generală a unui circuit CPLD.

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 şi 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 ieşiri.

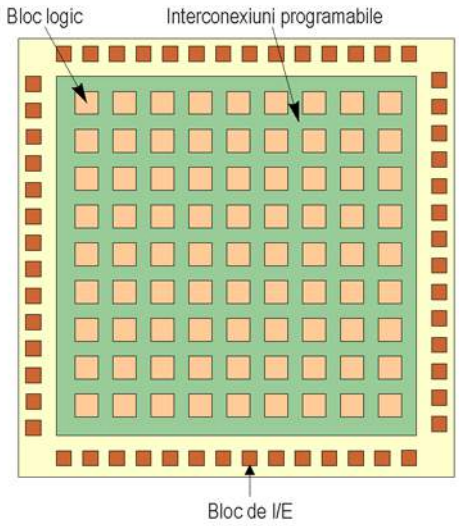

Figura 6.5. Structura unui circuit FPGA tipic.

Din punctul de vedere al tipului conexiunilor programabile, există două categorii principale de circuite FPGA: circuite cu memorii SRAM şi 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 şi 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 citeşte 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 tabelelor şi 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 şi 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 şi a programării, circuitele FPGA bazate pe antifuzibile constau de obicei din rânduri de elemente logice configurabile cu canale de interconectare între ele, ca şi 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 şi pini din canal şi 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 şi 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 şi 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.

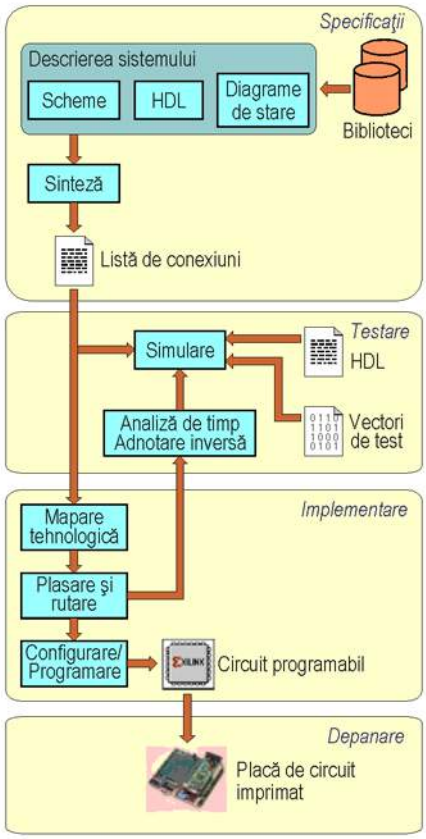

Figura 6.6. Fluxul de proiectare a sistemelor digitale utilizând circuite programabile.

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*) şi 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 şi a modului în care acestea trebuie interconectate. Această metodă este cea care va fi utilizată în primul rând în

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

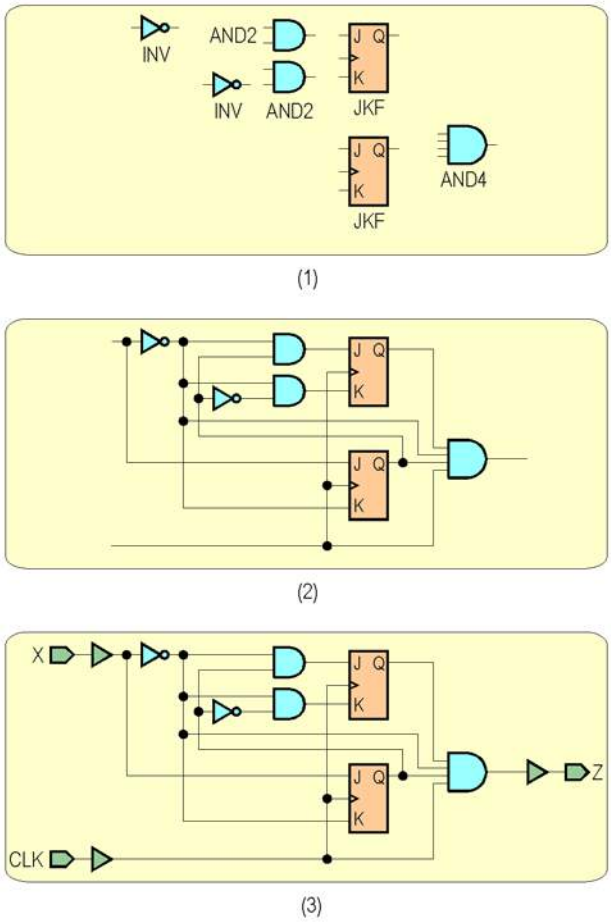

Figura 6.7. Etapele descrierii unui sistem digital utilizând scheme: (1) Selectarea şi amplasarea componentelor; (2) Conectarea componentelor; (3) Adăugarea porturilor de I/E.

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 şi diferitelor familii de circuite. Circuitele dintr-o anumită familie diferă prin capacitatea lor, viteza şi capsula utilizată. În această etapă, nu este însă necesară specificarea exactă a circuitului care va utilizat dintr-o anumită familie.
2. Componentele selectate şi 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 şi se etichetează porturile de I/E. Aceste porturi definesc intrările şi ieşirile sistemului, permițând aplicarea semnalelor la pinii de intrare ai sistemului digital şi preluarea semnalelor de ieşire generate de sistem la pinii de ieşire. Semnalele de intrare sunt aplicate la intrările sistemului digital prin intermediul unor buffere de intrare, iar semnalele de ieşire sunt preluate de la sistemul digital prin intermediul unor buffere de ieşire. 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 aceeaşi descriere se poate utiliza pentru implementarea sistemului într-un anumit circuit FPGA, dar şi 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 acelaşi 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 şi 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 şi a semnalelor de ieşire 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 şi 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 şi descrierea în limbajul VHDL din Figura 6.8.

