Razlaganje složenih projekata na manje celine

Prilikom podele složenog projekta na manje celine korisno je uočiti njegove prirodne gradivne ili funkcionalne delove, odnosno blokove. Iako ovo zvuči relativno jednostavno, nije redak slučaj da se ova činjenica prenebregne. Projekat obično sadrži nekoliko funkcija. Treba pristupiti tako da se uoče, izdvoje, a zatim i skiciraju blokovi pojedinih funkcija. Tako je npr., logično da se razvoj kontrolne logike razdvoji od ostatale, „kontrolisane" logike. Prirodno je, takođe, ako u jednom projektu postoji više aritmetičkih kola, ona budu razdvojena. Takođe je korisno razmotriti da li postoje resursi koje mogu da koriste dva ili više delova kola.

Podela projekta na manje celine može da pomogne da se kasnije, tokom razvoja projekta jedni blokovi zamene drugim blokovima, koji imaju poboljšane performanse.

Ne treba posebno naglašavati da se, tokom projektovanja logike, svaki blok tretira kao poseban par entitet-arhitektura. U okviru jedne arhitekture, može se javiti više različitih procesa. Ne postoji pouzdana formula kojom bi se procenilo da li je previše logike grupisano u okviru jednog bloka. Međutim, za početak, može se ispitati odnos između broja portova i složenost logike. Ukoliko je taj odnos mali, verovatno se preteralo sa logikom koja je pripisana jednom entitetu. Ako je taj odnos veliki, najverovatnije je da je jednoj arhitekturi pripisano nedovoljno logike. Naravno, ovo je samo procena, a ne striktno pravilo. Kao takve treba prihvatiti i sledeće preporuke, koje imaju smisla naročito sa stanovišta sinteze:

- jedan trostatički element po entitetu/arhitekturi, sem u slučaju da blok predstavlja bus (magistralu).
- jedan taktni signal po entitetu/arhitekturi.
- jedan reset (ili preset) signal po entitetu/arhitekturi.

Sa stanovišta *moda* i *tipa* signala u okviru projekta poželjno je da:

- svi portovi budu tipa ***std_logic*** ili ***std_logic_vector***,
- svi portovi budu moda ***in*** ili ***out***,
- samo portovi entiteta na najvišem nivou mogu da budu moda ***inout.***

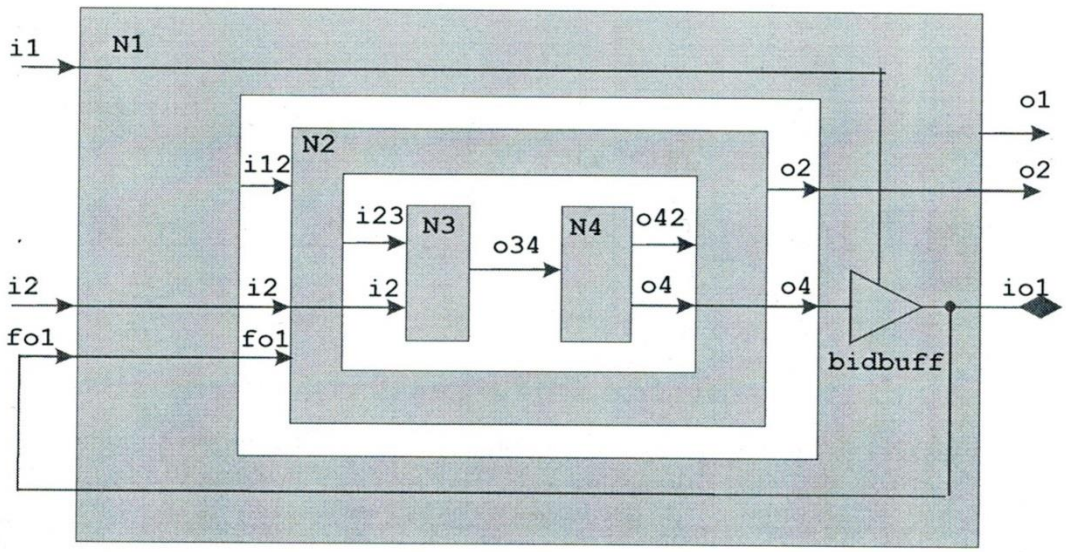
Tip ***std_logic*** je standardan i podržavaju ga svi alati za sintezu. On obezbeđuje potrebnu fleksibilnost jer podržava osim stanja „0", „1", „-"„Z" i „X".

