PROIECTAREA SISTEMELOR DIGITALE

Masterat ICCP - an 1

LABORATOR 2

PROIECTAREA STRUCTURALĂ ÎN LIMBAJUL VHDL

2.1. Avantajele proiectării structurale

Descrierile structurale specifică un sistem sub forma unor componente interconectate. Aceste descrieri permit crearea unor *nivele ierarhice* multiple, în care un proiect este divizat în unități de proiectare de dimensiuni mai mici. Fiecare unitate de proiectare sau componentă poate fi specificată fie printr-o descriere funcțională, fie printr-o descriere structurală. În ultimul caz, o componentă poate fi formată din mai multe subcomponente, care pot fi specificate la rândul lor sub forma unor descrieri structurale. În final, fiecare componentă primitivă de la nivelul cel mai de jos este specificată sub forma unei descrieri funcționale.

Proiectarea structurală poate fi realizată, cel puțin în cazul descrierilor la nivelul transferurilor între registre (RTL), prin utilizarea componentelor. Orice pereche entitate-arhitectură poate fi utilizată ca o componentă într-o arhitectură de nivel superior. Astfel, sistemele complexe pot fi construite în mai multe etape din componente de nivel inferior.

Principalele avantaje ale proiectării structurale sunt următoarele:

Proiectarea structurală pe mai multe nivele ierarhice permite definirea detaliilor unei porțiuni a proiectului la un moment dat, de preferință în paralel cu alți proiectanți.

Fiecare componentă poate fi proiectată şi testată individual, înainte de a fi integrată în nivelele superioare ale proiectului. Această testare a nivelelor intermediare este mai simplă decât testarea în cadrul sistemului, şi este de obicei mai completă. Aceasta înseamnă că proiectantul poate avea un grad mai ridicat de încredere în componentele utilizate, ceea ce contribuie şi la integritatea globală a sistemului.

Componentele utile pot fi colectate în biblioteci, astfel încât ele pot fi reutilizate ulterior în acelaşi proiect sau în alte proiecte. Unul din avantajele sintezei logice este că asemenea componente sau module sunt independente de tehnologie. Gradul de reutilizare a componentelor creşte pe măsură ce sunt disponibile mai multe componente.

2.2. Elementele unei descrieri structurale

O descriere structurală constă din componente interconectate prin intermediul semnalelor. O componentă poate fi definită în cadrul unei arhitecturi cu ajutorul unei declarații **component**, sau poate fi reprezentată de un sistem separat, care este specificat sub forma unei entități şi a unei arhitecturi. Pentru utilizarea unei componente declarate anterior, aceasta trebuie *instanțiat*ă în cadrul descrierii structurale.

Instanțierile componentelor reprezintă instrucțiunile de bază într-o arhitectură structurală. Aceste instanțieri sunt concurente unele față de altele. În cadrul instanțierii unei componente se specifică *maparea porturilor*, care indică semnalele conectate la porturile componentei. Aceste semnale pot fi specificate ca porturi sau pot fi semnale interne ale sistemului. În ultimul caz, acestea trebuie declarate în secțiunea declarativă a arhitecturii.

2.2.1. Exemplu de descriere structurală

Elementele unei descrieri structurale vor fi ilustrate mai întâi printr-un exemplu complet. Componentele descrierii structurale vor fi examinate apoi separat în secțiunile următoare. Exemplul prezentat constă din două bistabile de tip D conectate în serie, sub forma unui sistem *pipeline*. Structura circuitului este ilustrată în **Figura 2.1**.

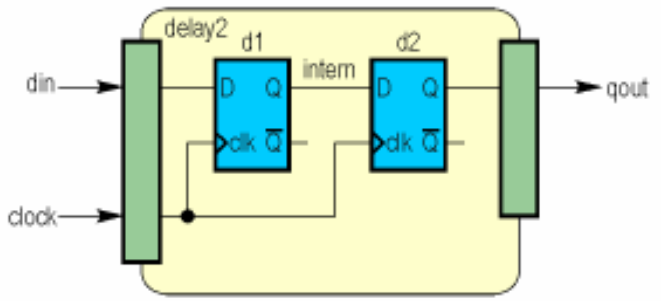


Figura 2.1. Exemplu de circuit pentru ilustrarea descrierii structurale.