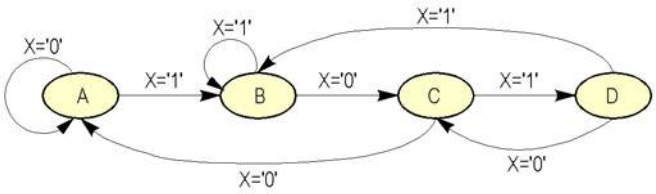

Figura 6.9. Diagrama de stare echivalentă cu circuitul reprezentat prin schema din Figura 6.7 şi descrierea în limbajul VHDL din Figura 6.8.

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 translatare 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 şi pinii de intrare/ieşire. 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 şi 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.

Relaţia dintre schema logică a unui circuit simplu şi 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 şi G2), două porţi ŞI cu două intrări (G3 şi G4), o poartă ŞI cu patru intrări (G7) şi două bistabile JK (G5 şi G6). Inversoarele au un pin de intrare IN, un pin de ieşire OUT, un pin de alimentare Vcc şi un pin de masă GND. Similar, porţile ŞI cu două intrări au doi pini de intrare IN1 şi IN2, un pin de ieşire OUT, un pin de alimentare şi un pin de masă. Bistabilele au doi pini pentru intrările de date J şi K, un pin pentru intrarea de ceas C şi un pin pentru ieşirea Q, pe lângă pinii de alimentare şi masă. Pentru simplitate, pinii şi semnalele de alimentare şi masă au fost omişi în această figură. O conexiune este indicată prin listarea tuturor pinilor care sunt conectaţi împreună. Semnalele de intrare *X* şi *CLK* sunt conectate la pinii de intrare cu aceleaşi nume ai circuitului, iar semnalul de ieşire *Z* este conectat la pinul de ieşire al circuitului.

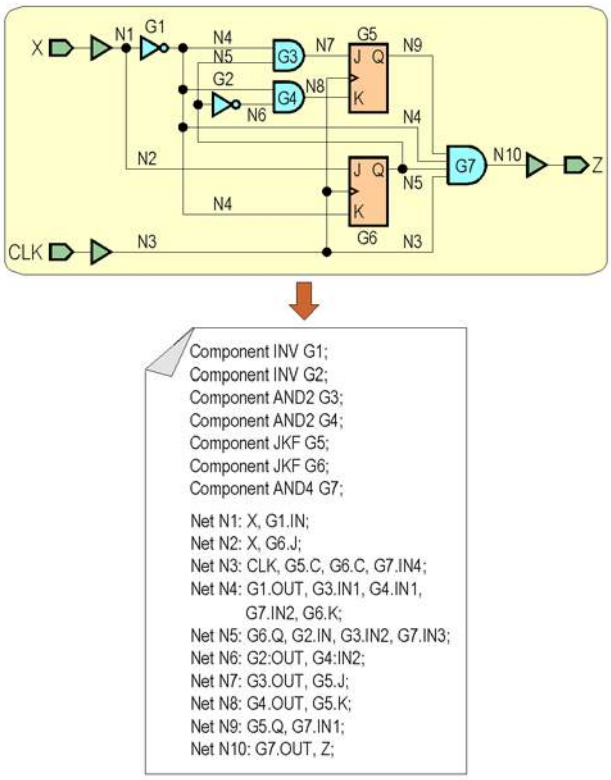

Figura 6.10. Relația dintre schema logică şi lista de conexiuni pentru circuitul din Figura 6.7.

Proiectantul poate specifica diferite criterii de optimizare de care să se țină cont în procesul de sinteză. Exemple de asemenea opțiuni sunt: minimizarea numărul 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 ieşire 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 ieşire care sunt generate de sistem şi le compară cu valorile acestor semnale care au fost specificate de proiectant. În cazul în care apar diferențe, simulatorul afişează mesaje care indică diferențele apărute. Proiectantul va efectua modificările necesare ale descrierii sistemului pentru a corecta erorile apărute, va efectua sinteza descrierii modificate şi 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 şi de ieşire ale circuitului detector de secvență utilizat ca exemplu în secțiunile precedente la simularea funcțională a circuitului.

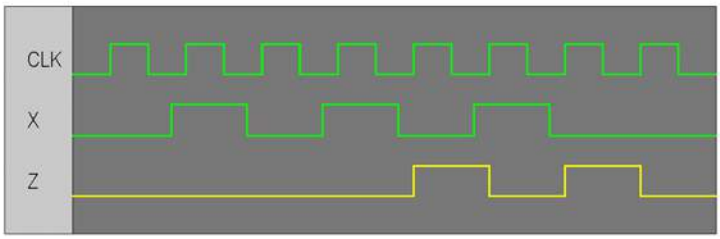

Figura 6.11. Semnalele de intrare şi de ieşire ale circuitului din Figura 6.7 vizualizate la simularea funcțională a circuitului.

2.2.1.4. Maparea tehnologică

Etapele 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 şi adaptarea acesteia la particularitățile şi resursele disponibile ale circuitului utilizat pentru implementare. Operațiile executate în această etapă diferă în funcție de sistemul de proiectare. Cele mai obişnuite operații sunt: adaptarea la elementele fizice ale circuitului, optimizarea şi verificarea regulilor de proiectare (de exemplu, testarea depăşirii 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 şi 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 şi 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 bistabile D în locul bistabilelor JK, iar porțile ŞI au fost înlocuite cu porți ŞI-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.

