ΗΡΥ203 Προχωρημένη Λογική Σχεδίαση
5η Εργαστηριακή Άσκηση

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

Ημερομηνία Διεξαγωγής: 16 Μαΐου 2017

1 Σκοπός Εργαστηριακής Άσκησης

Η πέμπτη και τελευταία εργαστηριακή άσκηση περιλαμβάνει το τελικό βήμα μετατροπής του κυκλώματός μας σε μία απλή αριθμομηχανή. Μετά την σχεδίαση του κυκλώματος αναγνώρισης έξι αριθμητικών και λογικών πράξεων στο τέταρτο εργαστήριο (push, pop, add, sub, unary sub, swap), καλούμαστε τώρα να προσθέσουμε την καθεαυτή λειτουργικότητά τους. Συνεπώς, για την υλοποίησή τους χρειαζόμαστε τόσο τα συνδυαστικά κυκλώματα που εκτελούν τις πράξεις (datapath) όσο και τις αντίστοιχες FSM που ελέγχουν τα βήματά τους (control path).

2 Προεργασία-Περιγραφή

Ως συνέχεια της προηγούμενης εργαστηριακής άσκησης, σε αυτό το σημείο θα αναφερθούμε μόνο στα τμήματα του κυκλώματος που επεκτάθηκαν. Το ανώτατο ιεραρχικό επίπεδο (με τα modules StackCircuit, SSDCircuit) παρέμεινε ίδιο με αυτό της τέταρτης άσκησης. Σχεδόν ίδιο παραμένει και το υποκύκλωμα των SSDs, καθώς δεν προσθέσαμε κάποια νέα ένδειξη και η μόνη διαφορά έρχεται στην εμφάνιση του υπάρχοντος OVF και στην υπερχείλιση πράξης. Στις επόμενες υποενότητες θα αναφερθούμε στις αλλαγές στο επίπεδο του StackCircuit, και κυρίως στις προσθήκες στο επίπεδο του submodule του, FSMTop, όπου και προστέθηκε η πλειοψηφία των συμπληρωματικών κυκλωμάτων.

