DISSENY DIGITAL BÀSIC 2020-2021

PRÀCTICA 2: Implementació estructural d'un sistema digital (data limit 13 de Novembre)

L'objectiu d'aquesta segona pràctica és realitzar la implementació de funcions lògiques utilitzant la filosofia de disseny estructural. La metodologia d'aquest disseny consisteix en construir un sistema digital més o menys complex a partir de components més senzills que realitzen funcions més simples. Com a exemple d'aquesta metodologia tenim la realització d'una funció lògica implementada a dos nivells utilitzant components més petits, com són les portes lògiques. Una vegada s'han construït aquests blocs funcionals més simples s'han d'incloure en un sistema superior i connectar-los de forma adequada per que realitzin la funció desitjada.

Fins ara el disseny que hem realitzat s'ha basat en una arquitectura algebraica o lògica. Ara el que volem fer és mostrar que el valor d'una funció lògica, **f**, es pot obtenir connectant sistemes digitals donats per les seves entitats i arquitectures prèviament implementades. Com es mostra a la figura 1, per obtenir la funció **f=/a·b + a·/c** (aquí la barra inclinada la utilitzem per indicar el complementari de **a** o **c**, respectivament) cal que utilitzem un bloc que realitzi la funció suma lògica (OR2) i connectar a les seves entrades les sortides de dos blocs, cadascun dels quals realitzen la funció producte (AND2), alguna entrada dels quals, a la seva vegada, està connectada al ports externs (**a**, **b** i **c**) mitjançant inversors.

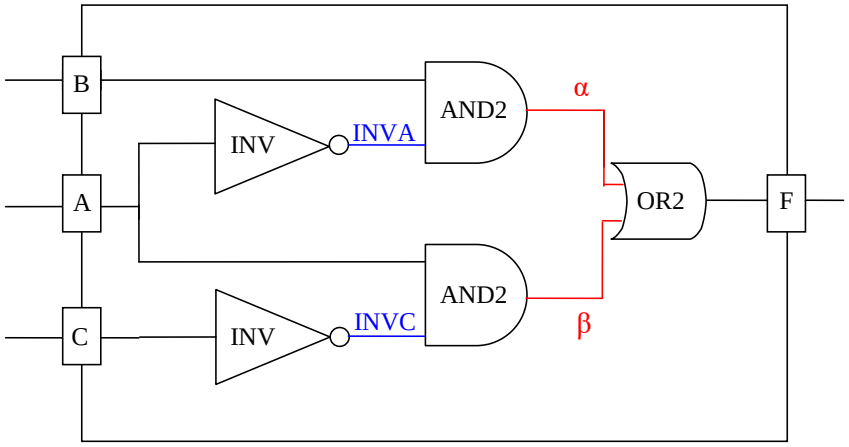

Figura 1. Realització de la funció **f** de forma jeràrquica (mètode estructural)

La porta AND2 de la part superior del dibuix té com a entrades els senyals del port **b** i el senyal intern **inva**. *És un senyal intern per què, a diferència del tipus de senyals que hem tractat fins ara, no és ni un senyal d'entrada ni un senyal de sortida (externs) i en el llenguatge VHDL cal que introduïm aquest concepte de senyal intern per tal de poder connectar elements internament.* Aquest senyal, a la seva vegada, és la sortida de la funció inversor, que té com a entrada el port **a**. La sortida d'aquesta porta AND2, que torna a ser un senyal intern α, és una de les entrades de la porta lògica OR2.

La porta AND2 inferior està connectada d'una forma similar a la que acabem de descriure per a la porta AND2 superior. Un dels seus dos terminals d'entrada està connectat al port d'entrada **a** i el segon, al senyal intern **invc**. Veiem, per analogia amb el senyal **inva**, que, efectivament, el senyal **invc** és un senyal intern i és la sortida d'un component inversor. Finalment la sortida de la porta AND2 inferior torna a ser un senyal intern, β, que és, a la vegada, una de les entrada de la porta OR2 que hi ha al final. **Noteu que la sortida d'aquesta darrera porta OR2 és també la sortida del circuit (no és un senyal intern).**