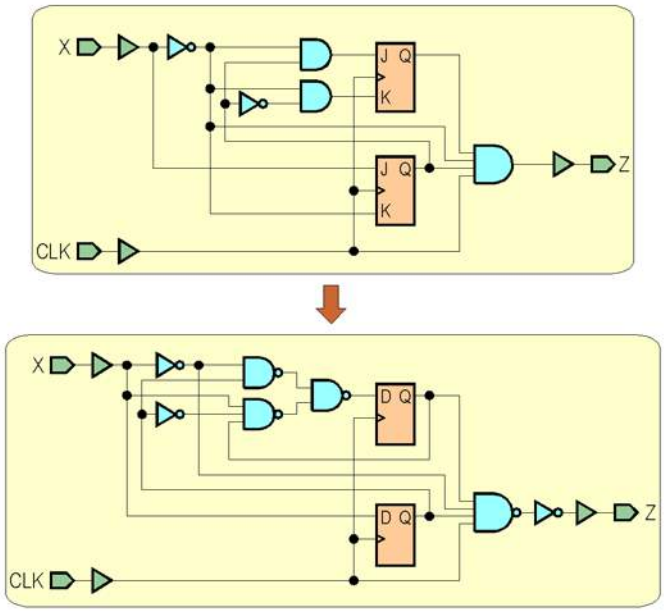

Figura 6.12. Ilustrarea etapei de mapare tehnologică pentru circuitul din Figura 6.7.

2.2.1.5. Plasarea şi 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 şi rutare în mod automat, astfel încât utilizatorul nu trebuie să cunoască detaliile arhitecturii circuitului utilizat pentru implementare. Anumite sisteme permit utilizatorilor experți plasarea şi rutarea manuală a unor porțiuni critice ale sistemului digital pentru a obține performanțe superioare.

Figura 6.13 ilustrează plasarea şi 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 şi verticale. Intrările şi ieşirile 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.

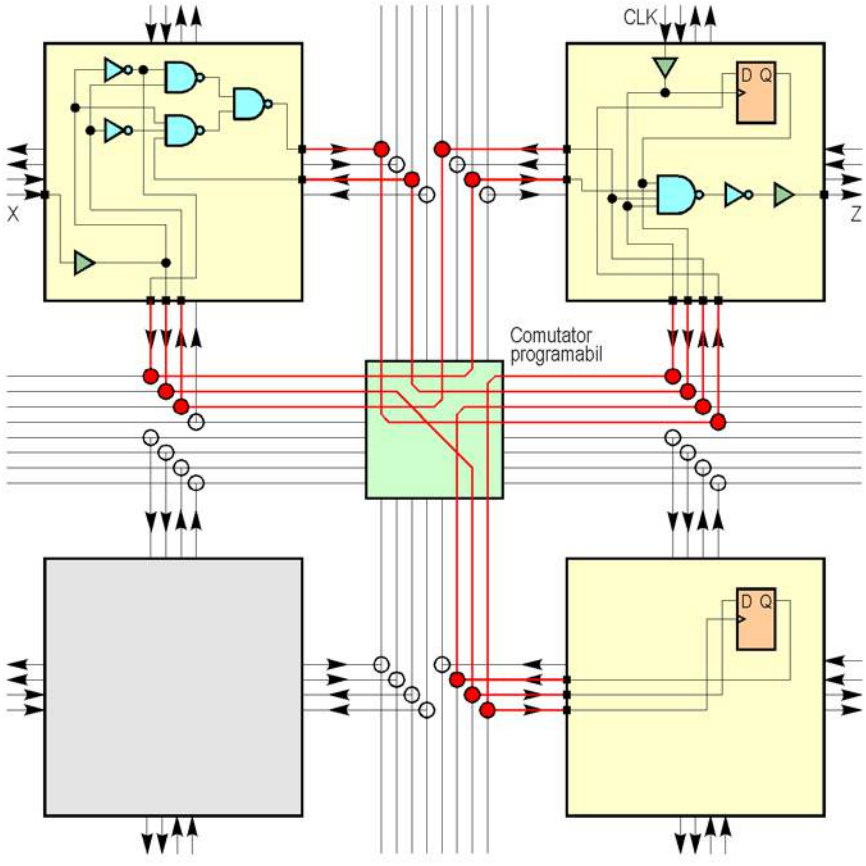

Figura 6.13. Ilustrarea etapelor de plasare şi rutare pentru circuitul din Figura 6.12.

Operațiile de plasare şi rutare pot necesita un timp ridicat pentru execuție în cazul sistemelor digitale complexe, deoarece sunt necesare operații complexe pentru determinarea şi configurarea blocurilor logice necesare din cadrul circuitului programabil, interconectarea corectă a acestora şi 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 conţin de obicei un program numit analizor de timp, care poate furniza informaţii despre întârzierile semnalelor. Aceste informaţii se referă atât la întârzierile introduse de blocurile logice, cât şi la întârzierile datorate interconexiunilor. Analizorul poate afişa aceste informaţii în diferite moduri, de exemplu, prin ordonarea conexiunilor în ordinea descrescătoare a întârzierilor semnalelor. Proiectantul poate utiliza informaţiile despre întârzierile semnalelor pentru a realiza o nouă simulare a sistemului, în care să se ţină cont de aceste întârzieri. Această operaţie prin care se furnizează simulatorului informaţii detaliate despre întârzierile semnalelor se numeşte adnotare inversă ("*back-annotation*").