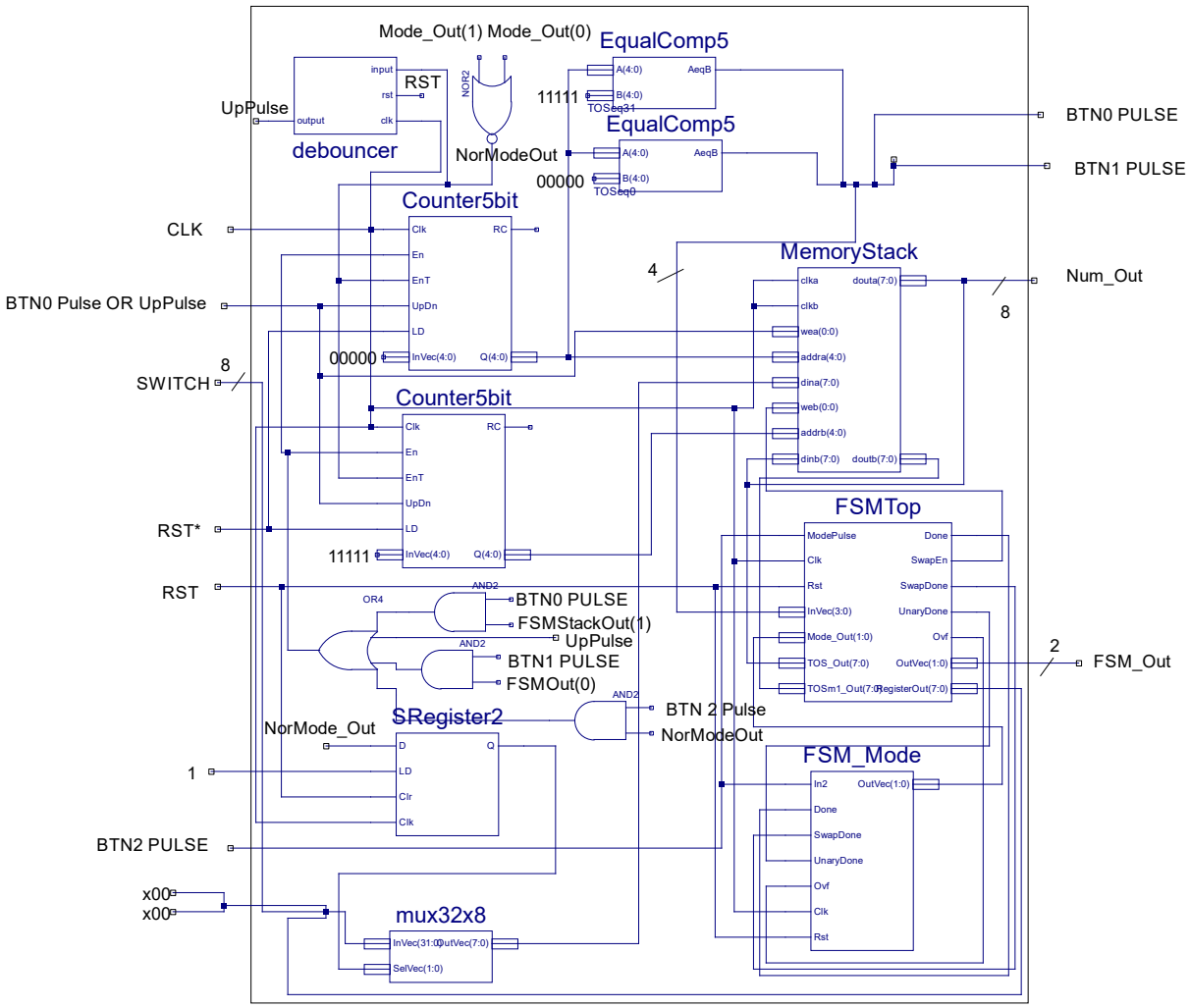
2.1 Γενική λειτουργία κυκλώματος

Προτού ξεκινήσουμε τη σύνοψη των αλλαγών στο επίπεδο αυτό (Σχήμα 1), είναι σημαντικό (για τη λογική της λύσης μας) να σημειώσουμε ότι στο κύκλωμά μας χρησιμοποιούμε δίθυρη μνήμη. Έχοντας δίθυρη μνήμη αξιοποιούμε και τον μετρητή TOS - 1, που είχαμε κατασκευάσει εξαρχής, για την πρόσβαση στην αντίστοιχη θέση της στοίβας. Επιπλέον, αυτό μας δίνει τη δυνατότητα να εκτελούμε μόνο ένα pop-push για κάθε πράξη, και κατά συνέπεια να μας αρκεί μόνο ένας 8-bit καταχωρητής.

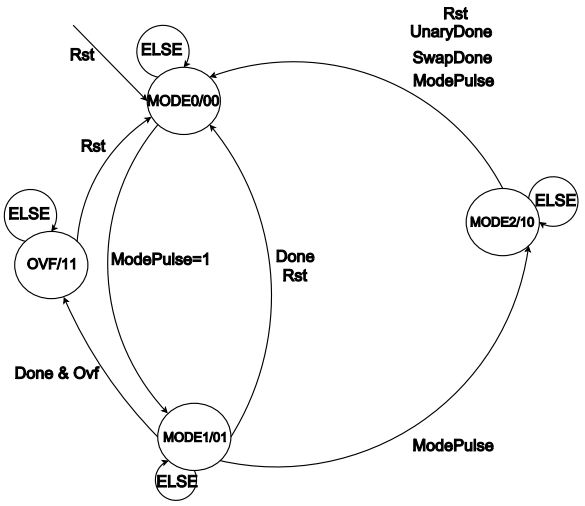
Για ευκολότερη υλοποίηση της απαραίτητης λειτουργικότητας, το κύκλωμά μας εκτελεί πάντα ένα pop κατά τη μετάβαση από Mode 0 → Mode 1, και ένα push κατά την επαναφορά στο Mode 0, ανεξαρτήτως εκτέλεσης κάποιας πράξης ή όχι. Όπως θα δούμε παρακάτω, η χρήση του σωστού αριθμού στο αυτόματο push εναπόκειται σε κατάλληλους πολυπλέκτες και σήματα επιλογής. Επιπλέον, με το επιτυχημένο πέρας κάθε πράξης το κύκλωμα επανέρχεται αυτόματα στο Mode 0 χωρίς εμείς να πατήσουμε κάποιο πλήκτρο. Αυτή ακριβώς η μετάβαση ενεργοποιεί αυτόματα και το αυτόματο push.