Presupunem că bistabilul de tip D este definit într-o bibliotecă, având definiția de entitate şi arhitectură prezentată în Exemplul 1.

Exemplul 1

```
library ieee;
use ieee.std_logic_1164.all;

entity Dff is
  port (D, Clk: in std_logic;
        Q, Qn:  out std_logic);
end Dff;
 architecture arh_dff of Dff is
begin
  process
  begin
    wait until Clk = '1';
    Q <= D;
  end process;
end arh_dff;
```

Există mai multe posibilități pentru descrierea circuitului considerat cu ajutorul componentelor. O posibilitate de descriere este prezentată în Exemplul 2.

Exemplul 2

```
library ieee;
use ieee.std_logic_1164.all;

entity delay2 is
  port (din, clock: in std_logic;
        qout: out std_logic);
end delay2;
 architecture structural of delay2 is
  signal intern: std_logic;
```

```
  -- Declarația componentei
  component dff is
    port (d, clk: in std_logic;
          q, qn:  out std_logic);
  end component dff;
  -- Specificarea configurației
  for all: dff use entity work.Dff (arh_dff)
    port map (D => d, Clk => clk, Q => q, Qn => qn);
begin
  -- Instanțierile componentei
  d1: dff port map (d => din, clk => clock, q => intern, qn => open);
  d2: dff port map (d => intern, clk => clock, q => qout, qn => open);
end structural;
```

Arhitectura conține trei părți care se referă la utilizarea componentelor. Acestea sunt indicate prin comentarii, fiind următoarele: declarația componentei, specificarea configurației şi instanțierile componentei. Cele trei părți sunt descrise în secțiunile următoare.

2.2.2. Declarația componentelor

Declarația unei componente defineşte interfața cu o entitate de proiectare care descrie componenta respectivă. Componenta declarată astfel poate fi utilizată ulterior în instrucțiunile de instanțiere a componentei respective. Declarația componentei nu specifică însă care este perechea entitate-arhitectură care descrie componenta şi nici porturile entității; aceste informații sunt conținute în specificația configurației sau în declarația configurației.

Sintaxa simplificată a declarației unei componente este următoarea:

```
component nume_componentă [is]
        generic (listă_generice);
        port (listă_porturi);
end component [nume_componentă];
```

Sintaxa declarației unei componente este similară cu cea a declarației de entitate. Clauza **generic** specifică lista genericelor componentei, iar clauza **port** specifică porturile acesteia. În practică, numele componentei, numele genericelor şi a porturilor acesteia, ca şi ordinea lor, sunt identice cu cele care apar în declarația entității corespunzătoare componentei.

O componentă poate fi declarată într-o arhitectură, un bloc, o entitate sau un pachet. În cazul în care componenta este declarată într-o arhitectură, aceasta trebuie plasată în partea declarativă a arhitecturii, înaintea cuvântului cheie **begin**. În acest caz, componenta poate fi utilizată (instanțiată) numai în cadrul arhitecturii respective. Atunci când componenta este declarată într-un pachet, ea va fi vizibilă în toate arhitecturile care utilizează pachetul respectiv.

Declarația componentei dff din Exemplul 2 este reprodusă în continuare:

```
component dff is
        port (d, clk: in std_logic;
              q, qn:  out std_logic);
end component dff;
```

2.2.3. Instanțierea componentelor

Instanțierea unei componente asociază semnale sau valori cu porturile unei componente definite anterior şi asociază valori cu genericele componentei respective. Sintaxa simplificată a unei instanțieri de componente este următoarea:

etichetă*: [component] *nume_componentă
[generic map (*listă_asociere_generice*)]
port map (*listă_asociere_porturi*);

Instanțierea unei componente introduce o relație cu o unitate declarată anterior ca o componentă.Numele componentei instanțiate trebuie să corespundă cu numele componentei declarate anterior. Pentru componenta instanțiată se specifică genericele şi porturile, care reprezintă parametrii actuali ai componentei declarate. Lista de asociere poate fi specificată prin nume sau poate fi poziționlă. Asocierea prin nume permite listarea genericelor şi a porturilor într-o ordine care este diferită de ordinea specificată în declarația componentei. În acest caz, fiecărui generic sau port i se asociază în mod explicit o valoare sau un semnal. Numele genericului, respectiv a portului, este urmat de simbolul =>, iar apoi de valoarea care i se atribuie genericului, respectiv de semnalul la care este conectat portul. Porturile unei componente pot fi lăsate neconectate prin specificarea cuvântului cheie **open**.

