ΗΡΥ203 Προχωρημένη Λογική Σχεδίαση
1η Εργαστηριακή Άσκηση

Ομάδα LAB20332002
Παντουράκης Μιχαήλ ΑΜ 2015030185
Τυχάλας Πέτρος ΑΜ 2015030169
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πολυτεχνείο Κρήτης

3 Μαρτίου 2017

1 Σκοπός Εργαστηριακής Άσκησης

Σημαντικός σκοπός του μαθήματος συνολικά αποτελεί η εκμάθηση του ιεραρχικού τρόπου σχεδίασης ψηφιακών συστημάτων. Στο πλαίσιο των εργαστηριακών ασκήσεων χρησιμοποιούνται τα προγραμματιζόμενα ολοκληρωμένα κυκλώματα FPGA, και πιο συγκεκριμένα το Basys 2 του κατασκευαστή Xilinx. Συνεπώς, η πρώτη στη σειρά εργαστηριακή άσκηση αποτελεί για τους φοιτητές μία εισαγωγή στη χρήση της γλώσσας περιγραφής υλικού VHDL, αλλά και του επίσημου εργαλείου σχεδίασης Xilinx ISE. Αυτό με τη σειρά του προσφέρει μία αυτοματοποιημένη πλατφόρμα σχεδιασμού συμπεριφορικού/δομικού μοντέλου κυκλωμάτων, προσομοίωσης της λειτουργίας τους και την περαιτέρω σύνθεση και υλοποίηση τους σε FPGA.

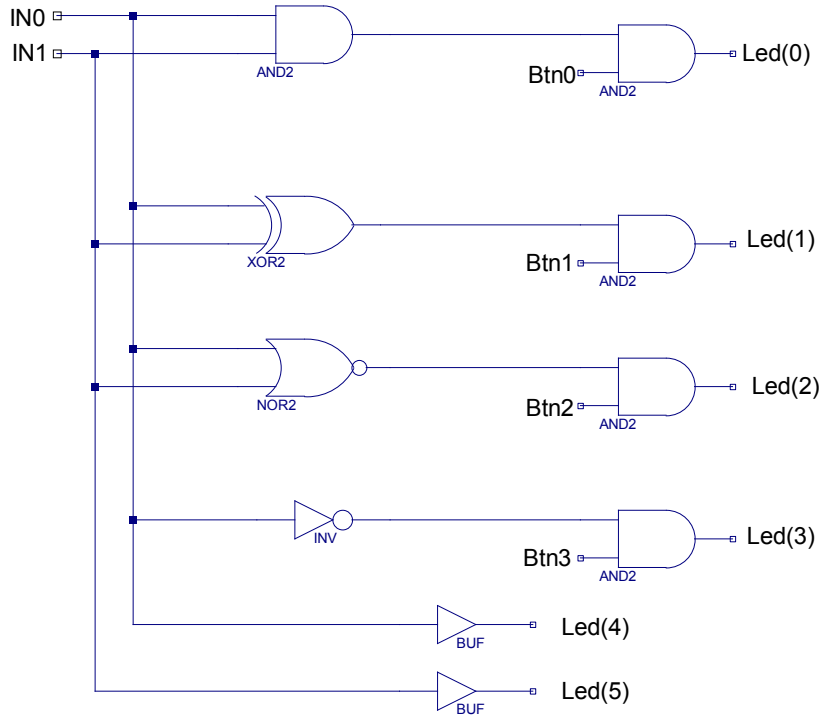
Στην παρούσα άσκηση λοιπόν χρησιμοποιούμε δύο πολύ απλά παραδείγματα κυκλωμάτων: 1) απλή υλοποίηση πυλών δύο εισόδων, 2) ιεραρχική υλοποίηση ενός πλήρους αθροιστή με χρήση ημιαθροιστών. Η ευκολία αυτών των δύο κυκλωμάτων επιτρέπει την άμεση εξοικείωση με τη γλώσσα και το εργαλείο σχεδίασης, με τη δομική σχεδιαστική ροή και φυσικά με το ίδιο το ολοκληρωμένο κύκλωμα.