2.2.1.7. Configurarea sau programarea circuitului

Operaţia de configurare se referă la circuitele programabile bazate pe memorii volatile SRAM (*Static Random Access Memory*) şi constă din încărcarea informaţiilor de configurare în memoria circuitului. Operaţia de programare se referă la circuitele programabile bazate pe memorii nevolatile (cum sunt circuitele care conţin antifuzibile). Această operaţie se execută similar cu cea de configurare, dar informaţiile de configurare sunt păstrate şi după întreruperea tensiunii de alimentare.

La sfârşitul operaţiilor de plasare şi rutare, se generează un fişier care conţine toate informaţiile necesare pentru configurarea circuitului. Aceste informaţii se referă atât la configurarea blocurilor logice ale circuitului, cât şi la specificarea interconexiunilor dintre blocurile logice. Fişierul în care se înscriu aceste informaţii conţine, în principiu, un şir de biţi ("*bitstream*"), fiecare bit indicând starea închisă sau deschisă a unui comutator. Circuitele programabile conţin 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 şirul de biţi va determina închiderea unui comutator şi, deci, stabilirea unei conexiuni. Biţii din acest fişier de configurare sunt aranjaţi într-un anumit format pentru a realiza o corespondenţă între un bit şi comutatorul corespunzător.

Conţinutul fişierului 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 funcţie de valorile biţilor din şirul de configurare. După terminarea configurării, circuitul va funcţiona conform descrierii sistemului digital care a fost implementat.

Din cauza memoriei volatile, circuitul trebuie configurat din nou după fiecare întrerupere a tensiunii de alimentare. Informaţiile 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 interfaţa paralelă a calculatorului. Pentru aceasta, este necesar ca placa cu circuitul programabil să conţină un conector pentru interfaţa paralelă, pentru transfer utilizându-se un cablu paralel. Figura 6.14 ilustrează un cablu paralel obişnuit, care conţine conectori cu 25 de contacte DB25.

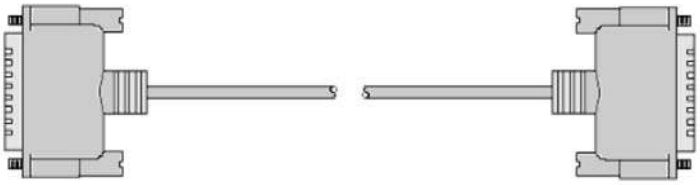

Figura 6.14. Cablu paralel care poate fi utilizat pentru configurarea circuitelor programabile.

O altă posibilitate este utilizarea unui cablu special şi a unei metodologii de configurare propuse de organizaţia JTAG (*Joint Test Advisory Group*). Această metodologie, cunoscută şi sub numele de "*Boundary-Scan*", a fost standardizată de institutele IEEE (*Institute of Electrical and Electronic Engineers*) şi ANSI (*American National Standards Institute*) ca standardul 1149.1, reprezentând un set de reguli de proiectare care facilitează configurarea sau programarea circuitelor, testarea şi depanarea acestora. Un capăt al cablului JTAG se conectează la interfaţa 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 şi sunt transferate serial (bit cu bit) la circuitul programabil. Un asemenea cablu permite şi 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 şi 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 *MultiLINX* 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 şi cablu de configurare, cât şi ca programator pentru memorii PROM şi circuite CPLD CoolRunner II cu ajutorul unor adaptoare. Cablurile *Parallel Cable* III şi *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).

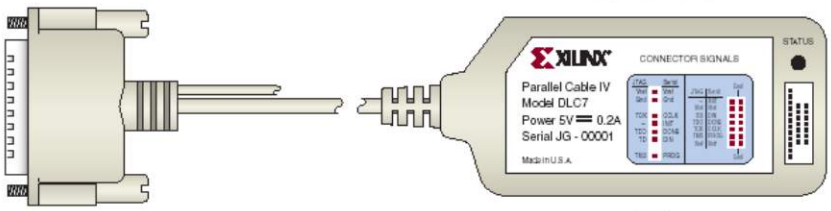

Figura 6.15. Cablul paralel JTAG *Parallel Cable* IV al firmei Xilinx.

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 şi pentru configurarea circuitelor în alte moduri decât modul JTAG, cum sunt modurile "*Slave Serial*" sau "*Slave Parallel*".

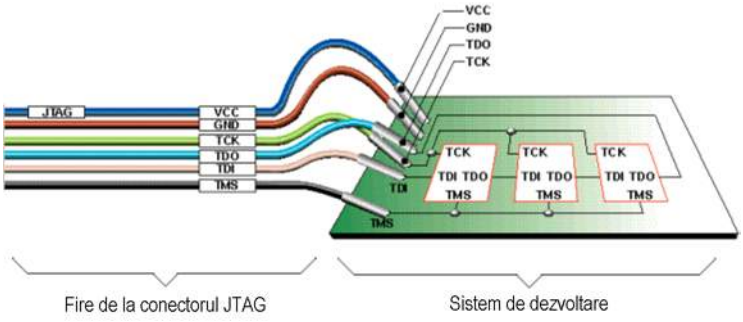

Figura 6.16. Conectarea unui cablu JTAG la un sistem de dezvoltare în modul "*Boundary-Scan*".

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

Tabelul 6.1. Denumirea şi semnificaţia semnalelor JTAG utilizate pentru configurarea circuitelor.