În Exemplul 2, pentru porturi s-a utilizat asocierea prin nume. Instanțierile componentei din acest exemplu sunt reproduse mai jos:

```
d1: dff port map (d => din, clk => clock, q => intern, qn => open);
d2: dff port map (d => intern, clk => clock, q => qout, qn => open);
```

Într-o listă de asociere poziționlă, parametrii actuali (genericele şi porturile) sunt specificați în aceeaşi ordine în care apar aceştia în declarația componentei. În acest caz, numele genericelor sau a porturilor şi simbolul => sunt omise. Instanțierile componentei din Exemplul 2 pot fi rescrise prin utilizarea asocierii poziționale în felul următor:

```
d1: dff port map (din, clock, intern, open);
d2: dff port map (intern, clock, qout, open);
```

În Exemplul 2, există două instanțieri ale componentei dff, care sunt etichetate cu d1 şi d2. Aceste etichete sunt obligatorii şi trebuie să fie unice. Fiecare instanțiere crează un subcircuit conținând componenta dff şi conexiunile cu această componentă.

În exemplul anterior, distincția dintre declarația entității şi cea a componentei a fost indicată prin faptul că pentru inițialele numelui entității şi a porturilor acesteia s-au utilizat litere mari, iar pentru instanțierea componentei şi pentru semnalele din circuitul rezultat s-au utilizat litere mici. Această distincție s-a realizat doar pentru claritate şi nu are o semnificație deosebită, deoarece în limbajul VHDL nu se ține cont de tipul literelor.

Observații

- Instanțierile componentelor reprezintă instrucțiuni concurente.
- Instanțierea unei componente reprezintă instanțierea declarației componentei şi nu a entității.
- Relația dintre declarația componentei şi entitatea care descrie componenta este controlată de specificația configurației.

2.2.4. Specificarea configurației

În cazul în care nu se utilizează instanțierea directă a entităților, declarațiile unor componente şi instanțierile acestora nu sunt suficiente pentru o specificare completă a unei arhitecturi structurale, deoarece nu este specificată descrierea implementării componentelor. În acest caz se poate utiliza o specificație a configurației. O *configurație* este o construcție care definește modul în care instanțierile de componente sunt asociate cu entitățile de proiectare şi arhitecturile corespunzătoare.

Motivul pentru separarea unei entități şi a componentelor acesteia este de a permite ca asocierea ("legarea") dintre entitate şi componente să fie realizată cât mai târziu posibil în procesul de simulare. Această asociere nu se realizează decât la începerea simulării, în faza de elaborare. În acest fel, modulele sursă ale unui proiect ierarhic pot fi compilate în orice ordine.

Sintaxa unei specificații a configurației este următoarea:

```
for etichetă_instanțiere: nume_componentă
  use entity nume_bibliotecă.nume_entitate
        [(nume_arhitectură)]
    [generic map (listă_asociere_generice)]
    [port map (listă_asociere_porturi)];
```

Mai multe specificații ale configurației unor componente pot fi incluse într-o declarație a configurației, care poate reprezenta o unitate separată de proiectare, şi deci poate apare într-un fişier separat. Sintaxa unei declarații a configurației este următoarea:

```
configuration nume_configurație of nume_entitate is
  for nume_arhitectură
    -- specificații de configurații
  end for;
  -- alte clauze for
end [configuration nume_configurație];
```

Se observă că sintaxa unei specificații a configurației este asemănătoare cu cea a instanțierii directe a unei entități. Totuşi, specificarea unei configurații reprezintă o metodă mai flexibilă în cazul în care trebuie utilizată o implementare diferită a aceleiaşi componente. Dacă trebuie efectuate anumite modificări, acestea vor fi introduse numai în fişierul de configurație, iar arhitectura structurală va rămâne neschimbată. Utilizarea instanțierii directe a entităților ar necesita ca toate modificările să fie introduse în cadrul arhitecturii.