2.2 Τροποποιήσεις Stack module

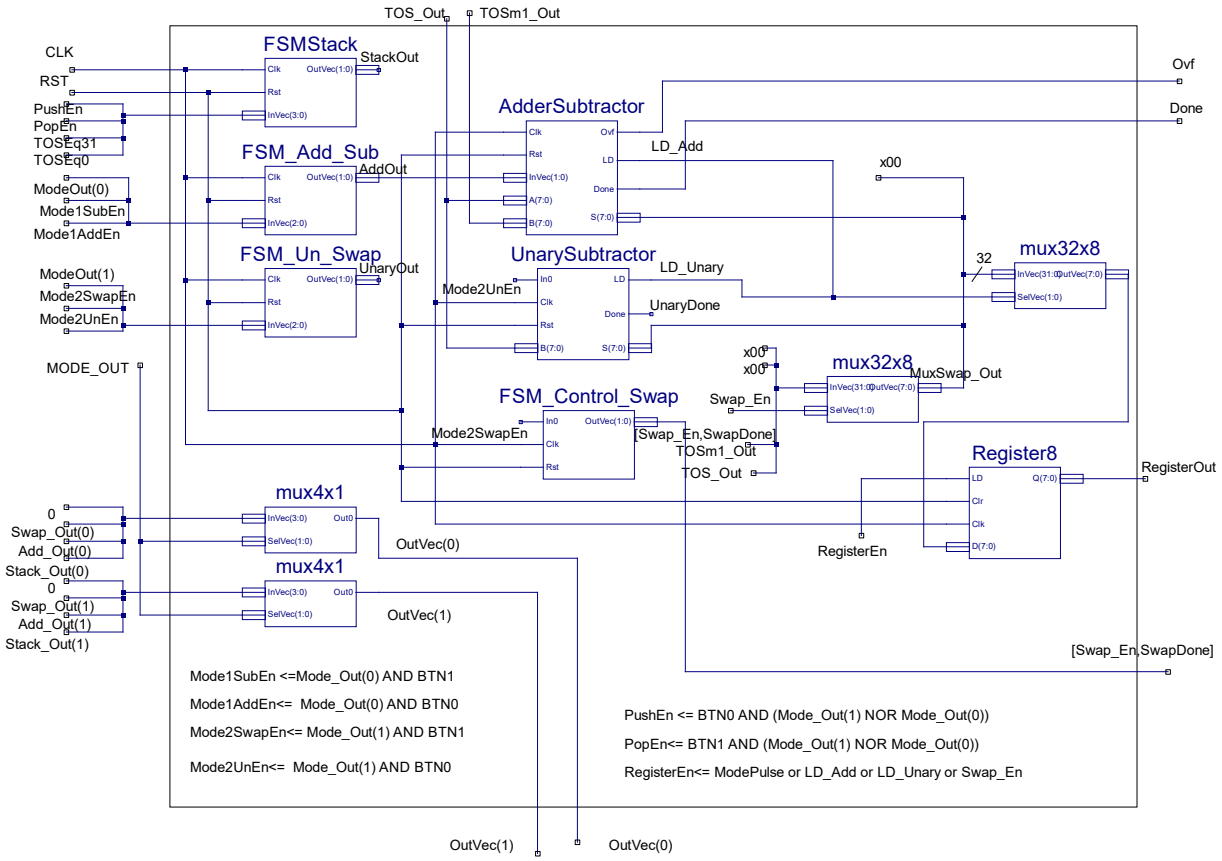
Μιας και οι περισσότερες προσθήκες νέων FSM πραγματοποιήθηκαν εσωτερικά του FSMTop, στο ανώτερο επίπεδο χρειάστηκε η εξαγωγή περισσότερων σημάτων για την επικοινωνία μεταξύ της ανώτερης ιεραρχικά FSM_Mode και αυτών των κατώτερων FSM. Έτσι, τα σήματα Done, SwapDone, UnaryDone



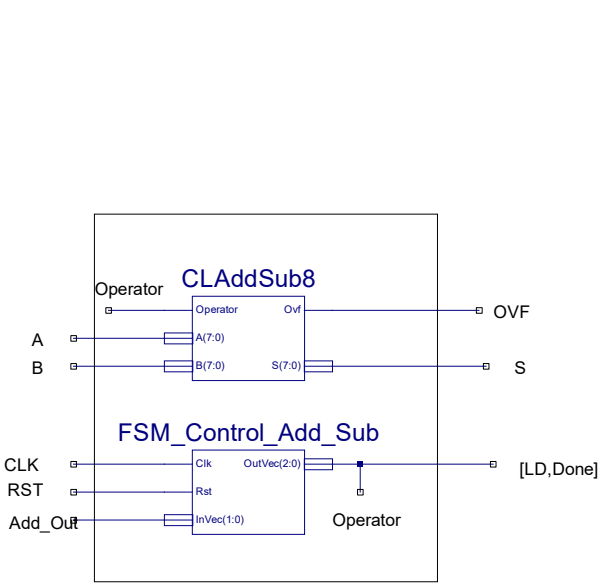
Σχήμα 1: Το τροποποιημένο StackCircuit module για τον έλεγχο των πράξεων. Το submodule FSMTop παρουσιάζεται στο Σχήμα 3 και το διάγραμμα καταστάσεων της FSM_Mode στο Σχήμα 2. Στις προσθήκες του παρόντος εργαστηρίου περιλαμβάνονται: ο πολυπλέκτης που επιλέγει τι θα δοθεί στη μνήμη για εγγραφή, ένα κύκλωμα debouncer (παραλλαγή του singlepulsegen που μας δόθηκε) και ένας καταχωρητής ολίσθησης (2-bit) για την εκτέλεση του Push κατά την επιστροφή στο Mode 0. Για χάρη καθαρότητας του σχήματος από το διάγραμμα παραλείπονται τα singlepulsegen που μεσολαβούν μεταξύ των πλήκτρων (εκτός του Reset) και του υπόλοιπου κυκλώματος.