2 Προεργασία-Περιγραφή

Το Σχήμα 1 απεικονίζει το πρώτο ζητούμενο κύκλωμα της άσκησης και τις εξισώσεις που το περιγράφουν. Οι πύλες του πρώτου επιπέδου αντιστοιχούν πλήρως στη ζητούμενη λειτουργία της εκφώνησης, ενώ οι πύλες εξόδου AND επιτρέπουν την εμφάνιση του αποτελέσματος των πρώτων πυλών μόνο με το πάτημα των αντίστοιχων Buttons. Επιπλέον, τα leds 4 έως 5 συνδέονται απευθείας με τις δύο εισόδους. Αντίθετα, τα leds 6 και 7, τα οποία δίνονται ως έξοδοι στον πίνακα της εκφώνησης, δεν συμμετέχουν σε κάποια λειτουργία και επιλέξαμε να τα κρατήσουμε σβηστά.

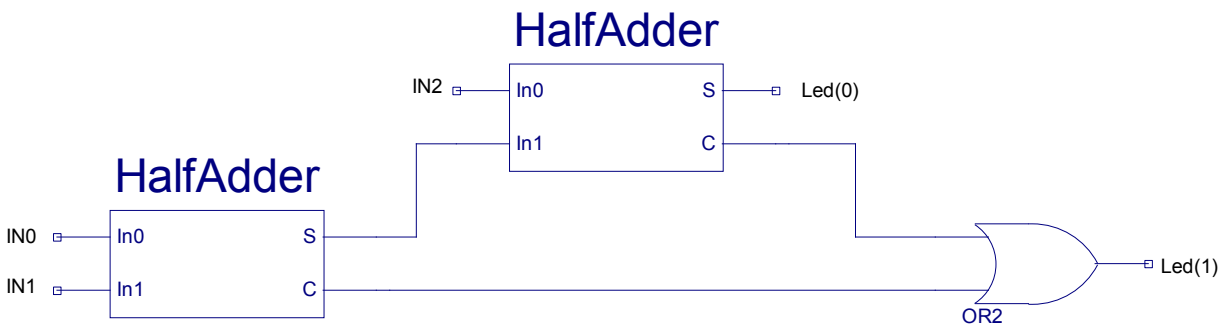
Στο Σχήμα 2 παρουσιάζουμε το block diagram και τις αντίστοιχες εξισώσεις της υλοποίησης ενός πλήρους αθροιστή (full adder) με τη χρήση ημιαθροιστών (half adders). Όπως ζητήθηκε από τις προδιαγραφές της άσκησης, ο πλήρης αθροιστής δέχεται τρεις 1-bit εισόδους (IN0...IN2), και παρέχει ως έξοδο 2 bit, δηλαδή το Sum και το Carry (στα Led(0) και Led(1) αντίστοιχα).

Για την καλύτερή εξάσκησή μας για τα επόμενα εργαστήρια, χρησιμοποιήσαμε δομικό τρόπο σχεδίασης (structural design) για την υλοποίηση και των δύο κυκλωμάτων. Συγκεκριμένα, για κάθε πύλη (AND, NOT, NOR, XOR, OR) δημιουργήσαμε ξεχωριστό module και entity, και ορίσαμε τη λειτουργία τους με behavioural architecture. Αυτές χρησίμευσαν ως δομικές μονάδες για την σύνθεση των ανώτερων ιεραρχικά μονάδων: Half Adder και Full Adder. Ως παράδειγμα αυτού του τρόπου σχεδίασης στο παράρτημα παραθέτουμε τους κώδικες (entity-architecture) για τον πλήρη αθροιστή.



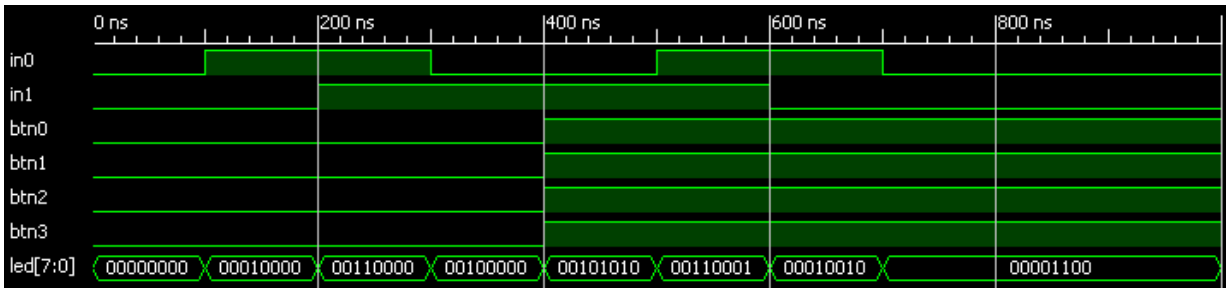
$$\text{Led}(0) = \text{IN0} \cdot \text{IN1} \cdot \text{Btn0} \qquad \text{Led}(3) = \overline{\text{IN0}} \cdot \text{Btn3}$$
$$\text{Led}(1) = (\text{IN0} \oplus \text{IN1}) \cdot \text{Btn1} \qquad \text{Led}(4) = \text{IN0}$$
$$\text{Led}(2) = \overline{(\text{IN0} + \text{IN1})} \cdot \text{Btn2} \qquad \text{Led}(5) = \text{IN1}$$