A continuació es presenta un codi estructural per fer aquesta funció:

```
-- Definim primer l'entitat i arquitectura de la porta OR de dues entrades, com s'ha fet a la pràctica anterior.
ENTITY or2 is
PORT (a,b: IN BIT;
        z: OUT BIT);
END or2;

ARCHITECTURE logicaretard OF or2 IS
BEGIN
z <= a OR b AFTER 3 ns;
-- Aquí introduïm un retard en la realització de la funció lògica des de que es presenten els valors de les entrades.
END logicaretard;

-- Definim l'entitat i arquitectura de la porta AND de dues entrades (també feta a la pràctica anterior).
ENTITY and2 IS
PORT (a,b: IN BIT;
        z: OUT BIT);
END and2;

ARCHITECTURE logicaretard OF and2 IS
BEGIN
z <= a AND b AFTER 3 ns;
END logicaretard;

-- Definim l'entitat i arquitectura de la porta inversora, feta a la pràctica 1.
ENTITY inv IS
PORT (a: IN BIT;
        z: OUT BIT);
END inv;

ARCHITECTURE logicaretard OF inv IS
BEGIN
z <= NOT a AFTER 3 ns;
END logicaretard;


-- Aquí comença la definició de la funció lògica que volem implementar. Com qualsevol
-- funció, cal definir entitat i arquitectura. La entitat segueix el patró habitual:
ENTITY Funcio_logica is
PORT ( a,b,c: IN BIT;
        f: OUT BIT);
END Funcio_logica;

-- Aquí definim l'arquitectura logica de la funció que volem implementar, que no és
-- l'objectiu d'aquesta pràctica, però que ens servirà per poder comprovar el funcionament
-- correcte del nostre codi. Aquí l'escrivim sense retard.
ARCHITECTURE logica OF funcio_logica IS
BEGIN
f <= ((NOT a) AND b) OR ((NOT c) AND a);
END logica;

-- Ara comença la definició de l'arquitectura estructural de l'entitat funcio_logica,
-- que és l'objectiu de la pràctica d'avui.

ARCHITECTURE estructural OF funcio_logica IS
-- Primer, a la part declaratòria de l'arquitectura, introduïm tots els components que volem fer servir.
-- Això es fa d'una forma anàloga a com ho hem fet amb l'arquitectura del banc de proves en les
-- pràctiques anteriors. En aquest exemple seran les portes or2, and2 e inv. Només cal que
-- definim UN SOL COP cadascun dels components que farem servir, encara que els fem servir diversos cops.
-- Aquesta situació ja s'ha donat a la part puntuable de la pràctica 2.
COMPONENT portaand2 IS
-- Els noms dels components són arbitraris i no tenen per què coincidir amb el nom de l'entitat.
-- Els noms dels ports del component han de coincidir EXACTAMENT amb els de l'entitat a la qual fan referència.
```

```
-- Per tant cal que coneguem exactament els noms d'aquestes entitats.
PORT(a,b: IN BIT;
        z: OUT BIT);
END COMPONENT;

COMPONENT portaor2 IS
PORT(a,b: IN BIT;
        z: OUT BIT);
END COMPONENT;

COMPONENT portainv IS
PORT(a: IN BIT;
        z: OUT BIT);
END COMPONENT;
-- Un cop introduïts els tipus de components que utilitzarem (en aquest exemple, les portes lògiques), caldrà afegir
-- SENYALS INTERNS per tal de poder fer les connexions entre els diferents components. En el nostre cas en
-- són quatre: inva, invc, alpha i beta. Fan la funció d'entrades i/o sortides dels components.

SIGNAL inva, invc, alpha, beta: BIT;
-- Definim ara els diferents dispositius que utilitzarem per implementar la funció. De la figura 1 deduïm que ens calen
-- cinc dispositius: dues portes and2, una porta or2 i dos inversors. Per tant, haurem d'escriure 5 DUTs.
FOR DUT1: portainv USE ENTITY WORK.inv(logicaretard);
FOR DUT2: portainv USE ENTITY WORK.inv(logicaretard);
FOR DUT3: portaand2 USE ENTITY WORK.and2(logicaretard);
FOR DUT4: portaand2 USE ENTITY WORK.and2(logicaretard);
FOR DUT5: portaor2 USE ENTITY WORK.or2(logicaretard);

-- Aquí s'acaba la part declarativa de l'arquitectura. Ara passarem al cos de l'arquitectura.
-- Un cop introduïts tots els dispositius i senyals, passem a realitzar les connexions i, d'aquesta forma, fer la definició
-- de la funció lògica en funció de les variables A, B i C. Això es fa d'una forma anàloga al que hem fet al banc de
-- proves de les pràctiques anteriors. Per cada dispositiu, mirem quins senyals s'han de posar en els seus terminals
-- per poder fer la funció demanada. Haurem de fer servir els senyals definits a l'entitat que volem implementar,
-- funcio_logica, i els senyals interns. Serà, doncs:
BEGIN
        DUT1: portainv PORT MAP(A,inva);
        DUT2: portainv PORT MAP(C,invc);
        DUT3: portaand2 PORT MAP(inva,B,alpha);
        DUT4: portaand2 PORT MAp(A,invc,beta);
        DUT5: portaor2 PORT MAP(alpha,beta,f);
END estructural;

-- Finalment, un cop definida l'entitat i arquitectures del circuit digital que volem implementar, cal que definim
-- l'entitat i l'arquitectura del banc de proves, és a dir, de l'entitat en què es proven que els circuits funcionen
-- correctament sota la presència de senyals externs donats, és a dir, en què es simulen les funcions.
ENTITY bancdeproves IS
END bancdeproves;


ARCHITECTURE test_de_proves OF bancdeproves IS
-- Ara introduïm el component que volem testejar, que és el propi circuit. Per a nosaltres aquest component
-- és el circuit digital de la figura 1, és a dir, té tres entrades i una sortida. El nom del component és el que
-- nosaltres vulguem.
-- Fixem-nos que no ens cal tornar a introduir els components que estan dins del circuit a simular, ja que aquests
-- formen part de l'arquitectura de la funció lògica que simulem.
COMPONENT bloc_que_simulem IS
PORT(A,B,C: IN BIT;
        f: OUT BIT);
END COMPONENT;
--Definim quins són els senyals externs que apliquem o que obtenim com a resultat de la/les funció/ons, que
-- anomenarem senyalA, senyalB, senyalC, sortida_f_logica i sortida_f_estructural. De les dues sortides, una
-- correspondrà a la realització funcional i l'altre, a l'estructural. D'aquesta forma podrem veure, simultàniament,
-- les dues realitzacions.
SIGNAL senyalA,senyalB,senyalC,sortida_f_logica,sortida_f_estructural: BIT;
FOR DUT1: bloc_que_simulem USE ENTITY WORK.funcio_logica(logica);
```

```
FOR DUT2: bloc_que_simulem USE ENTITY WORK.funcio_logica(estructural);
-- Aquí s'acaba la part declarativa de l'arquitectura. Ara passarem al cos de l'arquitectura.

BEGIN
-- Associem les entrades i sortides externes amb els ports que té el component que volem simular i que, tal com ja hem
-- vist, no cal que tinguin el mateix nom. El que si que és important és que l'ordre dels senyals externs sigui el mateix
-- que l'ordre de les variables del component i que no intercanviem entrades i sortides ni que posem més senyals externs
-- al dispositiu que els que estan definits (o menys senyals).
        DUT1: bloc_que_simulem PORT MAP(senyalA, senyalB, senyalC, sortida_f_logica);
        DUT2: bloc_que_simulem PORT MAP(senyalA, senyalB, senyalC, sortida_f_estructural);
                PROCESS (senyalA, senyalB, senyalC)
                        BEGIN
                        senyalA <= NOT senyalA AFTER 200 ns;
                        senyalB <= NOT senyalB AFTER 100 ns;
                        senyalC <= NOT senyalC AFTER 50 ns;
                END PROCESS;
-- Utilitzar aquesta forma de variar els senyals d'entrada, a més de ser més compacte, té l'avantatge afegit, si
-- s'escriu correctament, que ens permet de fer variar els senyals d'entrada recorrent tots els valors possibles.
-- Ens permet, doncs, de fer l'equivalent a la taula de veritat, presentada a la teoria de l'assignatura.
END test_de_proves;
```