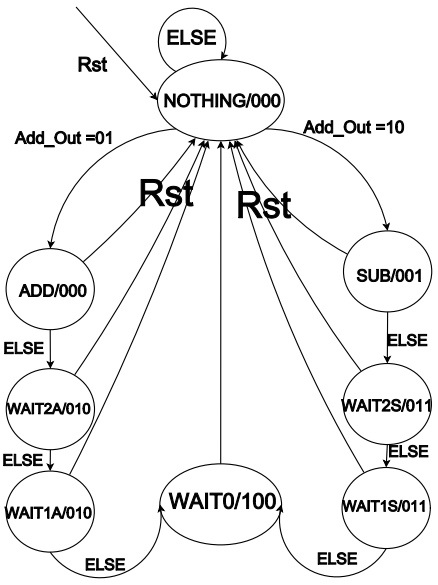


Σχήμα 2: Η τροποποιημένη μηχανή πεπερασμένων καταστάσεων ανώτερου ιεραρχικά επιπέδου FSM_Mode, η οποία ελέγχει τις μεταβάσεις μεταξύ των τριών modes. Πλέον περιέχει και μία τέταρτη κατάσταση που αντιστοιχεί στην υπερχείλιση κατά την πρόσθεση/αφαίρεση.

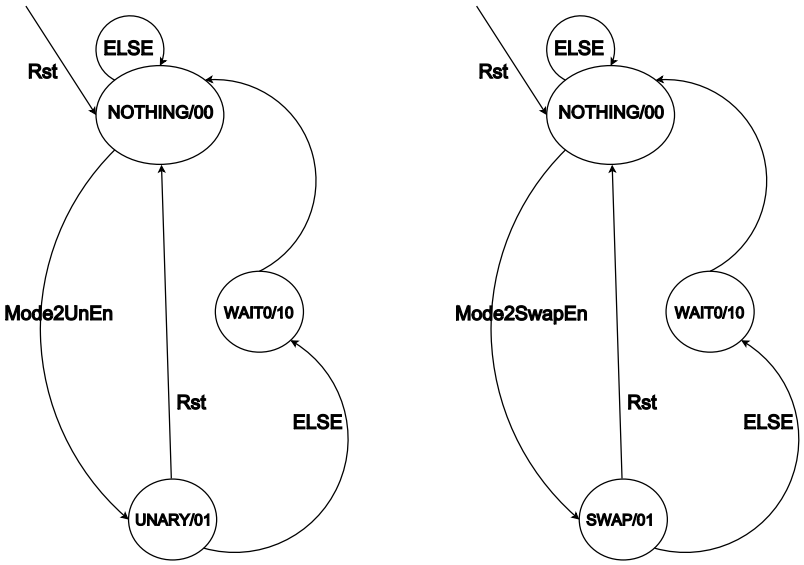


Σχήμα 3: Το επεκταμένο FSMTop module. Πέρα από τις FSM αναγνώρισης κάθε πράξης ανά mode που υλοποιήθηκαν στο τέταρτο εργαστήριο, προστέθηκαν ένας καταχωρητής (για την προσωρινή αποθήκευση του TOS ή TOS - 1 μετά από ένα pop), δύο κυκλώματα προσθαφαιρετών (ένα για πρόσθεση/αφαίρεση και ένα για μοναδιαία αφαίρεση, Σχήμα ;;), την FSM_Control_Swap (Σχήμα 5) για τον έλεγχο της αντίστοιχης πράξης, και δύο πολυπλέκτες 32x8 για την εγγραφή (αρχικά στον καταχωρητή και μετά στη μνήμη) του κατάλληλου αποτελέσματος κάθε φορά. Ο πολυπλέκτης 8x2 (στο παρόν φαίνεται ως δύο παράλληλοι 4x1) επιλέγει το σήμα που αλληλεπιδρά με το SSD και την ανώτερη FSM_Mode όπως και προηγουμένως.





Σχήμα 4: *Αριστερά*: Το υποκύκλωμα AdderSubtractor. Περιέχει το ακολουθιακό κύκλωμα προσθαφαιρέτη 8-bit, CLAddSub8, το οποίο συντίθεται από δύο κυκλώματα CLA 4-bit του δεύτερου εργαστηρίου, και την FSM εκτέλεσης των πράξεων FSM_Control_Add_Sub (*Δεξιά*). Το υποκύκλωμα UnarySubtractor περιέχει αντίστοιχα την FSM_Control_Unary και ως αφαιρετέο χρησιμοποιεί το 0.



Σχήμα 5: Οι μηχανές πεπερασμένων καταστάσεων FSM_Control_Unary (αριστερά) και FSM_Control_Unary (δεξιά) που ελέγχουν τα βήματα εκτέλεσης των αντίστοιχων πράξεων.

δηλώνουν το τέλος εκτέλεσης των πράξεων προσθαφαίρεσης, εναλλαγής και μοναδιαίας αφαίρεσης αντίστοιχα (όπως θα δούμε παρακάτω, είναι τα ms-bit της εξόδου των αντίστοιχων FSM), ενώ το σήμα Ovf ενημερώνει για υπερχείλιση κατά την πρόσθεση/αφαίρεση. Επιπλέον, το αποτέλεσμα της πράξης προς αποθήκευση δίνεται από το σήμα 8-bit RegisterOut, ενώ το SwapEn δίνει το σήμα για εγγραφή στο TOS - 1 (αντί στο TOS) αν το RegisterOut περιέχει αποτέλεσμα swap.

