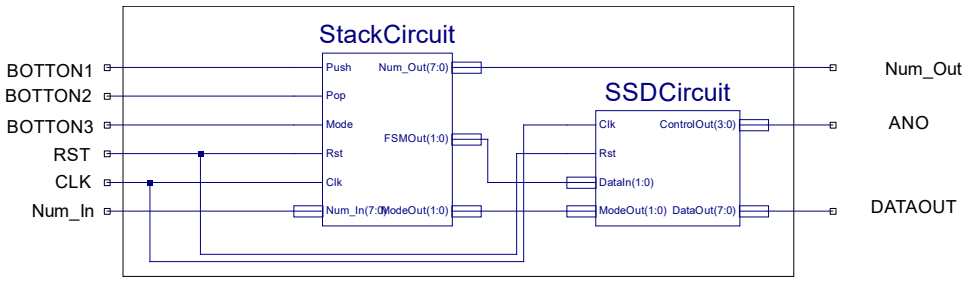
# ΗΡΥ203 Προχωρημένη Λογική Σχεδίαση
# 4η Εργαστηριακή Άσκηση

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

Ημερομηνία Διεξαγωγής: 28 Απριλίου 2017

## 1 Σκοπός Εργαστηριακής Άσκησης

Η τέταρτη εργαστηριακή άσκηση αποτελεί τη συνέχεια σχεδίασης μίας απλής αριθμομηχανής σε VHDL. Στόχος της παρούσας φάσης είναι η αναγνώριση και η αντίστοιχη απεικόνιση στα Seven Segment Displays (SSD), έξι αριθμητικών και λογικών πράξεων, ενώ στην πέμπτη εργαστηριακή άσκηση θα προστεθεί και η καθεαυτή λειτουργικότητά τους. Για την εκτέλεση των έξι πράξεων ορίζονται τρία modes, και είναι οι εξής: 1) Push και 2) Pop που ήδη έχουν υλοποιηθεί από την προηγούμενη άσκηση και αποτελούν το mode 0, 3) 2's complement addition και 4) substraction, που συνθέτουν το mode 1, και τέλος το mode 2 που περιέχει τις 5) unary substraction (-TOS) και 6) swap των TOS και TOS-1.




**Σχήμα 1:** Το πλήρες κύκλωμα στο ανώτατο ιεραρχικό επίπεδο. Το Σχήμα 2 απεικονίζει σε κατώτερο επίπεδο το StackCircuit. Η δομή του SSDCircuit παραμένει ίδια όπως παρουσιάστηκε στην τρίτη εργαστηριακή αναφορά.