A les dues primeres pràctiques es van mostrar dues formes de fer variar els senyals, la que s'indica aquí dalt i la consistent en la instrucció "WAIT FOR xx ns". Es poden combinar totes dues formes de fer variar els senyals per visualitzar la resposta a senyals periòdics i a senyals no periòdics. Per a això, cal fer dos processos, un per cada tipus de senyal. A continuació es mostra com es podrien substituir les línies en verd del codi que acabem d'escriure:

```
                PROCESS (senyalB, senyalC)
                    BEGIN
                    senyalB <= NOT senyalB AFTER 100 ns;
                    senyalC <= NOT senyalC AFTER 50 ns;
                END PROCESS;
-- Aquesta ha estat la variació dels senyals periòdics.
                PROCESS
                    BEGIN
                    senyalA <= '0'; WAIT FOR 200 ns;
                    senyalA <= '1'; WAIT FOR 200 ns;
                    senyalA <= '0'; WAIT FOR 200 ns;
                    senyalA <= '1'; WAIT FOR 200 ns;
                    senyalA <= '0'; WAIT FOR 200 ns;
                END PROCESS;
-- Aquí hem fet la variació arbitrària (en aquest cas, per poder comparar els resultats, s'ha fet periòdica, també).
```

<u>Treball a desenvolupar de forma autònoma</u>:
(a entregar a través del campus virtual abans del divendres 13/11/20 a les 23:59)

1. Analitzeu el codi sobre l'arquitectura estructural que es subministra més amunt, intentant entendre el seu funcionament. Comproveu que, efectivament, funciona correctament i que les dues arquitectures escrites descriuen correctament la mateixa funció, una amb retard i l'altre, sense. Raoneu per què es produeix un "rebot" en el senyal de sortida de la arquitectura estructural entre els instants 206 i 209 ns. Afegeix un comentari amb la resposta a l'inici del codi del primer fitxer que heu d'entregar a través del campus.