Το σήμα NorModeOut (που όπως και στο τέταρτο εργαστήριο δηλώνει ότι βρισκόμαστε στο Mode 0) πλέον χρησιμοποιείται και αυτό στον έλεγχο των αυτόματων push, pop κατά την είσοδο και έξοδο από το Mode 0. Εδώ για τον σωστό συγχρονισμό των σημάτων χρειάστηκε να χρησιμοποιήσουμε τόσο έναν τροποποιημένο debouncer (ίδιος με το κύκλωμα singlepulsegen που μας δόθηκε, με τη διαφορά ότι έχει ως αρχική κατάσταση την S2 αντί για την S0), όσο και έναν καταχωρητή ολίσθησης 2-bit. Επιπλέον, για το διαχωρισμό των περιπτώσεων push νέου αριθμού ή αυτόματου push προσθέσαμε έναν πολυπλέκτη (32×8, αλλά στην ουσία 16×8).

Τέλος, το Σχήμα 2 παρουσιάζει την τροποποιημένη FSM_Mode, όπου πλέον είναι εμφανής ο τρόπος αυτόματης μετάβασης είτε στο Mode 0 στο τέλος επιτυχημένης πράξης, είτε στο OVF για υπερχείλιση πρόσθεσης/αφαίρεσης (σημειωτέον ότι το OVF για τη στοίβα εμφανίζεται κανονικά στο Mode 0 όπως από το τρίτο εργαστήριο, και δεν αλλάζει ότι πράξη και αν κάνουμε στα υπόλοιπα modes).

2.3 Τροποποιήσεις FSMTop module

Όπως προαναφέρθηκε, τα συμπληρωματικά κυκλώματα τέλεσης των πράξεων προστέθηκαν όλα σε αυτό το επίπεδο (Σχήμα 3). Τα submodules FSMStack, FSM_Add_Sub, FSM_Un_Swap και ο πολυπλέκτης 8×2 ελέγχουν την αναγνώριση των πράξεων και μεταφέρθηκαν αυτούσια από το προηγούμενο εργαστήριο.

Ξεκινώντας την περιγραφή των προσθηκών, ο καταχωρητής 8-bit και δύο πολυπλέκτες 32×8 χρησιμοποιούνται σε όλες τις περιπτώσεις (εκτέλεσης ή μη πράξης) των mode 1, 2. Συγκεκριμένα, μέσω των πολυπλεκτών διαλέγουμε ποιον από τους πέντε αριθμούς (αποτέλεσμα πρόσθεσης/αφαίρεσης, αποτέλεσμα unary, αποτέλεσμα swap, τον αριθμό του TOS ή του TOS - 1) θα αποθηκεύσουμε προσωρινά στον καταχωρητή, ανάλογα πάντα με το ποια λειτουργία επιλέγεται κάθε φορά.

Τα υπόλοιπα τρία modules (AdderSubtractor, UnarySubtractor, FSM_Control_Swap) εκτελούν τις πράξεις και περιγράφονται στις αντίστοιχες υποενότητες που ακολουθούν.

2.4 Adder-Subtractor

Στο Σχήμα 4 παραθέτουμε το AdderSubtractor module, που περιέχει έναν προσθαφαιρέτη 8-bit, και την FSM_Control_Add_Sub που ελέγχει τα βήματα των πράξεων αυτών. Ο προσθαφαιρέτης έχει

κατασκευαστεί ενώνοντας δύο CLA 4-bit, δίνοντας ως κρατούμενο εισόδου τον Operator (1 για αφαίρεση), περνώντας τον (πιθανώς) αφαιρέτη από xor μαζί με αυτόν, και χρησιμοποιώντας τη συνάρτηση ελέγχου για υπερχείλιση (δείτε Κώδικα 5.1). Ο καθορισμός του Operator γίνεται από το LSBit της FSM_Control_Add_Sub, ενώ οι τελεστέοι έρχονται απευθείας από τα TOS, TOS - 1.

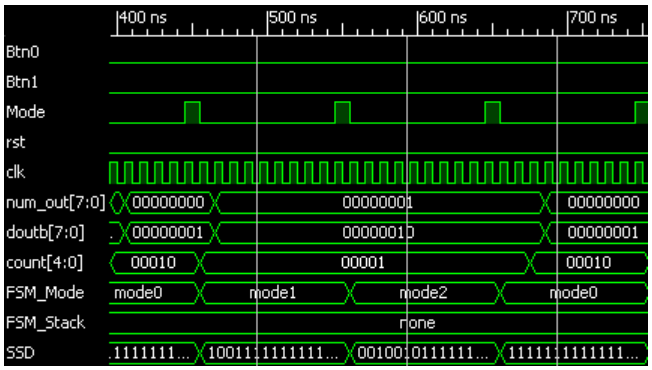
Για το σωστό συγχρονισμό των βημάτων της πράξης, η FSM_Control_Add_Sub εκτελεί συνολικά πέντε βήματα. Οι δύο πρώτες καταστάσης αναμονής δίνουν το σήμα LD, το οποίο και ενημερώνει τον καταχωρητή με το αποτέλεσμα της πράξης, ενώ στην τρίτη κατάσταση αναμονή παρέχει το σήμα Done που ειδοποιεί την FSM_Mode ότι το αποτέλεσμα της πράξης είναι έτοιμο για το τελικό push στη στοίβα.