Ne treba koristiti apstraktne tipove kao što su intedžeri i prebrojivi tipovi (*enumerated)* u port mapi. Ove tipove treba zadržati kao lokalne u okviru arhitekture, a koristiti funkcije za konverziju iz, ili u, tip ***std_logic***.

Mod signala takođe ima uticaja na podelu kola. Mod signala mora hijerarhijski da se prosledi do svih entiteta na koje se odnosi.

Sl. 2.7.1 poslužiće za ilustrovanje hijerarhije prosleđivanja signala sa stanovišta *moda* i *tipa* portova u složenom projektu. Napominjemo da primena samo ***in*** i ***out*** modova za opis entiteta na nižim hijerarhijskim nivoima pomaže u dokumentovanju projekta i njegovoj čitljivosti jer se jasno definiše smer protoka signala između pojedinih delova projekta.

Na Sl. 2.7.1 prikazan je projekat definisan entitetom nazvanim `N1`. On u sebi **sadrži** entitet `N2` u kome se nalaze i entiteti `N3` i `N4`, na najnižem hijerarhijskom nivou.



Sl. 2.7.1 Hijerarhija signala u složenom projektu (primer P. 2.7.1)

Ulazni signali u `N1` označeni su sa **`i1`, `i2`** i **`fo1`**, izlazni su **`o1`**, i **`o2`**, dok je sa **`io1`** označen bidirekcioni signal koji je moda ***inout***.

Iz bidirekcionog signala io1, za entitet N1, izdvaja se na „izlazu" iz bidirekcionog bafera **`bidbuff`**, povratni signal (feed-back) označen sa `fo1`. On predstavlja samo ulazni signal za entitete `N1` i `N2`, tako da je kao takav i naveden u deklaraciji portova u primeru P. 2.7.1. Na taj način put signala kroz projekat postaje očigledan.

Signal `i2` je ulazni signal i za entitet `N2` koji se nalazi u sastavu `N1`, kao i za `N3` koji se nalazui u satavu `N2`. Zato se taj signal pojavljuje kao port moda ***in*** i u `N2`, i u `N3`. Za entitet `N2`, osim `i2`, ulazni portovi (mod ***in***) su `i12` i `fo1`. Signal `i12` generiše se u okviru `N1`, tako da za njega predstavlja interni signal i nije deklarisan kao port entiteta `N1`.

Za entitet `N3` osim signala `i2`, kao ulazni port deklariše se i signal `i23` koji se generiše u okviru `N2` (čiji je `N3` deo) tako da je `i23` za `N2` interni signal koji se ne definiše u listi portova.

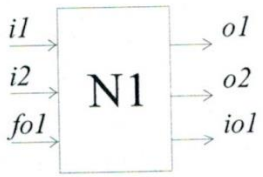
Slična hijerarhija važi i za izlane portove. Kao što je rečeno, izlazni portovi za `N1` su `o1` i `o2`. Signal `o2` generisan je u entitetu `N2`, tako da i za njega on predstavlja izlazni port. Slično važi i za signal `o4`, generisan u `N4`, čija se vrednost preko `N2` prosleđuje entitetu `N1`, za koji predstavlja interni signal. Zato je `o4` deklarisan kao izlazni port u `N4` i `N2`.