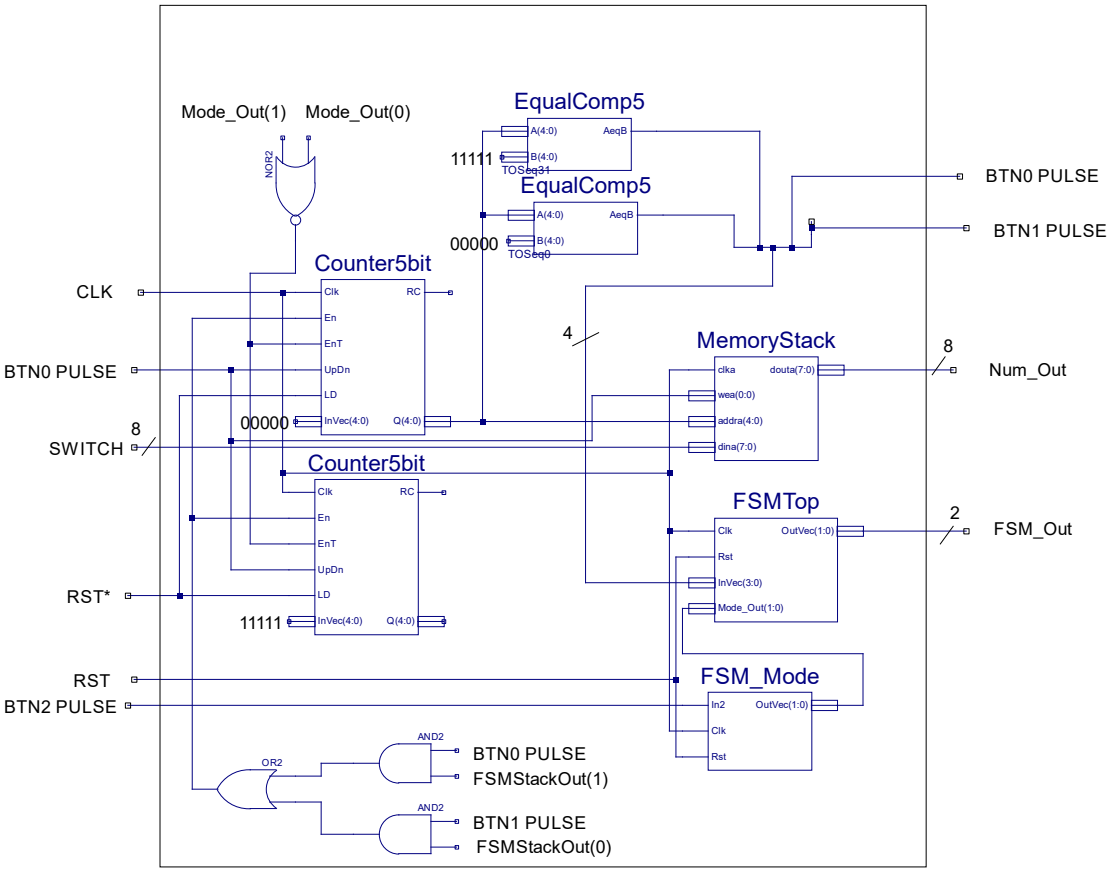
## 2 Προεργασία-Περιγραφή

Ως συνέχεια της προηγούμενης εργαστηριακής άσκησης, σε αυτό το σημείο θα αναφερθούμε μόνο στα τμήματα του κυκλώματος που επεκτάθηκαν. Το ανώτατο ιεραρχικό επίπεδο (Σχήμα 1) είναι σχεδόν το ίδιο με τη διαφορά ότι τώρα το η είσοδος mode που δεν χρησιμοποιόταν στο τρίτο εργαστήριο, τώρα συνδέθηκε με το button 2. Στις επόμενες υποενότητες θα αναφερθούμε στις αλλαγές εσωτερικά του StackCircuit, αλλά και στις μικρές αλλαγές απεικόνισης των νέων επιλογών στο SSDCircuit.

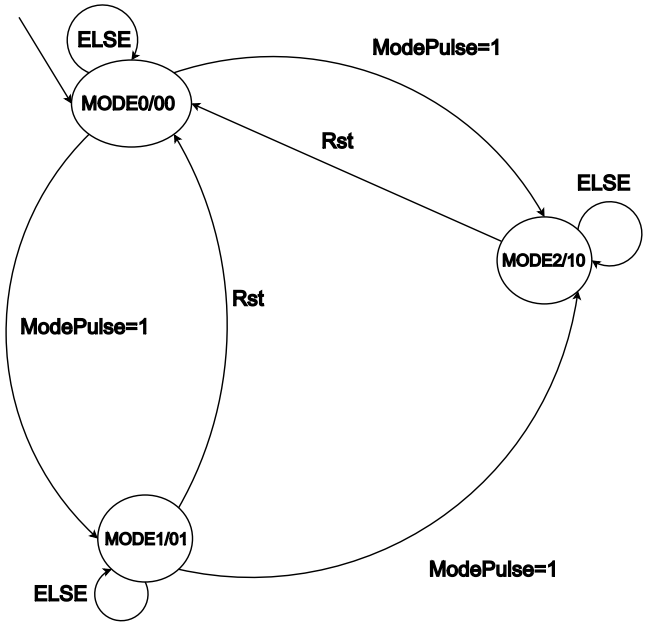
### 2.1 Τροποποιήσεις Stack module

Στο Σχήμα 2 ξαναπαρουσιάζουμε μαζί με τις νέες προσθήκες το κύκλωμα που υλοποιεί τη στοίβα. Όπως και τα PUSH, POP στο προηγούμενο μέρος της άσκησης, έτσι και το πλήκτρο του MODE περνάει μέσα από ένα debounce module (παραλείπεται στο σχήμα για μείωση της πολυπλοκότητάς του).