2.5 Unary

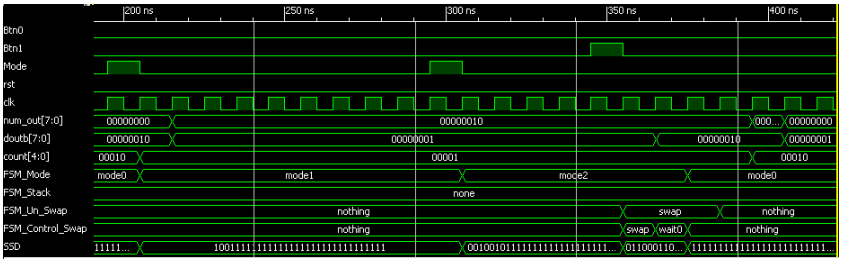
Ακριβώς αντίστοιχη είναι η διαδικασία Unary, καθώς για την εκτέλεσή της απλά εισάγαμε ένα δεύτερο αντίγραφο της AdderSubtractor (με όνομα UnarySubtractor), το οποίο και εκτελεί πάντα την αφαίρεση 0 - TOS. Αντίστοιχα γίνεται ο έλεγχος των βημάτων της πράξης από την FSM_Control_Unary (Σχήμα 5, αριστερά).

2.6 Swap

Για την εναλλαγή του TOS με TOS - 1 αρκεί μία FSM τριών κύκλων (Σχήμα 5, δεξιά). Με την επιλογή εκτέλεσης του swap, το LSB εξόδου (Swap_En) δίνει το σήμα αποθήκευσης του TOS - 1 στον καταχωρητή (και αποθήκευσης του TOS απευθείας στο TOS - 1), ενώ στο επόμενο βήμα το MSB ειδοποιεί την FSM_Mode ότι το αποτέλεσμα της πράξης είναι έτοιμο για το τελικό push.



Σχήμα 6: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας του κυκλώματος κατά την μετάβαση Mode 0 → Mode 1 → Mode 2 → Mode 0 με τριπλό πάτημα του πλήκτρου Mode (Btn2).

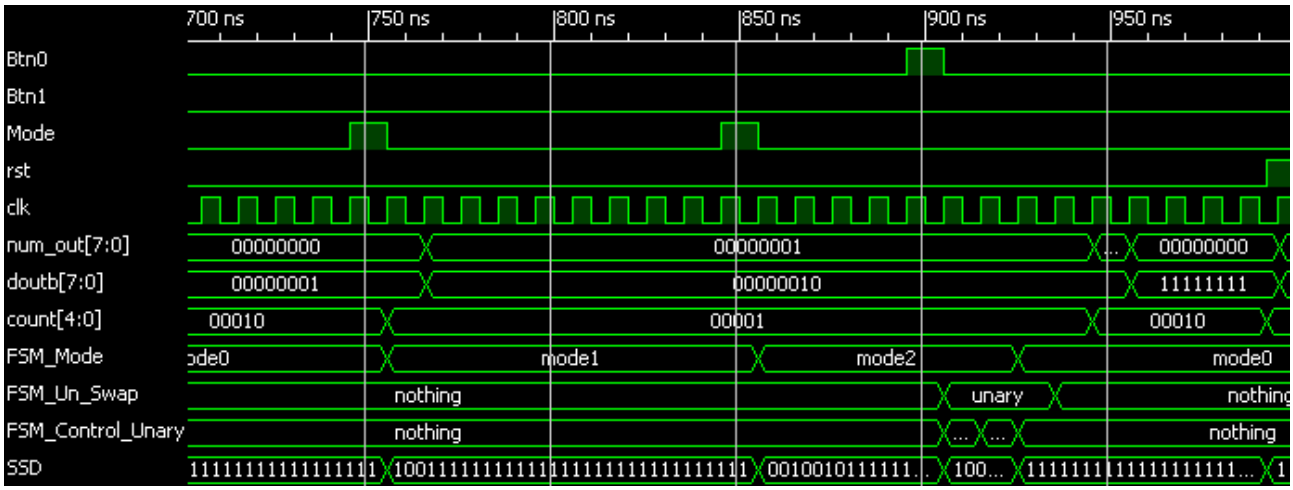


Σχήμα 7: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας swap.

