DISSENY DIGITAL BÀSIC 2020-2021

PRÀCTICA 3: Implementació de circuits combinacionals i seqüencials amb arquitectura "ifthen"
(24 de Novembre)

L'objectiu d'aquesta tercera pràctica és doble: per un costat realitzarem la implementació de funcions lògiques fent servir una nova arquitectura que anomenarem ***ifthen***, i, per l'altra, utilitzarem aquesta nova arquitectura per implementar sistemes seqüencials bàsics del tipus ***flip-flops***. De fet, fent servir aquests ***flip-flops*** i la filosofia de disseny ***estructural*** poden implementar els exercicis dels temes 4 i 5.

La metodologia del disseny '***ifthen***' consisteix en construir clàusules del tipus 'si passa això .... amb la variable de entrada, la sortida serà ...'. Es basa, doncs, en la valoració de les diferents situacions que es podem donar en el sistema electrònic i, per tant, és equivalent a escriure la taula de veritat. Això ens servirà, entre altres coses, per poder entendre el comportament d'aquest nou tipus de dispositius i complementar la teoria, comprovant-la.

L'arquitectura '***ifthen***' presenta un disseny molt més simple i directe que l'arquitectura lògica, on podem expressar de forma directa la taula de la veritat de la funció que realitza. Aquí us exposem el codi per a la implementació de la funció **f = /a·b + a·/c**, que és la mateixa funció que us hem presentat com a exemple a l'arquitectura estructural de la pràctica 2.

```
-- Defineixo l'entitat
--ENTITY funcio_logica is
--PORT ( a,b,c: IN BIT;
--     f: OUT BIT);
--END Funcio_logica;

-- Ara definirem la nova arquitectura, tipus 'ifthen'
ARCHITECTURE ifthen OF funcio_logica IS
BEGIN
       PROCESS (a, b, c)
-- Analitzem els canvis a les variables d'entrada a, b, c.
       BEGIN
-- Iniciem el cos de l'arquitectura.
              IF a='0' AND b='0' AND c='0' THEN
                     f<= '0' AFTER 3 ns;
              ELSIF a='0' AND b='0' AND c='1' THEN
                     f<= '0' AFTER 3 ns;
              ELSIF a='0' AND b='1' AND c='0' THEN
                     f<= '1' AFTER 3 ns;
              ELSIF a='0' AND b='1' AND c='1' THEN
                     f<= '1' AFTER 3 ns;
              ELSIF a='1' AND b='0' AND c='0' THEN
                     f<= '1' AFTER 3 ns;
              ELSIF a='1' AND b='0' AND c='1' THEN
                     f<= '0' AFTER 3 ns;
              ELSIF a='1' AND b='1' AND c='0' THEN
                     f<= '1' AFTER 3 ns;
              ELSIF a='1' AND b='1' AND c='1' THEN
                     f<= '0' AFTER 3 ns;
              END IF;
       END PROCESS;
END ifthen;

-- Realitzem el banc de proves
-- Aquest banc de proves també contempla les arquitectures logica i estructural del banc
de proves anterior (de fet, l'hem definit com a una entitat diferent).
ENTITY banc_de_proves IS
END banc_de_proves;

ARCHITECTURE test OF banc_de_proves IS

COMPONENT bloc_que_simulem IS
PORT ( a,b,c: IN BIT;
```

```
      f: OUT BIT);
-- Recordeu que els noms de les variables del component han de ser
-- exactament les mateixes que a l'entitat a que es refereixen.
END COMPONENT;

SIGNAL senyalA,senyalB,senyalC: BIT;
SIGNAL sortida_f_logica,sortida_f_estructural, sortida_f_ifthen: BIT;

FOR DUT1: bloc_que_simulem USE ENTITY WORK.Funcio_logica(logica);
FOR DUT2: bloc_que_simulem USE ENTITY WORK.Funcio_logica(estructural);
--recordeu afegir el codi de l'arquitectura estructural que ja vàrem presentar a la
pràctica 2
FOR DUT3: bloc_que_simulem USE ENTITY WORK.Funcio_logica(ifthen);

BEGIN

DUT1: bloc_que_simulem PORT MAP(senyalA, senyalB, senyalC, sortida_f_logica);
DUT2: bloc_que_simulem PORT MAP(senyalA, senyalB, senyalC, sortida_f_estructural);
DUT3: bloc_que_simulem PORT MAP(senyalA, senyalB, senyalC, sortida_f_ifthen);


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;

END test;
```