Σχήμα 1: Λογικό διάγραμμα σε επίπεδο πυλών και οι αντίστοιχες εξισώσεις του κυκλώματος 1.

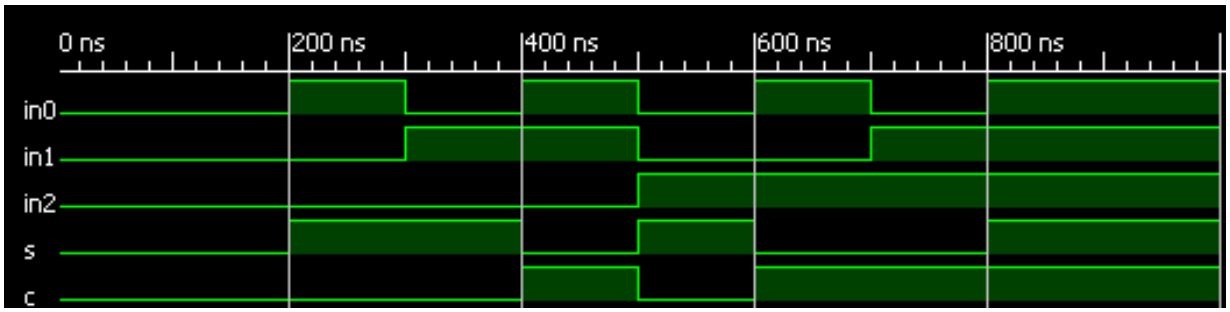


$$\text{First HalfAdder:} \quad \text{S} = \text{IN0} \oplus \text{IN1} \qquad \text{C}_0 = \text{IN0} \cdot \text{IN0}$$
$$\text{Second HalfAdder:} \quad \text{Led}(0) = \text{IN2} \oplus \text{S}_0 \qquad \text{C}_1 = \text{IN2} \cdot \text{S}_0$$
$$\text{Led}(1) = \text{C}_0 + \text{C}_1$$

Σχήμα 2: Block diagram και οι αντίστοιχες εξισώσεις του 1-bit full adder, δομημένος με δύο half adders και μία πύλη OR.



Σχήμα 3: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας του κυκλώματος 1 για όλες τις πιθανές περιπτώσεις.



Σχήμα 4: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας του Full Adder για όλες τις πιθανές περιπτώσεις. Το S και C αντιστοιχούν στο Led(0) και Led(1) αντίστοιχα.

3 Κυματομορφές-Προσομοίωση

Μετά την υλοποίηση των παραπάνω κυκλωμάτων σε κώδικα VHDL, προχωρήσαμε στη δημιουργία κώδικα test bench για τον έλεγχο της ορθής λειτουργίας τους μέσω προσομοίωσης του δομικού τους μοντέλου. Και στις δύο περιπτώσεις η προσομοίωση παρήγαγε τα αναμενόμενα αποτελέσματα.

Ειδικότερα, στο Σχήμα 3 παραθέτουμε την κυματομορφή της προσομοίωσης του κυκλώματος 1. Η συγκεκριμένη προσομοίωση καλύπτει όλες τις δυνατές περιπτώσεις, καθώς η ένδειξη κάθε led αντιστοιχεί αποκλειστικά στην έξοδο μίας πύλης του πρώτου επιπέδου (όταν φυσικά τα buttons είναι ενεργά). Επομένως, έως τα 400 ns που τα buttons δεν είναι πατημένα, τα λιγότερο σημαντικά ψηφία (leds) είναι 0000. Μετά τα 400 ns, όλα τα buttons ενεργοποιούνται και μπορούμε να δούμε τα λογικά αποτελέσματα της NOT (400-500 και μετά τα 700 ns), της XOR (400-500 και 600-700 ns), ths AND (500-600 ns), και της NOR (700 ns και μετά). Από την άλλη πλευρά, το πέμπτο και έκτο ψηφίο ενεργοποιούνται αναμενόμενα όταν τα In0 και In1 είναι ενεργά αντίστοιχα (ανεξαρτήτως button).