O specificație a configurației are trei părți. Prima parte indică acele componente la care se referă configurația. Fiecare componentă este indicată prin eticheta instrucțiunii în care este instanțiată componenta respectivă. Este posibilă utilizarea cuvântului cheie **all** pentru selectarea tuturor componentelor cu numele specificat. Acest cuvânt cheie a fost utilizat şi în Exemplul 2, specificația configurației din acest exemplu fiind reprodusă mai jos:

```
for all: dff use entity work.Dff (arh_dff)
  port map (D => d, Clk => clk, Q => q, Qn => qn);
```

În locul specificării configurației pentru toate componentele cu numele dff, se puteau prevedea specificații separate pentru fiecare componentă instanțiată:

```
for d1: dff ...
for d2: dff ...
```

A doua parte a specificației unei configurații selectează entitatea care trebuie utilizată pentru o anumită componentă sau pentru toate componentele cu numele indicat, ca şi biblioteca în care se află entitatea respectivă. Această parte poate specifica de asemenea şi arhitectura care va fi utilizată

pentru entitatea selectată, în cazul în care există mai multe arhitecturi. A treia parte a specificației este opțională. Această parte poate specifica în mod explicit modul în care genericele şi porturile unei componente instanțiate sunt asociate cu genericele şi porturile entității. Pentru aceasta se utilizează clauzele **generic map** şi **port map**, iar asocierea poate fi pozițională sau prin nume. Asocierea explicită este necesară numai dacă numele genericelor şi porturilor din declarația unei componente sunt diferite de numele genericelor şi porturilor din declarația entității utilizate pentru componenta respectivă. În practică se recomandă însă ca aceste nume să fie aceleaşi.

În cazul în care pentru o componentă specificația unei configurații lipseşte complet, se va realiza o asociere implicită. Aceasta înseamnă că pentru acea componentă va fi selectată o entitate cu acelaşi nume din biblioteca curentă, se va utiliza arhitectura compilată cel mai recent, iar genericele şi porturile sunt asociate cu genericele şi porturile cu aceleaşi nume din cadrul entității.

De cele mai multe ori, asocierea implicită este şi cea dorită, astfel încât în aceste cazuri nu este necesară specificarea unei configurații. Există însă un caz în care specificația unei configurații este necesară, şi anume atunci când o componentă trebuie asociată cu o entitate dintr-o bibliotecă diferită. O posibilitate pentru realizarea acestei asocieri ar fi utilizarea clauzelor **library** şi **use** pentru ca toate entitățile din biblioteca respectivă să fie vizibile.

Observații

- În general, sistemele de sinteză nu permit specificarea unei configurații. Proiectantul trebuie să se asigure că numele entităților şi a componentelor, ca şi a genericelor şi a porturilor, sunt aceleaşi.
- Pentru configurația unei entități de proiectare, atât entitatea cât şi configurația trebuie declarate în aceeaşi bibliotecă.

2.3. Biblioteci

O bibliotecă reprezintă un director sau un fişier în care se pot compila unitățile de proiectare. De exemplu, o declarație de entitate şi arhitectura corespunzătoare, aflate într-un fişier, pot fi compilate într-o bibliotecă. Formatul intern al bibliotecii poate fi specific unui anumit sistem de proiectare. Unitățile de proiectare (de exemplu, entitățile) dintr-o bibliotecă pot fi utilizate în cadrul altor entități, dacă biblioteca şi unitățile respective sunt vizibile.

Pentru a se utiliza unitățile de proiectare dintr-un pachet, în primul rând trebuie ca biblioteca din care face parte pachetul să fie accesibilă, ceea ce se obține prin utilizarea unei clauze **library**. Într-o asemenea clauză se pot specifica mai multe nume de biblioteci, separate prin virgule. Pentru utilizarea unităților de proiectare dintr-o anumită bibliotecă, trebuie ca pachetele, componentele, declarațiile, funcțiile şi procedurile respective să fie vizibile, ceea ce se obține prin utilizarea unei clauze **use**.

