ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

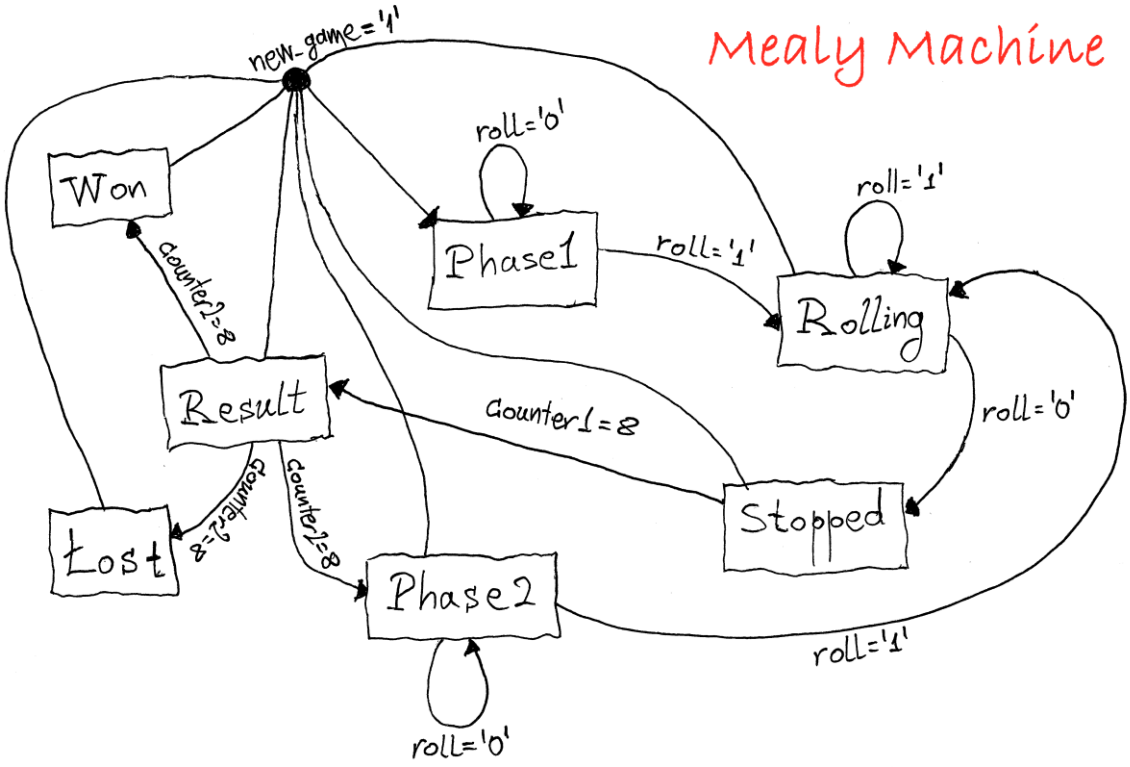
Εργασία Μαθήματος ***Σχεδίαση Υπολογιστικών Συστημάτων***

Τελική εργασία	***Παιχνίδι με ζάρια, ενός παίκτη***
Όνομα φοιτητή – Αρ. Μητρώου	Στέφανος-Μάριος Δανιήλ, Π16026
	Χρήστος Κατέβας, Π18068
Ημερομηνία παράδοσης	15/07/2021, 23:59:59

Περιεχόμενα

Μηχανή Πεπερασμένων Καταστάσεων (FSM)

Η μηχανή πεπερασμένων καταστάσεων που κατασκευάσαμε για το παιχνίδι, αποτελείται από 7 καταστάσεις : Phase1 , Rolling, Stopped, Result, Phase2, Won, Lost. Η αρχική κατάσταση είναι η Phase1. Μπορούμε να μεταβούμε ανά πάσα στιγμή από οποιαδήποτε κατάσταση στην αρχική, χρησιμοποιώντας το ασύγχρονο reset το οποίο στην πραγματικότητα είναι η είσοδος *new_game* του κυκλώματός μας.