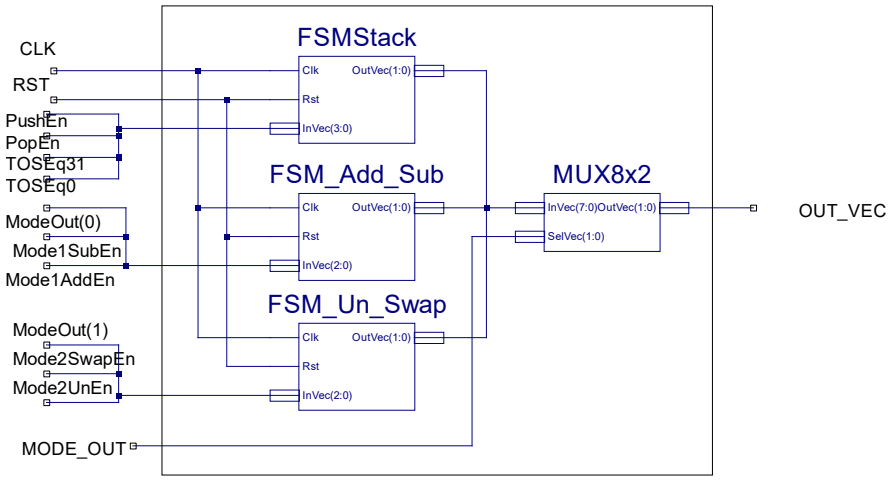
Οι κυριότερες αλλαγές μας για το αυτό το εργαστήριο έρχονται στη δημιουργία μικρών επικοινωνούσων FSM. Συγκεκριμένα, εισάγαμε μία νέα μηχανή καταστάσεων, την FSM_Mode (Σχήμα 3), της οποίας οι



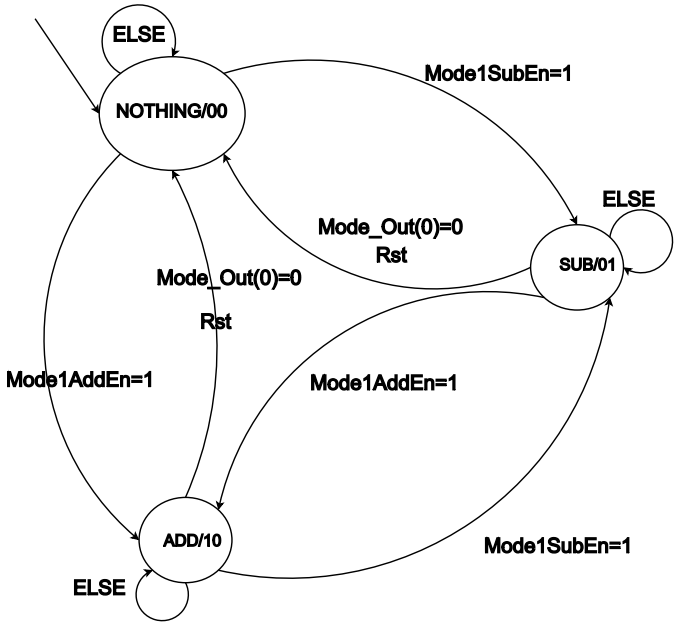

**Σχήμα 2:** Το τροποποιημένο StackCircuit module για τον έλεγχο των πράξεων. Το submodule FSMTop παρουσιάζεται στο Σχήμα 4 και το διάγραμμα καταστάσεων της FSM_Mode στο Σχήμα 3. Για χάρη καθαρότητας του σχήματος από το διάγραμμα παραλείπονται τα κυκλώματα παλμών εισόδου (singlepulsegen) που μεσολαβούν μεταξύ των πλήκτρων (εκτός του Reset) και του υπόλοιπου κυκλώματος.