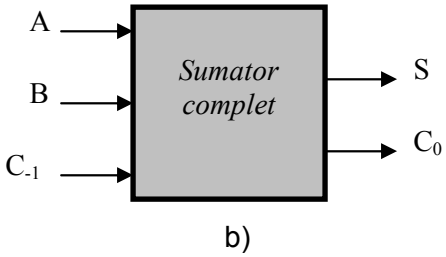
Există două biblioteci predefinite care sunt utilizate în mod implicit în fiecare proiect: **std** şi **work**. Biblioteca **std** conține pachetele cu numele **standard** şi **textio**. Biblioteca **work** este locul în care se plasează, în mod implicit, unitățile de proiectare care sunt compilate în timpul dezvoltării proiectului. După verificarea corectitudinii unei unități de proiectare aflată în biblioteca work, aceasta poate fi compilată într-o altă bibliotecă, dacă aceasta trebuie reutilizată în cadrul aceluiaşi proiect sau în proiectele ulterioare. Bibliotecile **std** şi **work** sunt vizibile în mod implicit pentru toate proiectele, astfel încât pentru aceste biblioteci nu este necesară utilizarea clauzei **library**.

Observații

- bibliotecă specificată într-o clauză **library** înainte de o unitate primară de proiectare (entitate, configurație sau pachet) este vizibilă în fiecare unitate secundară de proiectare (arhitectură sau corp al pachetului) asociată cu unitatea primară de proiectare.

2.4 TEME PROPUSE

1. Să se realizeze o descriere structurală a circuitului din *figura 2.2,* circuit ce reprezintă structura internă a unui sumator numeric de 1 bit.



b)

C_{-1}	A	B	C_0	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

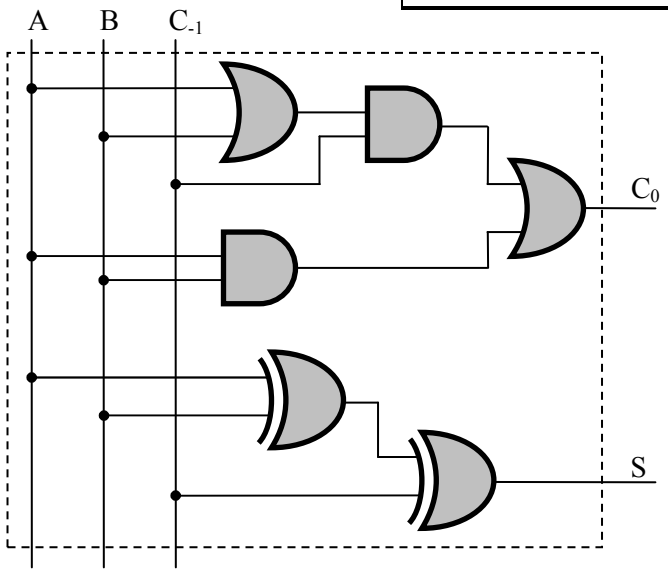


Figura 2.2 *Structura logică a unui sumator elementar*

2. Să se conecteze 4 sumatoare de 1 bit pentru a obține un sumator complet de 4 biți.

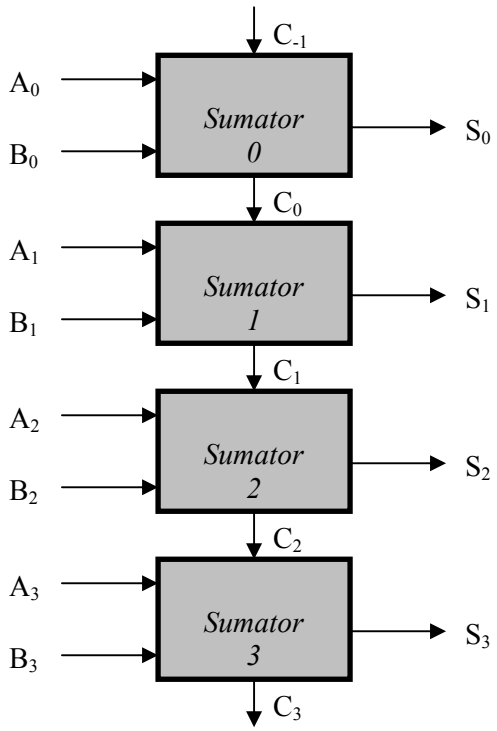


Figura 2.3 *Structura unui sumator de 4 biți*