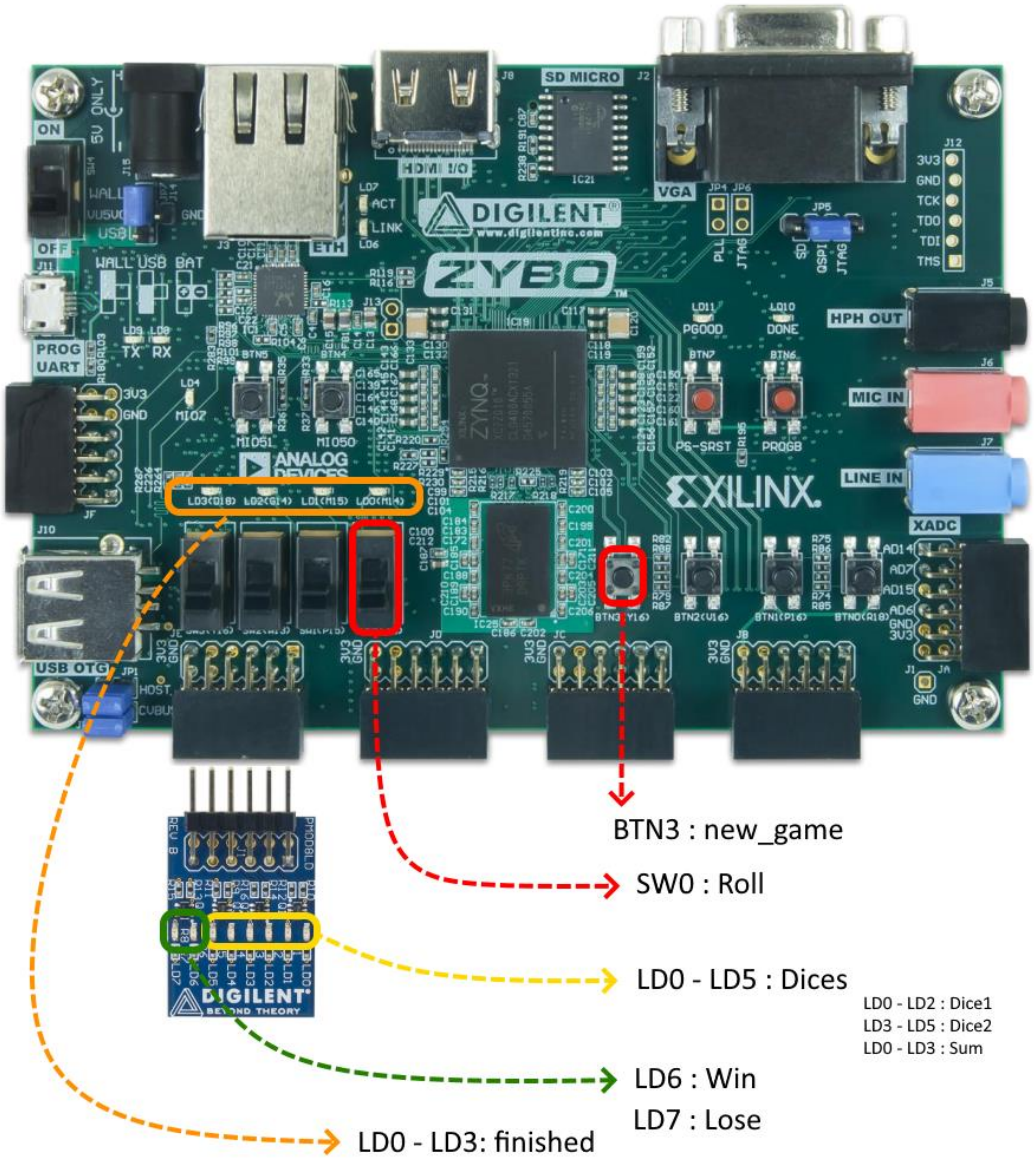


Εικόνα 1: Αφαιρετικό σχήμα FSM

Ξεκινώντας από την *Phase1*, όσο ο διακόπτης Roll είναι σβηστός παραμένουμε στην ίδια κατάσταση. Μπορούμε να μεταβούμε στην επόμενη κατάσταση, δηλαδή την Rolling, ενεργοποιώντας το Roll. Αφού μεταβούμε στην *Rolling*, τα ζάρια ξεκινούν και κυλούν. Όσο το Roll παραμένει ενεργό, οι τιμές των ζαριών θα φαίνονται στα LEDS. Για να μεταβούμε στην κατάσταση Stopped, θα πρέπει να απενεργοποιήσουμε το Roll. Αφού βρισκόμαστε στην κατάσταση *Stopped* ο μετρητής counter1 ξεκινά να μετράει και τα ζάρια για τα επόμενα 4 δευτερόλεπτα, συνεχίζουν να παίρνουν τιμές. Μετά τη λήξη του μετρητή μεταβαίνουμε στην κατάσταση *Result* και τα ζάρια παγώνουν. Σε αυτή την κατάσταση ενεργοποιείται ο μετρητής counter2 και τα LEDS δείχνουν τις τιμές των ζαριών που σταμάτησαν να κυλούν. Μετά τη λήξη του μετρητή, το που θα μεταβούμε αμέσως μετά εξαρτάται καθαρά από:

- Το σήμα first_time
- Το άθροισμα των ζαριών

Πιο συγκεκριμένα εάν έχουμε ρίξει τα ζάρια μονάχα μια φορά, τότε μετά τη λήξη του μετρητή το άθροισμα των ζαριών είναι 7 ή 11 μεταβαίνουμε στην κατάσταση Won. Εάν το άθροισμα ισούται με 2, 3 ή 12 τότε μεταβαίνουμε στην κατάσταση Lost. Διαφορετικά το άθροισμα των ζαριών αποθηκεύεται στον καταχωρητή *dice_sum* και μεταβαίνουμε στην κατάσταση Phase2. Στην περίπτωση που δεν ρίχνουμε τα ζάρια πρώτη φορά, δηλαδή το σήμα first_time έχει την τιμή 0, τότε στην κατάσταση Won θα μεταβούμε εάν το άθροισμα των ζαριών