ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ CAD ΓΙΑ ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ – ΗΡΥ 608 / 419
Εαρινό Εξάμηνο 2023

Προθεσμία: Δευτέρα 15/5/23 έως τα μεσάνυχτα, υποβολή στο eClass

ΑΣΚΗΣΗ 5η (Έκδοση 1.0 της εκφώνησης, μπορεί να υπάρχουν διορθώσεις)

SYNTHESIS: ΟΛΟΚΛΗΡΩΣΗ ΚΥΚΛΟΥ ΜΕ ΚΥΚΛΩΜΑΤΑ ΠΟΥ ΕΧΟΥΝ ΠΕΡΙΣΣΟΤΕΡΑ ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΑΠΟ FULL ADDER

1.0 ΕΙΣΑΓΩΓΗ

Όπως έχει εξηγηθεί στο μάθημα, η άσκηση αυτή είναι για διευθέτηση μικροθεμάτων που έχουν μείνει εκκρεμή, καθώς και η ολοκλήρωση ενός συστήματος που έχει περισσότερα δομικά στοιχεία από full adder, και δημιουργία τελικού netlist πυλών. Πιο συγκεκριμένα, στην παρούσα άσκηση θα κάνουμε τα εξής:

- Ονομασία των υποσυστημάτων που χρησιμοποιούμε (στην Άσκηση 4 ο full adder εννοείται, εδώ θα τον ονοματίσουμε με επέκταση του format αρχείου εισόδου).
- Η τελική έξοδος θα είναι netlist πυλών (κανονικά εδώ πρέπει να χρειάζεται μηδενικός νέος κώδικας γιατί η έξοδος της άσκησης 4 είναι απόλυτα συμβατή με την είσοδο της άσκησης 3).
- Δημιουργία κυκλωμάτων που είναι κάπως πιο σύνθετα από full adder. Για να κρατήσουμε απλό τον κώδικα, απλά αντί για n-bit full adder θα κάνετε n-bit adder/subtractor (από πλευράς κυκλωμάτων η αλλαγή είναι πάρα πολύ απλή, και έχουμε ένα επί πλέον σήμα το ADD'_SUB). Δεν απαιτείται καμμία αλλαγή στις βιβλιοθήκες πυλών, καθότι η μοναδική πρόσθεση σε επίπεδο λογικού κυκλώματος είναι μία XOR σε κάθε Bit. Πρέπει να ορίσετε ένα νέο δομικό στοιχείο μαζί με το netlist του, π.χ. FULL_ADDER_SUBTRACTOR, με εισόδους A,B,CIN,ADD'_SUB, και εξόδους S,COUT. Σε κάθε περίπτωση πρέπει να μπορείτε να περιγράψετε ένα νέο υποσύστημα, από adder που ήταν στην άσκηση 4 σε adder (για να μην χάσουμε λειτουργικότητα) κα adder_subtractor (το νέο υποσύστημα). ΠΡΟΣΟΧΗ: Κρατάμε τον FULL_ADDER και στο τέλος μπορούμε να δημιουργούμε n-bit adder ή n-bit adder/subtractor.

2.0 SYNTHESIS – ΣΥΝΘΕΣΗ ΑΛΛΑΓΕΣ ΣΤΗΝ ΑΣΚΗΣΗ 4

Η μόνη ουσιαστική αλλαγή στην άσκηση 4 είναι η προσθηκη **LIB** που μας λέει τι να «τραβήξουμε» από την βιβλιοθήκη υποσυστημάτων, αφού πλέον έχουμε δύο υποσυστήματα (τρία με τον πολυπλέκτη, που όμως δεν χρησιμοποιούμε).

```
ENTITY adder IS
LIB {COMPONENT | SUBSYSTEM}
VAR N = {1 UP to 8}  %% Number of Bits in the Adder
PORT ( IN A     : {A_MSB,…,A_LSB}        %% A
       IN B     : {B_MSB,…,B_LSB}        %% B
       IN Cin   : {Cin}                  %% Cin
       OUT S    : {S_MSB;…;S_LSB}        %% Sum
       OUT Cout : {Cout}                 %% Cout
END adder
```

Όλες οι μεταβλητές είναι έως δέκα χαρακτήρες, όχι απαραίτητα ίδιο μέγεθος σε χαρακτήρες η καθεμία, και το N μπορεί να πάρει τιμές από 1 έως 4 (αντί για 8, ώστε να είναι πιο εύκολη η αποσφαλμάτωση). Αν ο χαρακτήρας «'» είναι πρόβλημα μπορείτε να τον αποφύγετε βάζοντας ένα άλλο χαρακτήρα για την αντιστροφή, π.χ. ADDN_SUB.