Denumire	Semnificaţie
VCC	Alimentare - Tensiunea de alimentare (5 V, 3,3 V sau 2,5 V)
GND	Masă - Referinţa pentru masa electrică
TCK	*Test Clock* - Semnal de ceas pentru circuitele programabile
TDO	*Test Data Out* - Semnal pentru citirea datelor de la circuitele programabile
TDI	*Test Data In* - Semnal pentru transmiterea instrucţiunilor şi datelor la circuitele programabile
TMS	*Test Mode Select* - Semnal decodificat de controlerul JTAG pentru controlul operaţiilor

2.2.1.8. Depanarea sistemului

În această ultimă etapă a procesului de proiectare se verifică funcţionarea sistemului digital proiectat în condiţii reale. O funcţionare necorespunzătoare se poate datora nerespectării specificaţiilor de proiectare, a specificaţiilor circuitului utilizat pentru implementare, a unor aspecte legate de întârzierea semnalelor etc. Depanarea poate fi simplificată dacă circuitul se configurează astfel încât să conţină unele module speciale care permit citirea valorii unor semnale în timpul funcţionării şi transferul acestor informaţii la calculator, utilizând un cablu JTAG şi 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 conţine 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 aceluiaşi 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 şi 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 conţine o versiune a simulatorului ModelSim al firmei Model Technology (http://www.model.com), denumită MXE (*ModelSim Xilinx Edition*). Acest simulator permite simularea funcţională a descrierilor înainte de sinteză, sau simularea după procesul de implementare pentru verificarea întârzierii semnalelor. Sistemul WebPACK conţine de asemenea o interfaţă grafică pentru specificarea unor vectori de test. Pe baza acestora se generează un banc de test sub forma unui fişier HDL, care va fi compilat împreună cu descrierea sistemului digital pentru simularea funcţionării acestuia.

Sistemul WebPACK permite utilizarea următoarelor tipuri de circuite FPGA şi 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.

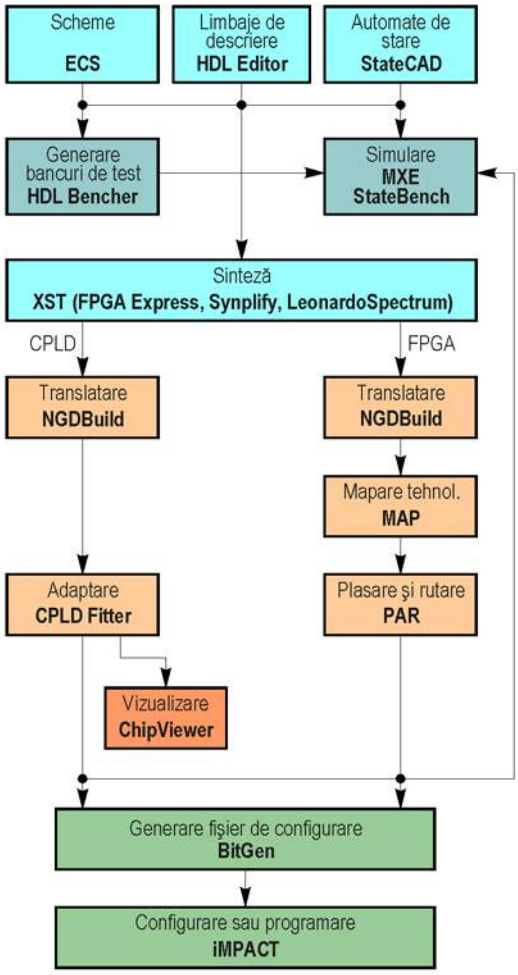

Figura 6.17. Structura sistemului de proiectare Xilinx WebPACK.

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 fişier care va conține informațiile principale despre sistemul proiectat: fişierele care conțin descrierea sistemului digital, tipul circuitului care va fi utilizat pentru implementare etc. Toate fişierele care conțin descrierea sistemului digital şi 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 şi fişierele asociate cu un anumit proiect. Acest "navigator" vizualizează fişierele de intrare ale proiectului, fişierele intermediare sau de ieşire generate pe parcursul etapelor de proiectare, procesele care pot fi executate asupra diferitelor fişiere şi 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 fişierelor sursă* se afişează toate fişierele sursă care au fost incluse în proiectul curent. Prin execuția unui clic dublu pe numele unui fişier din această fereas-

tră se va deschide fişierul respectiv de către modulul CAD corespunzător. *Fereastra proceselor* indică toate procesele care sunt disponibile pentru un anumit tip de fişier sursă. Execuţia unui clic dublu pe numele unui proces va lansa în execuţie procesul respectiv. *Fereastra editorului HDL* afişează conţinutul fişierului sursă selectat, conţinând descrierea unei componente digitale într-un limbaj de descriere hardware. Conţinutul fişierului poate fi editat în această fereastră. În *fereastra consolei* se afişează toate mesajele care indică starea execuţiei unui proces, inclusiv mesajele de avertisment sau de eroare care sunt generate în urma execuţiei.

Pe lângă aceste ferestre principale, interfaţa *Project Navigator* poate deschide şi alte ferestre necesare pentru unele aplicaţii.