Definicija entiteta N1, N2 , N3 i N4 data je u primeru P. 2.7.1 zajedno sa odgovarajućim grafičkim interpretacijama na Sl. 2.7.2 .a, Sl. 2.7.1.b, Sl. 2.7.2 Sl. 2.7.1.c, iSl. 2.7.2 .d, respektivno.

P. 2.7.1

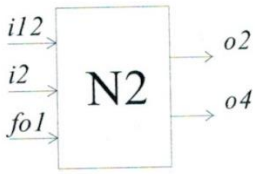
```
library ieee;
use ieee.std_logic_1164.all;

entity N1 is port (
        i1: in std_logic;
        i2: in std_logic;
        fo1: in std_logic;
        o1: out std_logic;
        o2: out std_logic;
        io1: inout std_logic
          );
end entity N1;
```



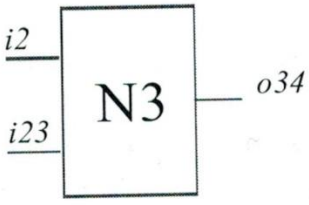
a)

```
library ieee;
use ieee.std_logic_1164.all;
entity N2 is port (
        i12: in std_logic;
        i2: in std_logic;
        fo1: in std_logic;
        o2: out std_logic;
        o4: out std_logic
        );
      end entity N2;
```



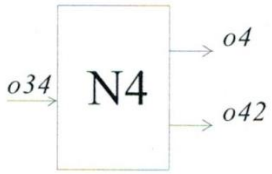
b)

```
library ieee;
use ieee.std_logic_1164.all;
entity N3 is port (
        i2: in std_logic;
        i23: in std_logic;
        o34: out std_logic
        );
      end entity N3;
```



c)

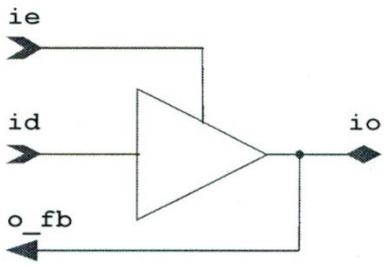
```
library ieee;
use ieee.std_logic_1164.all;
entity N4 is port (
        o34: in std_logic;
        o4: out std_logic;
        o42: out std_logic
        );
end entity N4;
```



d)

Sl. 2.7.2 Grafičke interpretacija entiteta a) N1, b) N2, c) N3 i d) N4

Ponašanje bidirekcionog bafera sa Sl. 2.7.3 opisano je Tabelom T. 2.7.1, u kojoj Z* označava da vrednost signala zavisi od pobude, jer se, tada, `io` ponaša kao ulazni port.



Sl. 2.7.3 Bidirekcioni bafer

Tabela

T. 2.7.1 Tablica istinitosti kojom se opisuje ponašanje bidirekcinog bafera

id	ie	io	o_fb
0	1	0	0
1	1	1	1
X	0	Z*	Z*

Na bidirekcionom izlazu pojaviće se ulazni signal `io`, ukoliko je enable ulaz `ie` u stanju logičke jedinice. Kada je `ie=0`, izlaz bafera je u stanju visoke impedanse, a stanje na portu `io` definiše pobuda. Za bidirekcioni port `io`, vezan je izlazni port `o_fb`. Na ovaj port dolaze signali koji se javljaju na portu `io`, nezavisno od toga sa koje strane signal dolazi: iz bafera ili spolja. Ovaj signal se koristi za vraćanje na ulaz tako da se naziva povratni ili *feed-back* signal.

Definicija para entitet/arhitektura bidirekcionog bafera sa Sl. 2.7.3 data je u primeru P. 2.7.2.

P. 2.7.2

```
library ieee;
use IEEE.STD_LOGIC_1164.all;
entity BIDBUFF is port
 (
  id: in std_logic;
  ie: in std_logic;
  o_fb: out std_logic;
  io: inout std_logic
  );
end entity BIDBUFF;

architecture rtl of BIDBUFF is
begin
 io <= id when ie = '1' else 'Z';
 o_fb <= io;
end architecture rtl;
```