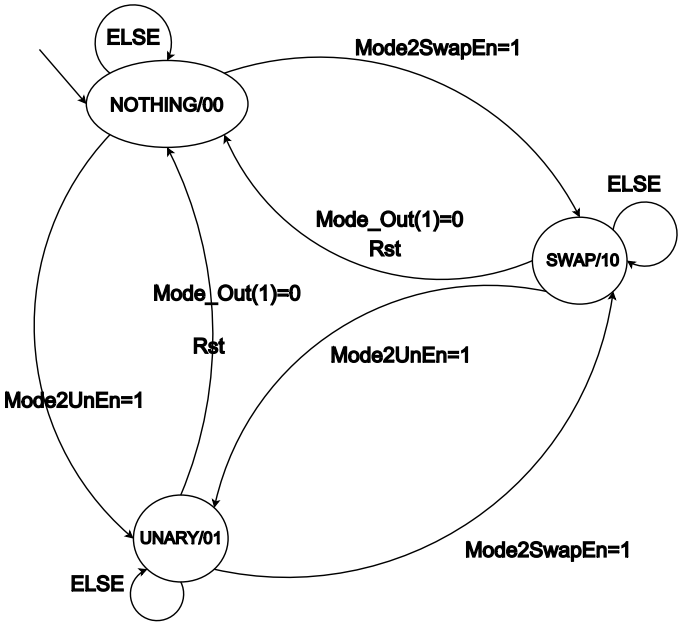

**Σχήμα 3:** Η μηχανή πεπερασμένων καταστάσεων ανώτερου ιεραρχικά επιπέδου FSM_Mode, η οποία ελέγχει τις μεταβάσεις μεταξύ των τριών modes.




**Σχήμα 4:** Το FSMTop module με τις FSM αναγνώρισης κάθε πράξης ανά mode που περιέχει (Σχήματα 5-6, η FSMStack παρουσιάστηκε στην τρίτη αναφορά). Η επιλογή του σήματος που προβάλλεται στα SSD πραγματοποιείται από την έξοδο της FSM_Mode μέσω ενός πολυπλέκτη.




**Σχήμα 5:** Η μηχανή πεπερασμένων καταστάσεων FSM_Add_Sub, η οποία ελέγχει την επιλογή πράξεων του mode 1.




**Σχήμα 6:** Η μηχανή πεπερασμένων καταστάσεων FSM_Un_Swap, η οποία ελέγχει την επιλογή πράξεων του mode 2.

τρεις καταστάσεις αντιστοιχούν στα τρία ζητούμενα modes. Έτσι, η 2-bit έξοδος αυτής FSM ελέγχει ιεραρχικά τη λειτουργία όλων των υπόλοιπων FSM του StackCircuit.

Οι υπόλοιπες FSM με τη σειρά τους έχουν τον έλεγχο των πράξεων κάθε mode, και για ευκολότερη διάκρισή τους στο κύκλωμα συνέθεσαν ένα ξεχωριστό submodule, το FSMTop. Η FSM_Add_Sub (Σχήμα 5) με τις τρεις καταστάσεις τις ελέγχει το τι θα απεικονιστεί τελικά στα SSD: 1) κατάσταση Nothing = απεικόνιση ΄1΄, 2) κατάσταση Add = απεικόνιση Α΄, και 3) κατάσταση Sub = απεικόνιση 'S'. Με αντίστοιχο τρόπο λειτουργεί και η FSM_Un_Swap (Σχήμα 6). Τα σήματα με όνομα ModeXYYYEn αντιστοιχούν στο πάτημα του αντίστοιχου mode και πράξης, ενώ και οι δύο FSM είναι προγραμματισμένες έτσι ώστε να επιστρέφουν στην κατάσταση Nothing με το που η αριθμομηχανή μεταβεί σε άλλο mode. Επιπλέον, η FSMStack παρέμεινε ίδια όπως περιγράφηκε στην τρίτη αναφορά, με τη διαφορά ότι πλέον δεν αλλάζει κατάσταση αν η FSM_Mode δεν βρίσκεται στο mode 0. Καθώς όλες οι κατώτερες ιεραρχικά FSM έχουν έξοδο 2-bit, αυτές περνούν από ένα πολυπλέκτη, ο οποίος και ελέγχει ποια FSM θα δωθεί ως έξοδος από το module (σύμφωνα το mode που βρισκόμαστε).