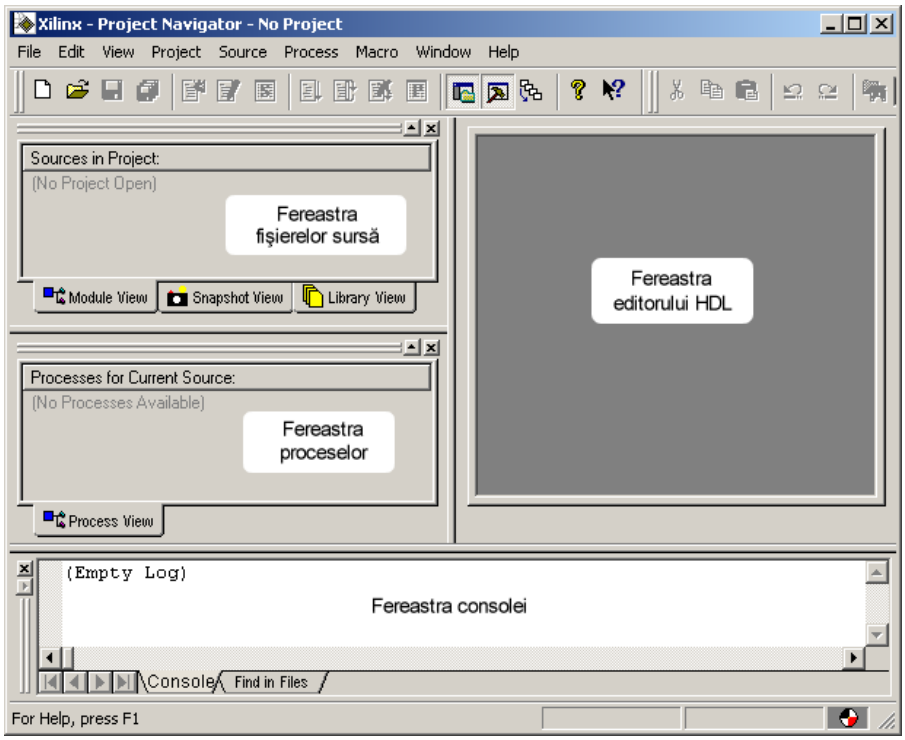

Figura 6.18. Ecranul principal al interfeţei grafice *Project Navigator*.

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 şi editorul pentru diagrame de stare StateCAD. Editorul schematic utilizează un modul care conţine primitivele de bibliotecă pentru circuitele FPGA şi CPLD care pot fi utilizate pentru implementarea sistemului digital. Editorul HDL permite editarea fişierelor sursă care conţin 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, tranziţiile între stări şi poate specifica semnalele de ieşire care trebuie generate în fiecare stare.

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

2.3.2.3. Modulul HDL Bencher

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 şi rezultatele aşteptate ale simulării, ceea ce permite simulatorului să afişeze mesaje de avertisment dacă rezultatele obținute prin simulare nu coincid cu cele aşteptate.

2.3.2.4. Simulatoarele MXE şi StateBench

Simulatorul *ModelSim Xilinx Edition* (MXE) permite atât o simulare funcțională a sistemului digital proiectat, cât şi 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 şi 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 şi cele datorate interconexiunilor.

Pentru simularea funcționării automatelor de stare descrise cu ajutorul editorului StateCAD se poate utiliza simulatorul StateBench. Acest simulator permite şi 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 obişnuit este cea care indică modul în care se doreşte 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 şi 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 fişiere 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 şi 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 şi realizează adaptarea acesteia la particularitățile circuitului CPLD din familia XC9500 sau CoolRunner, astfel încât să se utilizeze numărul minim de macrocelule şi termeni produs. Modulul CPLD Fitter poate genera şi un raport conținând caracteristici de timp ale semnalelor, raport care indică şi 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 fişierul 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 fişier în format NCD (*Native Circuit Description*), care specifică elementele fizice care vor fi utilizate din circuitul FPGA. Programul MAP generează un raport cu rezultatele operaţiilor efectuate. Raportul conţine mesajele de eroare şi de avertisment care au fost afişate, logica eliminată în urma optimizărilor efectuate, numărul şi procentul de utilizare a blocurilor logice, a blocurilor de I/E şi a bistabilelor.

După maparea tehnologică, se rulează programul PAR, care execută operaţiile de *plasare* şi *rutare* a circuitului FPGA. Acest program prelucrează fişierul în format NCD obţinut în urma mapării tehnologice şi generează un fişier în acelaşi format care conţine şi informaţiile de rutare. Programul PAR generează mai multe rapoarte care conţin: numărul semnalelor care nu au putut fi rutate complet, caracteristici de timp ale semnalelor, pinii circuitului sortaţi mai întâi după numele semnalelor, iar apoi după număr, frecvenţa maximă de funcţionare a circuitului.

După implementare, se poate genera un model al sistemului proiectat conţinând şi întârzierile semnalelor, aceste informaţii fiind adăugate prin operaţia 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 şi unul din programele NGD2VHDL, NGD2VER sau NGD2EDIF, în funcţie de fluxul de proiectare utilizat (XST VHDL, XST Verilog, respectiv EDIF).

2.3.2.7. Modulul pentru generarea fişierului de configurare

După executarea etapelor de implementare, se poate genera un fişier care va fi utilizat pentru configurarea circuitului programabil astfel încât acesta să execute funcţiile dorite ale sistemului digital proiectat. Fişierul de configurare este sub forma unui şir de biţi ("*bitstream*"), conţinând toate informaţiile din fişierul NCD care definesc logica internă şi interconexiunile circuitului utilizat pentru implementare, plus alte informaţii din unele fişiere asociate cu acest circuit. Asemenea informaţii sunt, de exemplu, cele care indică asignarea semnalelor la pinii circuitului. Fişierul de configurare este generat de programul BitGen. Acest fişier poate fi utilizat apoi pentru configurarea circuitului sau pentru crearea unui fişier necesar programării unei memorii PROM cu informaţiile de configurare ale circuitului.