Ένα (πραγματικό) ενδεικτικό αρχείο εισόδου της αλλαγμένης Άσκησης 4, με το αρχικό υποσύστημα FULL_ADDER στην βιβλιοθήκη είναι:

```
ENTITY adder IS
VAR N = 3  %% Number of Bits in the Adder
LIB FULL_ADDER
PORT ( IN FULL_ADDER A     : SO , YOU , THINK       %% A, MSB TO LSB
       IN FULL_ADDER B     : CAN , TELL , HEAVEN    %% B, MSB TO LSB
       IN FULL_ADDER Cin   : {FROM}                 %% Cin
       OUT FULL_ADDER S    : {HELL , BLUE , SKIES } %% Sum, MSB TO LSB
       OUT FULL_ADDER Cout : {PAIN}                 %% Cout
END adder
```

Αυτή είναι η αλλαγή στον κώδικα της άσκησης 4. Για το παραπάνω αρχείο εισόδου, το αντίστοιχο αρχείο εξόδου θα είναι το ίδιο όπως στην άσκηση 4 .Αν όμως βάλουμε και ένα νέο υποσύστημα (όπως περιγράφηκε πάνω, τότε μπορούμε να έχουμε το εξής, πιο ρεαλιστικό σενάριο για τον ADDER/SUBTRACTOR, όπου (προσοχή) έχουμε στο μεν υποσύστημα της βιβλιοθήκης FULL_ADDER_SUBTRACTOR με σήμα εισόδου ADD'_SUB επί πλέον από τα υπόλοιπα, ενώ στην χρήση του (το έβαλα επίτηδες με διαφορετικό κάπως όνομα για να μην μπερδευόμαστε) έχουμε το σήμα ADD_SUB_MODE.

```
ENTITY adder_subtractor IS
VAR N = 3  %% Number of Bits in the Adder/Subtractor
LIB FULL_ADDER_SUBTRACTOR
PORT ( IN FULL_ADDER_SUBTRACTOR A     : A2, A2, A0      %% A, MSB TO LSB
       IN FULL_ADDER_SUBTRACTOR B     : B2, B1, B0      %% B, MSB TO LSB
       IN FULL_ADDER_SUBTRACTOR Cin   : ADD_SUB_MODE    %% Cin
       IN FULL_ADDER_SUBTRACTOR ADD'_SUB  : ADD_SUB_MODE %% Control Signal
       OUT FULL_ADDER_SUBTRACTOR S    : S2, S2, S0      %% Sum, MSB TO LSB
       OUT FULL_ADDER_SUBTRACTOR Cout : COUT            %% Cout
END adder_subtractor
```

Όπως και στην άσκηση 4 θεωρούμε ότι ξέρει το σύστημα πως να μεταφέρει τα κρατούμενα εξόδου μίας τάξης μεγέθους στα κρατούμενα εισόδου της επόμενης, και αυτό το κάνουμε για να αποφύγουμε το να πρέπει να γράψετε κώδικα που να παράγει κώδικα σαν κανονικός compiler. Επομένως κρατάμε το ότι ο κώδικάς σας «ξέρει» (και μάλιστα με τον ίδιο τρόπο) να συνδέει τα δύο διαφορετικά υποσυστήματα που έχουμε πλέον για να βγάλει ένα σύστημα n-bits και του προσθέσαμε μόνο το εξωτερικό σήμα που χρειάζεται για αφαίρεση και το να καλούμε από την βιβλιοθήκη όποιο υποσύστημα χρειαζόμαστε.

3.0 ΣΗΜΑΝΤΙΚΕΣ «ΛΕΠΤΟΜΕΡΕΙΕΣ»

Είναι σημαντικό τα τρία αρχεία (component library, subsystem library, αρχείο εισόδου) να τα διαβάζουμε μία φορά το καθένα. Είναι απόλυτα αποδεκτό το πρόγραμμά σας να είναι δύο ή τρία προγράμματα που εκτελούνται σε αλληλουχία, όπου το καθένα βγάζει αρχείο εισόδου που διαβάζει το επόμενο, μέχρ να βγει το τελικό Netlist πυλών. Δεν επιτρέπεται όμως να πρέπει να παρέμβει ο χρήστης (πέραν του ανδεχόμενου να καλεί την αλληλουχία των προγραμμάτων όπως θα έκανε ένα script).

Αν υπάρχουν λάθη ή ασάφειες παρακαλώ στείλετέ μου e-mail στο adollas@tuc.gr ώστε να το διευθετήσουμε.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!!!