2. Implementeu la següent funció **funcio** de 4 variables d'entrada, **a**, **b**, **c** i **d**, i una sortida **f** tal com està escrita (és a dir, sense simplificar). La implementarem amb arquitectures **logica**, **logicaretard** i **estructural**.

$$f(a,b,c,d) = a \cdot c \cdot (a \text{ XOR } d) + (/b \cdot c)$$

   a) Per tal d'implementar l'arquitectura estructural, primer caldrà que recupereu les portes lògiques de la pràctica anterior, amb arquitectures '**logica**' i '**logicaretard**'. En aquesta darrera arquitectura, imposeu un retard de 3 ns (fes-ho en un fitxer apart).
   b) Implementeu ara l'entitat **funcio** amb la seva arquitectura **logica** i **logicaretard** a partir de l'expressió de dalt, SENSE SIMPLIFICAR-LA. En l'arquitectura **logicaretard** imposeu un retard de 3 ns. Podeu fer servir els noms de **bdp** i **test** per l'entitat i l'arquitectura del banc de proves, respectivament. Feu que els senyals variïn cada 50 ns (tal i com es fa a l'exemple de codi).
   c) Escriviu l'arquitectura **estructural** corresponent a la **funcio** que heu implementat. Utilitzeu primer les arquitectures **logica** de les portes. Afegiu en el mateix banc de proves la **funcio** amb la seva arquitectura **estructural** i comproveu el resultat. Noteu que la sortida de la **funcio** ha de ser un senyal diferent per a cada arquitectura (**logica**, **logicaretard**, **estructural**).
   Per tal de comprovar el funcionament correcte, podeu genereu-vos en un paper la taula de veritat corresponent a la funció. Després, comproveu, que sota les 16 combinacions de les variables d'entrada, la sortida de la simulació coincideix amb la taula de veritat.
   d) Ara considereu l'arquitectura retardada '**logicaretard**' de les portes en una nova arquitectura '**estructural_R**'. Afegiu ara en el mateix banc de proves la **funcio** amb aquesta darrera arquitectura **estructural_R** i comproveu el resultat (heu d'afegir la sortida en el banc de proves corresponent a la **funcio** amb aquesta arquitectura **estructural_R**). Construïu el corresponent banc de proves. Comproveu que hi ha diferències amb l'apartat b) que no són, exclusivament, un endarreriment de tota la funció de sortida. Descriviu i justifiqueu aquestes diferències, afegint un comentari en el codi, on indiqueu explícitament a quins instants de temps apareixen.
   e) A partir del banc de proves anterior, modifiqueu-lo de forma que els senyals variïn cada 5 ns (després torneu a deixar-ho amb la variació de 50 ns). Compareu aquest comportament amb el que s'esperaria per la funció lògica que realitza el circuit, que seria el donat a l'apartat b). Per què són diferents? Penseu-hi i justifiqueu-nos-ho (afegint un altre comentari al final del codi).

Haureu de pujar 2 fitxers, SENSE COMPRIMIR, que continguin les següents informacions:

1) Un fitxer amb les entitats i arquitectures **logica** i **logicaretard** de les portes lògiques que heu implementat a la pràctica 1. És a dir, les portes **inversor**, **and2**, **and3**, **and4**, **or2**, **or3**, **or4** i **xor2**. El nom del fitxer serà **P2a_Cognom1_Cognom2_Nom_portes.vhd**. Els codis d'aquest fitxer, amb EXACTAMENT aquests noms d'entitats, arquitectures i terminals, seran necessaris per a les properes pràctiques. Per tant, no utilitzeu noms diferents.

2) Un segon fitxer (**P2a_Cognom1_Cognom2_Nom_funcio.vhd**) que implementi la funció que s'ha descrit a l'apartat 2 (nom de l'entitat **funcio**) amb les seves arquitectures **logica, logicaretard, estructural** i **estructural_R**. Afegiu l'entitat **bdp** amb l'arquitectura **test**, amb els senyals d'entrada **ent3**, **ent2**, **ent1** i **ent0** i les sortides, **sort_logica, sort_logica_R, sort_estructural_R** i **sort_estructural**. Responeu les dues preguntes que es plantegen als apartats 1), 2d) i 2e), posant-les com a comentaris en el codi.

Recordeu que aquestes entitats, arquitectures i bancs de proves es faran servir en properes pràctiques

Aquest és el treball que haureu de pujar a través del campus virtual abans de les 23:59 del divendres previ a la sessió de pràctiques (13 de Novembre). Un cop passat aquest temps ja no serà possible pujar els fitxers.
NO ENS ENVIEU ELS CODIS PER CORREU ELECTRÒNIC.