Pentru circuitele CPLD, fişierul de configurare este creat în formatul JEDEC (*Joint Electron Device Engineering Council*) şi are extensia **.jed**. Ulterior, se poate crea un fişier în formatul SVF (*Serial Vector Format*) cu ajutorul modulului iMPACT. Pentru circuitele FPGA, fişierul de configurare creat este un fişier binar şi 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). Fişierul care conţine şirul de biţi pentru configurare va fi transferat la circuitul programabil.

Modulul iMPACT conţine şi un program (*PROM File Formatter*) pentru generarea fişierului necesar programării unei memorii PROM. O asemenea memorie nevolatilă poate fi utilizată pentru păstrarea informaţiilor de configurare necesare circuitelor CPLD sau FPGA care se bazează pe tehnologia SRAM şi necesită configurarea după fiecare întrerupere a tensiunii de alimentare. Fişierul generat de modulul iMPACT poate avea formatul MCS-86 (Intel), EXORMACS (Motorola) sau TEKHEX (Tektronix). Se poate genera şi un fişier HEX, care conţine reprezentarea hexazecimală a şirului de configurare. Fişierul 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 şi de ieşire, a ecuaţiilor sem-

nalelor a modului de asignare a pinilor de I/E şi 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 ieşire ale numărătorului vor fi conectate la segmentul de sus şi cel de jos al afişajului 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.

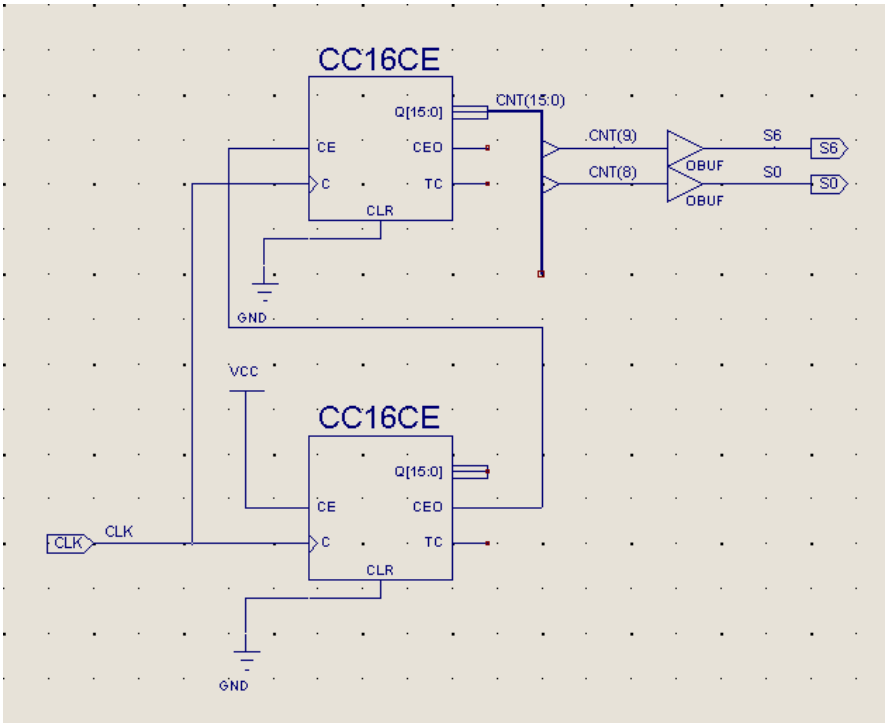

Figura 6.19. Schema circuitului utilizat ca exemplu de proiectare.

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 şi 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 fişierele 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 şi capacitatea circuitului, **-5** este un indicator al vitezei circuitului, iar **tq144** reprezintă tipul capsulei circuitului şi 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 fişierului 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 afişate în fereastra *Categories* a editorului schematic, iar componentele dintr-o categorie selectată sunt afişate 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 ataşat cursorului simbolul unui numărător. Se deplasează cursorul şi se plasează numărătorul prin execuţia unui clic în poziţia 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 şi execuţia unui clic în poziţia dorită.
5. Se apasă tasta **Esc** pentru terminarea plasării componentelor.

Componentele adăugate pot fi deplasate ulterior în poziţiile convenabile. Ştergerea unei componente se poate realiza prin selectarea acesteia şi apăsarea tastei **Delete**.

Se pot utiliza butoanele *Zoom In*  şi *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 roşie plasate în colţurile 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 mulţi pini, dar atunci se execută un clic dublu doar pe ultimul pin. Terminarea interconectării se poate realiza şi prin apăsarea tastei **Esc**.

Este posibilă şi 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 şi se execută un clic dublu în poziţia dorită. La această conexiune se va ataşa 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 ieşire *CEO* (*Count Enable Out*) al numărătorului de jos la terminalul de validare al ceasului *CE* (*Clock Enable*) al numărătorului de sus. Pentru a obţine forma dorită a conexiunii, se poate executa un clic în punctele intermediare în care se doreşte schimbarea direcţiei liniei de conexiune.

Prin această conectare a terminalelor *CEO* şi *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 ieşirea *CEO* a numărătorului de jos va trece în 1 logic numai atunci când valoarea numărătorului este FFFFh. Prin conectarea ieşirii *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 şi conectarea simbolului **VCC**. Fiecare numărător are o intrare de ştergere *CLR* (*Clear*) care resetează 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 şi conectarea simbolului **GND**.

Pentru conectarea intrării *CE* la 1 logic şi 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 şi se verifică forma simbolului ataşat 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 şi 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** şi 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 şi 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** şi terminalele *CLR* ale numărătoarelor.