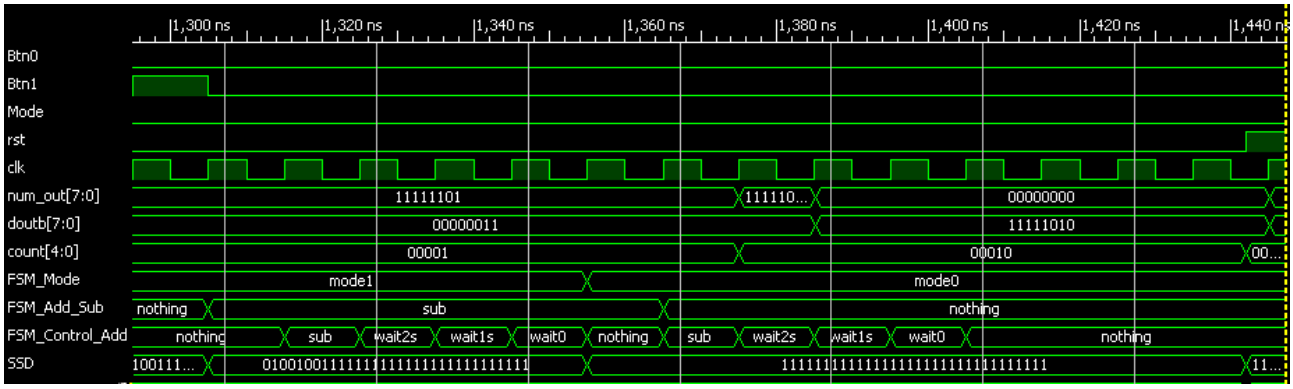
3 Κυματομορφές-Προσομοίωση

Με τις παραπάνω προσθήκες, το κύκλωμα απέκτησε πολλές νέες λειτουργίες που δεν έχουν δειχθεί στα προηγούμενα εργαστήρια. Λόγω του χωρικού περιορισμού στην αναφορά, εδώ θα αρκεστούμε στην παρουσίαση πέντε παραδειγμάτων λειτουργίας: 1) πλοήγηση στα modes χωρίς τέλεση πράξης, 2) swap, 3) unary, 4) αφαίρεση χωρίς υπερχείλιση, και 5) πρόσθεση με υπερχείλιση. Πέρα των παραπάνω, όλη η λειτουργικότητα μιας απλής αριθμομηχανής λειτουργεί κανονικά όπως δείχθηκε και στο board.

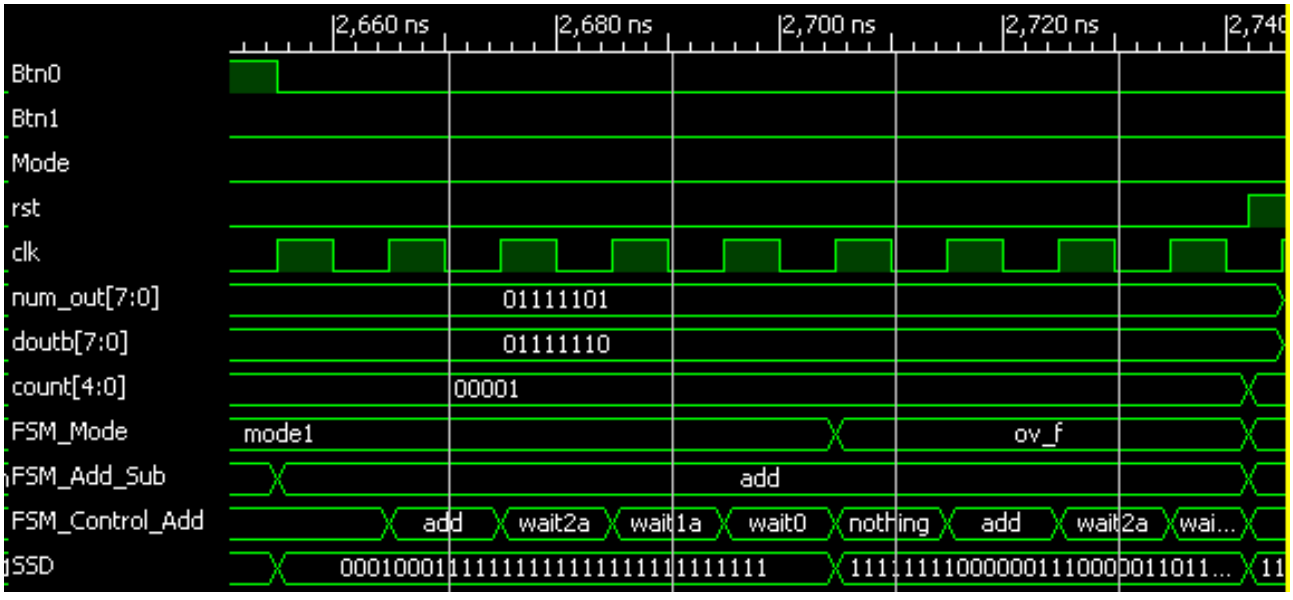
Όσον αφορά τη μετάβαση από το Mode 0 ξανά στο Mode 0 απλά με το πλήκτρο Mode (Σχήμα 6), παρατηρούμε ότι αμέσως μετά τη μετάβαση Mode 0 → Mode 1 πραγματοποιείται ένα pop, με συνέπεια τα TOS (num_out) και TOS - 1 (doutb) να δείχνουν μία θέση παρακάτω. Αντίστοιχα, μετά τη μετάβαση



Σχήμα 8: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας unary.