Εκτός των παραπάνω, στο StackCircuit χρειάστηκαν μικρές προσθήκες λογικών πυλών για τον έλεγχο των counters, όπου πλέον έχουμε και δεύτερο enable (EnT) για να μεταβάλλονται μόνο όταν βρισκόμαστε στο mode 0 (όπως δηλαδή ελέγχεται και η μνήμη). Τέλος, το σήμα ελέγχου του SSDCircuit από το StackCircuit επεκτάθηκε στα 4 bit (έξοδοι FSM_Mode, FSMTop), καθώς πλέον στο κύκλωμα έχουμε εννέα διαφορετικές απεικονίσεις (βλέπετε επόμενη υποενότητα).

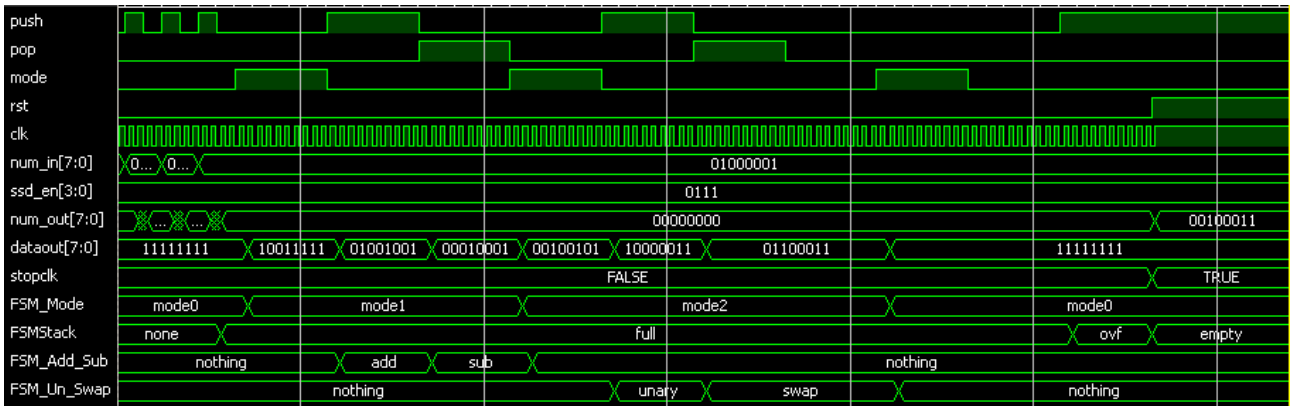
### 2.2 Τροποποιήσεις SSD module

Για τις απεικονίσεις στο SSD panel του basys 2 αξιοποιήσαμε σχεδόν αυτούσιο το SSD module που υλοποιήσαμε για το τρίτο εργαστήριο. Όπως αναφέρθηκε, πέρα τον απεικονίσεων 'E', 'F', 'OVF' του mode 0, σε αυτή την άσκηση προστέθηκαν οι '1', 'A', ' S', '2', 'U', '[]' των mode 1 και 2. Τροποποιώντας απλά τον αποκωδικοποιητή που είχαμε σχεδιάσει από 2-to-32, σε 4-to-32 καταφέραμε να απεικονίσουμε όλες τις περιπτώσεις (δείτε αντίστοιχο κώδικα στο Παράρτημα 5.1).