Aquesta nova arquitectura és molt útil per definir sistemes seqüencials, on la seva sortida depèn de l'estat anterior, tal com són els biestables. Com a exemple d'aquesta metodologia implementarem primer un '**FF_D**' per flanc de baixada amb '**Preset**' i '**Clear**', i després un '**Latch JK**' amb '**Preset**' i '**Clear**':

```
ENTITY D_Bajada_PreClr IS
PORT(D,Clk,Pre,Clr: IN BIT; Q,NO_Q: OUT BIT);
END D_Bajada_PreClr;

ARCHITECTURE ifthen OF D_Bajada_PreClr IS
SIGNAL qint: BIT;
-- Aquesta és la forma de definir sortides que es realimenten
-- a l'entrada. Recordem que en VHDL si una variable és de sortida no es
-- pot utilitzar com a entrada de nou. Per tant definim una variable interna
-- que després assignarem a la variable de sortida.
BEGIN

PROCESS (D,Clk,Pre,Clr)
BEGIN
IF Clr='0' THEN qint<='0' AFTER 2 ns;
-- Aquí la funció Clear és Active-low, és a dir, quan Clr=0 la sortida es posa
-- a 0 de forma asíncrona; quan Clr=1 el circuit farà alguna altra funció.
      ELSIF Pre='0' THEN qint<='1' AFTER 2 ns;
-- Hem imposat nosaltres que si es posen, simultàniament, Clear i Preset
-- a 0, el Clear és qui té prioritat. Per això aquí la condició sobre el Preset
-- s'analitza només quan el Clear està posat a 1.
      ELSIF Clk'EVENT AND Clk='0' THEN
-- Clk'EVENT és una instrucció que dóna una sortida veritat, és a dir, un 1
-- quan es produeix un canvi en el valor de la variable Clk.
-- La instrucció, tal com està aquí, indica que si es produeix un canvi en
-- el senyal Clk i, a més, el valor posterior és 0 (baixada), llavors …
      qint <= D AFTER 2 ns;

END IF;
END PROCESS;
Q<=qint; NO_Q<=NOT qint;
-- Aquí es on es fa l'assignació de les variables
-- internes a les variables de sortida
END ifthen;
```

```
ENTITY JK_Latch_PreClr IS
PORT(J,K,Clk,Pre,Clr: IN BIT; Q,NO_Q: OUT BIT);
END JK_Latch_PreClr;

ARCHITECTURE ifthen OF JK_Latch_PreClr IS
SIGNAL qint: BIT;
BEGIN
PROCESS (J,K,Clk,Pre,Clr,qint)
BEGIN
IF Clr='0' THEN qint<='0' AFTER 2 ns;
ELSE
IF Pre='0' THEN qint<='1' AFTER 2 ns;
ELSE
             IF Clk='1' THEN
                    IF J='0' AND K='0' THEN qint<=qint AFTER 2 ns;
                    ELSIF J='0' AND K='1' THEN qint<='0' AFTER 2 ns;
                    ELSIF J='1' AND K='0' THEN qint<='1' AFTER 2 ns;
                    ELSIF J='1' AND K='1' THEN qint<= NOT qint AFTER 2 ns;
                    END IF;

             END IF;
       END IF;
END IF;

END PROCESS;
Q<=qint; NO_Q<=NOT qint;
END ifthen;
```