είναι 12 ή ίσο με το άθροισμα που έχει αποθηκευτεί στον dice_sum. Στη Lost θα μεταβούμε εάν το άθροισμα είναι ίσο με 7 ή 11 και τέλος στην Phase2 κατά τον ίδιο τρόπο με τις προηγούμενες ρίψεις. Στην κατάσταση *Phase2* το κύκλωμα δείχνει στα LEDS του, το άθροισμα των ζαριών πλέον και όχι τις τιμές των ζαριών. Όσο η είσοδος Roll είναι ανενεργή θα παραμένουμε σε αυτή την κατάσταση. Για να συνεχίσουμε τη ρίψη των ζαριών, εφόσον δεν έχει υπάρξει ακόμη αποτέλεσμα, θα πρέπει να ανοίξουμε τον διακόπτη Roll. Και συνεχίζεται η ίδια διαδικασία με τις καταστάσεις έως ότου υπάρξει νίκη ή ήττα. Στην κατάσταση *Won* το κύκλωμα έχει αναμμένο το LED6 που υποδεικνύει την νίκη, καθώς και στα LED0-LED3 το άθροισμα των ζαριών που κέρδισαν. Αντίστοιχα στην *Lost* θα ανάψει το LED7 για την ήττα και τα ίδια LEDS για το άθροισμα που έχασε το παιχνίδι.



Εικόνα 2: Κάτοψη του συστήματος

Περιγραφή της σχεδίασης

Ας αναλύσουμε λίγο περισσότερο τον τρόπο με τον οποίο υλοποιήσαμε το παιχνίδι. Αρχικά ο τρόπος σχεδίασης του συστήματός μας είναι δομικός. Χρησιμοποιούμε δύο συμπεριφορικά μοντέλα, το ένα είναι η γεννήτρια τιμών των ζαριών μας και το άλλο είναι ο διαιρέτης συχνότητας ρολογιού από 125MHz σε 2Hz, που χρησιμοποιείται αφενός στην παραγωγή των ζαριών και αφετέρου στην ρεαλιστική μέτρηση χρόνου για τις καταστάσεις Stopped και Result. Θα πούμε περισσότερα για τη γεννήτρια των ζαριών στο επόμενο κεφάλαιο. Το δομικό μας μοντέλο αποτελεί και το top level μοντέλο στην ιεραρχία.