## 3 Κυματομορφές-Προσομοίωση

Μετά την υλοποίηση των παραπάνω κυκλωμάτων, προχωρήσαμε στην τροποποίηση του κώδικα test bench για τον έλεγχο της ορθής λειτουργίας τόσο εξ΄ ολοκλήρου, όσο και των νέων τμημάτων του κυκλώματος. Για αποφυγή επαναλήψεων με το προηγούμενο εργαστήριο, στην παρούσα αναφορά παρουσιάζουμε μόνο τη νέα λειτουργικότητα του κυκλώματος. Συγκεκριμένα, στο Σχήμα 7 παρουσιάζουμε ένα παράδειγμα εναλλαγής όλων των νέων modes και πράξεων. Μετά από συνεχόμενα push που γέμισαν τη μνήμη

(κατάσταση full), αλλάζουμε mode πατώντας το αντίστοιχο button. Στην απεικόνιση του πρώτου από τα αριστερά SSD βλέπουμε ότι αλλάζει σε ένδειξη ΄1΄. Στη συνέχεια, διαλέγοντας τα πλήκτρα push, pop (δηλαδή τα πλήκτρα 0 και 1), μεταβαίνουμε στις καταστάσεις add, sub της FSM_Add_Sub. Μεταβαίνοντας στη συνέχεια στο mode 2, επαληθεύουμε ότι η FSM_Add_Sub επανέρχεται στην αρχική κατάσταση nothing. Αντίστοιχα παρατηρούμε τις αναμενόμενες ενδείξεις του SSD και για τις υπόλοιπες καταστάσεις του κυκλώματος. Τέλος, για να δοκιμάσουμε την ακεραιότητα της λειτουργίας της στοίβας, βλέπουμε ότι με ένα ακόμη push μεταβαίνουμε σε ovf όπως αναμενόταν, ανεξάρτητα από το τις υπόλοιπες FSM.




**Σχήμα 7:** Τμήμα του διαγράμματος χρονισμού προσομοίωσης της λειτουργίας αναγνώρισης και απεικόνισης των νέων modes και πράξεων. Η λειτουργικότητα της στοίβας όπως παρουσιάστηκε στην τρίτη αναφορά παραμένει ίδια.

# 4 Συμπεράσματα

Η τέταρτη εργαστηριακή άσκηση αποτέλεσε το βήμα σχεδίασης του κυκλώματος αναγνώρισης και απεικόνισης των πράξεων που ζητούνται στην τελική αριθμομηχανή. Σε αυτό το εργαστήριο είχαμε την ευκαιρία να εξασκηθούμε για πρώτη φορά τόσο στη σχεδίαση επικοινωνούσων FSM, όσο και στη διαδικασία επέκτασης ενός προϋπάρχοντος κυκλώματος με νέα λειτουργικότητα.

# 5 Παράρτημα - Κώδικας VHDL

## 5.1 Decoder4to32

```
architecture Behavioral of Decoder4to32 is
begin
with InState select
OutVec        <= X"FF038371"        when "0000", -- printing OVF: F = X"71", V = X"83", O = X"03"
                 X"FFFFFF71"        when "0001", -- printing F
                 X"FFFFFF61"        when "0010", -- printing E: E = X"61"
                 X"9FFFFFFF"        when "0100", -- printing 1: 1 = X"9F"
                 X"11FFFFFF"        when "0101", -- printing A: A = X"11"
                 X"49FFFFFF"        when "0110", -- printing S: S = X"49"
                 X"25FFFFFF"        when "1000", -- printing 2: 2 = X"25" 0010 0101
                 X"83FFFFFF"        when "1001", -- printing U: U = X"83"
                 X"630FFFFF"        when "1010", -- printing []: [ = X"63", ] = X"0F"
                 X"FFFFFFFF"        when others; -- printing NONE (all off)
end Behavioral;
```