Σχήμα 9: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας αφαίρεσης ((-3) - (3) = -6) χωρίς overflow.



Σχήμα 10: Διάγραμμα χρονισμού προσομοίωσης της λειτουργίας πρόσθεσης (126+125) με overflow.

Mode 2 → Mode 0 έχουμε push (αντιγραφή με καθυστέρηση τριών κύκλων) με την επαναποθήκευση του αριθμού του TOS (από τον καταχωρητή στη μνήμη.)

Το swap (Σχήμα 7) πραγματοποιείται όπως περιγράφθηκε και ο προηγουμένως. Ευρισκόμενοι στο Mode 2, επιλέγουμε να γίνει swap με το Btn1. Με τη μετάβαση στην κατάσταση wait0 της FSM_Control _Swap πραγματοποιείται η αντιγραφή του TOS - 1 στον καταχωρητή (δεν φαίνεται στο παρόν σχήμα), και η αντιγραφή του TOS στο TOS - 1. Έχοντας τρεις κύκλους περιθώριο μέχρι το push μετά την επαναφορά στο mode 0, στο δεύτερο κύκλο αντιγράφεται το TOS - 1 από τον καταχωρητή στη μνήμη, ολοκληρώνοντας επιτυχημένα το swap στη στοίβα.

Με αντίστοιχο τρόπο λειτουργεί και το unary (Σχήμα 8). Έτσι, δύο κύκλους μετά την επαναφορά της FSM_Control_Unary στην αρχική της κατάσταση ο αντίθετος αριθμός αντιγράφεται (μέσω του καταχωρητή) στο TOS, και στον τρίτο κύκλο πραγματοποιείται το push.

Μιας και το unary στην ουσία αποτελεί ένα παράδειγμα αφαίρεσης, η αφαίρεση (Σχήμα 9) λειτουργεί με την ίδια χρονικά λογική. Αξίζει εδώ να σημειώσουμε ότι η FSM_Control_Add στην κατάσταση nothing ελέγχεται από την έξοδο της FSM_Add_Sub. Λόγω των επικοινωνούσων FSM τα βήματα της πρόσθεσης και αφαίρεσης ξεκινούν ένα κύκλο αργότερα, ενώ εκτελείται και μία δεύτερη επανάληψη της αλληλουχίας μεταβάσεων της FSM_Control_Add, το αποτέλεσμα της οποίας ωστόσο αγνοείται από το υπόλοιπο κύκλωμα. Κλείνοντας την ανάλυσή μας, στο Σχήμα 10 παραθέτουμε ένα παράδειγμα υπερχείλισης κατά την πρόσθεση. Βλέπουμε λοιπόν ότι η FSM_Mode (και άρα η έξοδος όλου του κυκλώματος) κλειδώνει στην κατάσταση υπερχείλισης με τον τερματισμό των βημάτων της πρόσθεσης.

4 Συμπεράσματα

Η πέμπτη και τελευταία εργαστηριακή άσκηση ολοκλήρωσε το σχεδιασμό μίας απλής αριθμομηχανής σε γλώσσα VHDL. Έχοντας ήδη τη λειτουργικότητα της στοίβας και αναγνώρισης των επιλογών μας από τα προηγούμενα εργαστήρια, σε αυτό το στάδιο προσθέσαμε τα λογικά ανεξάρτητα κυκλώματα εκτέλεσης των πράξεων, των οποίων και η λειτουργία ελέγχεται και συγχρονίζεται κατάλληλα με το υπόλοιπο κύκλωμα βάσει των καταστάσεων επικοινωνούσων FSM.

5 Παράρτημα - Κώδικας VHDL

5.1 CLAddSub8

```
1  architecture Structural of CLAddSub8 is
2  component CLA4 is
3      Port ( A : in STD_LOGIC_VECTOR (3 downto 0); B : in STD_LOGIC_VECTOR (3 downto 0);
4             Cin : in STD_LOGIC; S : out STD_LOGIC_VECTOR (3 downto 0);
5             C : out STD_LOGIC_VECTOR (3 downto 0));
6  end component;
7  signal C, B_un, Sum : STD_LOGIC_VECTOR (7 downto 0);
8  begin
9                CLA4U0: CLA4 Port Map ( A => A(3 downto 0),
10                                                  B => B_un(3 downto 0),
11                                                  Cin => Operator,
12                                                  S => Sum(3 downto 0),
13                                                  C => C(3 downto 0));
14               CLA4U1: CLA4 Port Map ( A => A(7 downto 4),
15                                                  B => B_un(7 downto 4),
16                                                  Cin => C(3),
17                                                  S => Sum(7 downto 4),
18                                                  C => C(7 downto 4));
19               XOR8U : For i in 0 to 7 generate
20                      B_un(i) <= B(i) xor Operator;
21               end generate;
22               S <= Sum;
23               Ovf <= ( C(7) xor C(6) ) ;
24 end Structural;
```