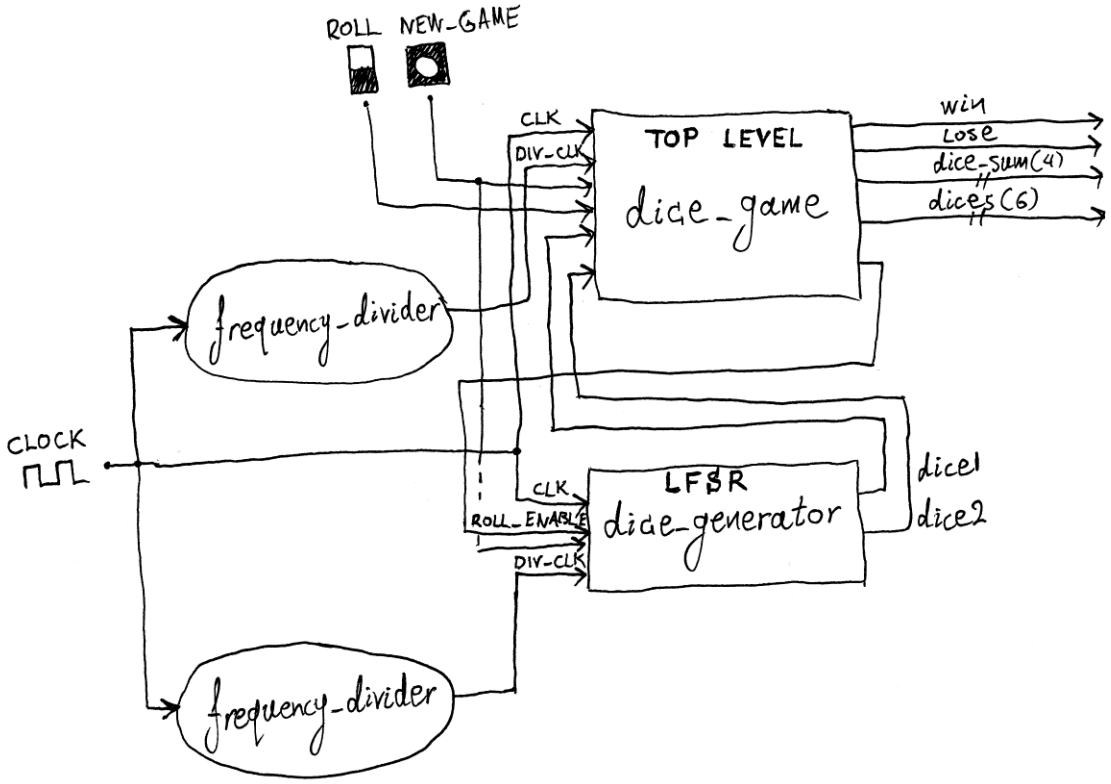
Εφόσον έχουμε περιγράψει τη μηχανή καταστάσεων προηγουμένως θα εξηγήσουμε εν συντομία τον τρόπο με τον οποίο το κύκλωμα λειτουργεί εσωτερικά. Όπως έχουμε αναφέρει, το κύκλωμα έχει δύο εισόδους, την Roll για να ρίξουμε τα ζάρια και το new_game για να ξεκινήσουμε ένα νέο παιχνίδι. Έχουμε μια γενική έξοδο dices, του κυκλώματος, που εμφανίζεται στα LED0-LED5 όπου τα τρία κάτω LEDS αναπαριστούν το πρώτο ζάρι και τα τρία πάνω LEDS το δεύτερο. Επιπλέον το άθροισμα τους εμφανίζεται στα LED0 – LED3 στις καταστάσεις Phase2, Won και Lost (αναλόγως το άθροισμα των ζαριών που θα τύχει). Τα LED6 και LED7 έχουν ανατεθεί για την ένδειξη του σήματος win για τη νίκη και του lose για την ήττα αντίστοιχα. Επιπροσθέτως όταν προκύψει οποιοδήποτε αποτέλεσμα στο παιχνίδι, τότε εκτός των 8 επιπρόσθετων LEDS, οδηγούμε το 4-bit σήμα finished στα τέσσερα ενσωματωμένα LEDS της πλακέτας LED0-LED3 τα οποία αναβοσβήνουν κάθε 2 δευτερόλεπτα και φανερώνουν το τέλος του παιχνιδιού. Το νέο παιχνίδι θα αρχίσει πιέζοντας το new_game στο κουμπί BTN3.