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

2.4.7. Adăugarea şi conectarea unei magistrale

În cazul editorului ECS, o magistrală este ca şi 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 şi 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 afişajului de pe placa XSA-50. Pentru a avea acces la aceşti biți, mai întâi se prelungeşte magistrala de ieşire *Q* a numărătorului de sus şi 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 ieşirea *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)** şi se apasă tasta **Enter**. Prin aceasta se ataşează cursorului numele magistralei.
5. Se execută un clic pe capătul de sus al magistralei pentru a se ataşa numele acesteia. Se apasă apoi tasta **Esc**.

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

fiind reprezentați de biții 9, respectiv 8, ai numărătorului de sus. Pentru accesul la aceşti 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 afişat *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 şi conectarea bufferelor de ieşire

Semnalele de ieşire se vor conecta la pinii de ieşire prin intermediul unor buffere de ieşire. Pentru adăugarea şi 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 aceştia. Se apasă apoi tasta **Esc**.
3. Se trasează câte o linie de conexiune de la fiecare conector al magistralei la bufferul de ieşire corespunzător.
4. Se prelungesc la dreapta liniile de la ieşirile componentelor **obuf**.

2.4.9. Ataşarea numelor la conexiuni

În continuare se vor ataşa 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** şi se apasă tasta **Enter**.
3. Se execută un clic în partea stângă a liniei conectate la intrarea de ceas *C* a numărătorului de jos şi se verifică ataşarea numelui **CLK** la această linie. Se apasă tasta **Esc**.
4. Se procedează similar pentru ataşarea numelui **CNT(9)** la linia dintre conectorul de sus al magistralei şi bufferul de ieşire corespunzător.
5. Se ataşează numele **CNT(8)** la linia dintre conectorul de jos al magistralei şi bufferul de ieşire corespunzător.
6. Se ataşează numele **S6** la linia din dreapta bufferului de ieşire conectat la semnalul *CNT(9)*.
7. Se ataşează numele **S0** la linia din dreapta bufferului de ieşire 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 numărătorul proiectat, singurul port de intrare este cel de ceas. Porturile de ieşire sunt liniile **CNT(9)** şi **CNT(8)** ale numărătorului. 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 şi, 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 şi, cu butonul apăsat, se trasează un dreptunghi în jurul semnalelor *S6* şi *S0*. Se observă adăugarea a două simboluri pentru porturile de ieşire. 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 afişează 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 şi se repetă testul.

2.4.12. Asignarea pinilor la semnalele de intrare şi ieşire

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

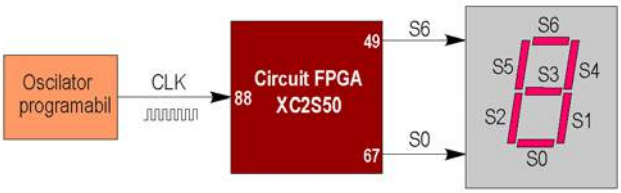

Figura 6.21. Conectarea oscilatorului programabil şi a afişajului cu 7 segmente la pinii circuitului FPGA de pe placa XSA-50.

Pinii la care se vor conecta semnalele de intrare şi de ieşire se pot specifica prin ataşarea 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* şi 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 afişajului cu 7 segmente).
6. Se procedează similar pentru conectarea semnalului *S0* la pinul 67 al circuitului FPGA (segmentul S0 al afişajului).

   Se salvează schema utilizând butonul *Save* .

2.4.13. Sinteza şi implementarea proiectului

După terminarea editării schemei, se revine în fereastra *Project Manager*. În fereastra fişierelor sursă se selectează fişierul care conține schema (fişierul cu extensia `.sch`), dacă fişierul nu este selectat. În fereastra proceselor se vor afişa procesele disponibile pentru acest fişier.

1. Pentru lansarea procesului de sinteză, se execută un clic dublu pe numele procesului *Synthesize* din fereastra proceselor. În timpul sintezei, se afişează diferite mesaje în fereastra consolei. După terminarea sintezei, se verifică dacă s-au afişat mesaje de eroare (marcate cu culoarea roşie). În cazul unor erori, se revine la editorul schematic şi 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 afişa conținutul fişierului de raport generat în urma sintezei. Se parcurge fişierul, iar în secțiunea "*Timing Summary*" se verifică frecvența maximă de funcționare a circuitului. Se închide apoi fişierul 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 fişierul afişat, 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*) şi numărul de porți echivalente (*Total equivalent gate count for design*). Se închide apoi fişierul de raport.
5. Se execută un clic pe semnul + din stânga numelui *Place & Route* din cadrul procesului *Implement Design*. Se vizualizează fişierul de raport care conține asignarea pinilor de I/E (*Pad Report*), generat în urma proceselor de plasare şi rutare (*Place & Route*). Se verifică dacă semnalele sunt asignate în mod corect la pinii de I/E. Se închide apoi fişierul de raport.
6. Pentru generarea fişierului de configurare, se execută un clic dublu pe numele procesului *Generate Programming File* din fereastra proceselor. Se verifică în directorul proiectului existența unui fişier cu extensia `.bit`, care este fişierul 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 fişierului 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 fişierelor din directorul proiectului. Se reduce dimensiunea ferestrei acestui program pentru a fi vizibilă şi fereastra programului GXSLOAD.
5. Se selectează fişierul cu extensia `.bit` şi se plasează acest fişier în fereastra FPGA/CPLD a programului GXSLOAD.
6. Se execută un clic pe butonul *Load*.

Fişierul 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 afişajului 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. Desfăşurarea lucrării

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

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

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

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

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