Podem fer el corresponent banc de proves per tal de testar aquestes dues entitats. A continuació teniu l'exemple d'un banc de proves:

```
-- Banc de Proves d'ambos
ENTITY banc_proves IS
END banc_proves;

ARCHITECTURE test OF banc_proves IS
COMPONENT mi_D_Bajada_PreClr IS
PORT(D,Clk,Pre,Clr: IN BIT; Q,NO_Q: OUT BIT);
END COMPONENT;
COMPONENT mi_JK_Latch_PreClr IS
PORT(J,K,Clk,Pre,Clr: IN BIT; Q,NO_Q: OUT BIT);
END COMPONENT;
SIGNAL ent1,ent2,clock,preset,clear,Dsort_Q,Dsort_noQ,JKsort_Q,JKsort_noQ: BIT;
FOR DUT1: mi_D_Bajada_PreClr USE ENTITY WORK.D_Bajada_PreClr(ifthen);
FOR DUT2: mi_JK_Latch_PreClr USE ENTITY WORK.JK_Latch_PreClr(ifthen);
BEGIN
DUT1: mi_D_Bajada_PreClr PORT MAP (ent1,clock,preset,clear,Dsort_Q,Dsort_noQ);
DUT2: mi_JK_Latch_PreClr PORT MAP (ent1,ent2,clock,preset,clear,JKsort_Q,JKsort_noQ);
ent1 <= NOT ent1 AFTER 800 ns;
ent2 <= NOT ent2 AFTER 400 ns;
clock <= NOT clock AFTER 500 ns;
preset <= '0', '1' AFTER 600 ns;
clear <= '1','0' AFTER 200 ns, '1' AFTER 400 ns;
-- simuleu fins a 15000 ns
END test;
```

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

1. Comproveu el funcionament dels FFs descrits anteriorment i compareu-los amb la teoria.
2. Definiu les entitats amb les seves corresponents arquitectures de tipus '**ifthen**' (amb retard), actius per nivell alt o per flanc de pujada, dels següents sistemes seqüencials:
   a) Latch D i Flip-Flop D, tots dos amb els següents ports d'entrada i sortida:
      ```
      IN BIT: D,Clk,Pre,Clr
      OUT BIT: Q,NO_Q
      ```
   b) Latch J-K i Flip-Flop J-K, tots dos amb els següents ports d'entrada i sortida:
      ```
      IN BIT: J,K,Clk,Pre,Clr
      OUT BIT: Q,NO_Q
      ```
   c) Latch T i Flip-Flop T, tots dos amb els següents ports d'entrada i sortida:
      ```
      IN BIT: T,Clk,Pre,Clr
      OUT BIT: Q,NO_Q
      ```
3. Escriviu un banc de proves ***bdp_biestables*** amb la seva arquitectura ***test_biestables*** per tal de comprovar el funcionament de tots aquests biestables a la vegada. Amb l'ajuda del cursor, comproveu que el comportament de cada dispositiu és el que s'espera. Veus algun comportament estrany? On? Afegeix un comentari al final del codi raonant la teva resposta.
4. Quina diferència hi ha entre un Latch i un FF? (ja sigui del tipus D, JK o T). Afegeix un altre comentari amb el raonament de la teva resposta.

Recordeu de fer córrer la simulació un temps prou llarg per tal que es puguin veure totes les combinacions possibles de les entrades (indiqueu-lo també al codi amb un comentari, dins del banc de proves).

Haureu de pujar 1 fitxer, SENSE COMPRIMIR, que continguin les següents informacions:

1) Pugeu un fitxer VHD amb les entitats i arquitectures dels biestables i el seu banc de proves. El fitxer es dirà **P3_Cognom1_Cognom2_Nom.vhd**.

<u>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 (20 de Novembre). Un cop passat aquest temps ja no serà possible pujar els fitxers.</u>
<u>NO ENS ENVIEU ELS CODIS PER CORREU ELECTRÒNIC.</u>