Στο προηγούμενο κεφάλαιο αναφέραμε το σήμα first_time. Το σήμα αυτό χρησιμοποιείται με σκοπό να αναγνωρίσουμε το αν ο παίκτης ρίχνει για πρώτη φορά τα ζάρια του ή όχι. Ένα παιχνίδι ξεκινά με το σήμα αυτό να παίρνει την τιμή 1. Από τη στιγμή που δεν υπάρξει κάποιο αποτέλεσμα την πρώτη φορά, ο παίκτης θα ξαναρίξει τα ζάρια και το σήμα first_time θα λάβει την τιμή 0, ώστε να διαχειριστούμε το νέο άθροισμα των ζαριών που θα προκύψει (επειδή το αποτέλεσμα διαφέρει μεταξύ της πρώτης ρίψης και όλων των υπόλοιπων).

Για την αποθήκευση του αθροίσματος των ζαριών, χρησιμοποιούμε τον καταχωρητή dice_sum. Αυτός ο καταχωρητής έχει αρχική τιμή 0 και κάθε φορά που ο παίκτης μεταβαίνει στην κατάσταση Phase2, το παρόν άθροισμα αποθηκεύεται με σκοπό να ελεγχθεί κατά την επόμενη ρίψη. Σημειώνεται επίσης ότι στην εκκίνηση ενός νέου παιχνιδιού, αυτός ο καταχωρητής πρέπει να μηδενίζεται!

Η πυροδότηση του καταχωρητή dice_sum εκτός από το ρολόι χρειάζεται και ένα σήμα επίτρεψης, για να τον χρησιμοποιούμε μόνο σε ορισμένες χρονικές περιόδους. Δηλαδή μόνο στην περίοδο που ο παίκτης μεταβαίνει από την κατάσταση Result στην Phase2. Αυτό το σήμα επίτρεψης είναι το sum_enable.

Τέλος έχουμε τους τρείς μετρητές μας. Ο counter1 χρησιμοποιείται για να χρονομετρήσει 4 δευτερόλεπτα από τη στιγμή που βρεθήκαμε στην κατάσταση Stopped. Αντίστοιχα ο counter2, χρονομετρά 4 δευτερόλεπτα από τη στιγμή που μεταβήκαμε στην Result. Και οι δύο counters παίρνουν ως ρολόι το διαιρεμένο ρολόι των 2Hz, επομένως ο έλεγχος της τιμής τους, γίνεται διπλασιάζοντας τον χρόνο που θέλουμε να επιτύχουμε. Για παράδειγμα, εάν χρησιμοποιήσουμε ρολόι 1Hz θα μετρούσαμε μέχρι το 4. Στην περίπτωσή μας, μετράμε μέχρι το 8 για να έχουμε το ίδιο αποτέλεσμα. Ο μετρητής res_counter λειτουργεί με παρόμοιο τρόπο.