Τέλος, όσον αφορά την προσομοίωση του πλήρους αθροιστή, η παραγόμενη κυματομορφή (Σχήμα 4) παρουσιάζει όλες τις δυνατές περιπτώσεις πρόσθεσης τριών 1 bit ψηφίων (με χρονική σειρά, το + συμβολίζει εδώ αριθμητική πράξη): α) 0+0+0 = 00, β) 0+0+1 = 01, γ) 0+1+0 = 01, δ) 0+1+1 = 10, ε) 1+0+0 = 01, στ) 1+0+1 = 10, ζ) 1+1+0 = 10, και η) 1+1+1 = 11).

4 Συμπεράσματα

Η παρούσα εργαστηριακή άσκηση προσέφερε μία πρώτη πρακτική εξοικείωση τόσο με τη γλώσσα VHDL όσο και με τη σχεδιαστική ροή κυκλωμάτων μέσω του εργαλείου Xilinx ISE. Στηριζόμενοι σε δύο πολύ απλά παραδείγματα κυκλωμάτων, είχαμε την ευκαιρία να εξασκηθούμε στο δομικό τρόπο σχεδίασης κυκλωμάτων, χρησιμοποιώντας και συνδέοντας κατάλληλα ιεραρχικά κατώτερες μονάδες για την σύνθεση μεγαλύτερων σε ανώτερο ιεραρχικά επίπεδο. Επιπλέον, εξοικειωθήκαμε με τη χρήση προσομοίωσης του συμπεριφορικού/δομικού μοντέλου και την ανάγνωση των αντίστοιχων διαγραμμάτων χρονισμού. Τέλος, γνωρίσαμε μέσω του Xilinx ISE και την υπόλοιπη αλληλουχία διαδικασιών (Translation, Map, Place and Route) έως τη δημιουργία του αρχείου .bit και τη φόρτωσή του στην FPGA Basys 2.

5 Παράρτημα - Κώδικας VHDL

5.1 Full Adder

```
entity FullAdder is
    Port ( In0 : in  STD_LOGIC;
           In1 : in  STD_LOGIC;
           In2 : in  STD_LOGIC;
           S : out STD_LOGIC;
            C : out STD_LOGIC);
end FullAdder;

architecture Structural of FullAdder is
        signal HA0_C, HA0_S, HA1_C: STD_LOGIC;

        component OrGate is
                Port(   In0 : in STD_LOGIC;
                        In1 : in STD_LOGIC;
                        Out0 : out STD_LOGIC);
        end component;

        component HalfAdder is
                Port (  In0 : in  STD_LOGIC;
                        In1 : in  STD_LOGIC;
                        S : out  STD_LOGIC;
                        C : out  STD_LOGIC);
        end component;

        begin
                HA0: HalfAdder Port Map (In0 => In0,
                                         In1 => In1,
                                         S => HA0_S,
                                         C => HA0_C);

                HA1: HalfAdder Port Map (In0 => In2,
                                         In1 => HA0_S,
                                         S => S,
                                         C => HA1_C);

                Or0: OrGate Port Map (In0 => HA1_C,
                                         In1 => HA0_C,
                                         Out0 => C);

end Structural;
```

5.2 Half Adder

```
entity HalfAdder is
    Port ( In0 : in  STD_LOGIC;
           In1 : in  STD_LOGIC;
           S : out  STD_LOGIC;
           C : out  STD_LOGIC);
end HalfAdder;

architecture Structural of HalfAdder is
        component AndGate is
                Port        (In0 : in STD_LOGIC;
                             In1 : in STD_LOGIC;
                             Out0 : out STD_LOGIC);
        end component;
        component XorGate is
                Port        (In0 : in STD_LOGIC;
                             In1 : in STD_LOGIC;
                             Out0 : out STD_LOGIC);
        end component;
        begin
                XorU: XorGate Port Map (In0 => In0,
                                        In1 => In1,
                                        Out0 => S);

                AndU: AndGate Port Map (In0 => In0,
                                        In1 => In1,
                                        Out0 => C);
end Structural;
```