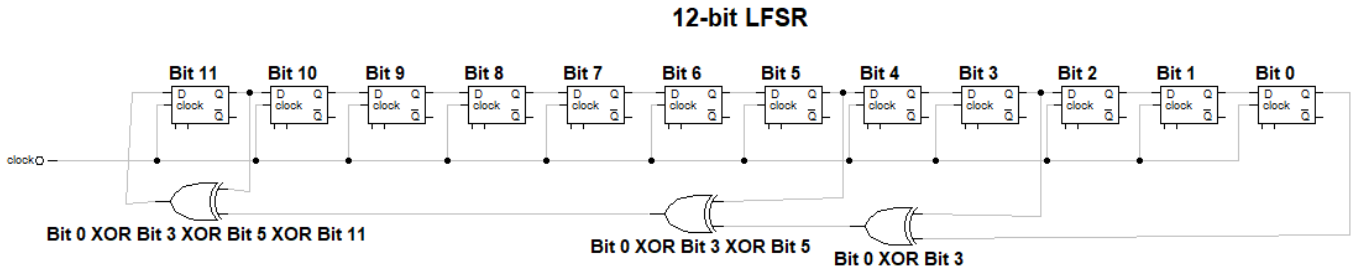


Εικόνα 3: Δομική περιγραφή του μοντέλου

Γεννήτρια τιμών για τα ζάρια

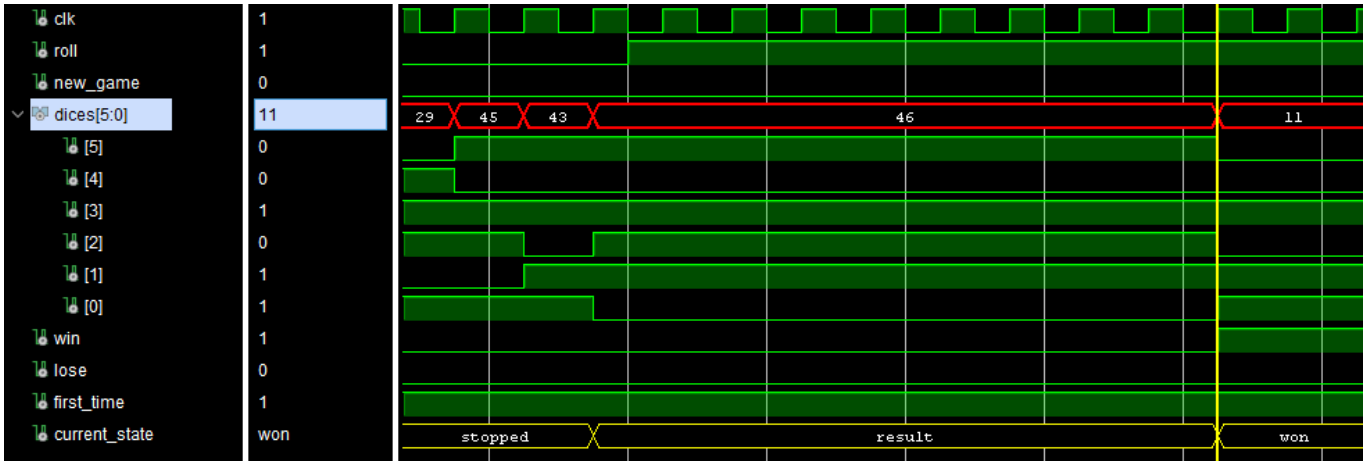
Για να επιτύχουμε μια ψευδοτυχαία εναλλαγή τιμών στα ζάρια μας, χρησιμοποιήσαμε έναν 12-bit LFSR καταχωρητή. Δέχεται ως ρολόι το ρολόι του συστήματος και η αρχική του κατάσταση είναι η `000000000011`. Ο καταχωρητής αυτός σε κάθε περίοδο του ρολογιού κάνει ολίσθηση των bit του ενώ ταυτόχρονα, χρησιμοποιείται ένα σήμα feedback το οποίο είναι τα tapping bits του LFSR. Αναλυτικότερα, για τα tapping bits του LFSR επιλέξαμε κάποια(1) από τα bit του καταχωρητή ούτως ώστε να τα περάσουμε από πύλες XOR και να τα ανατροφοδοτήσουμε στον LFSR. Τα bits που επιλέχθηκαν για το σήμα feedback είναι τα 0,3,5 και 11, καθώς με αυτά επιτυγχάνουμε το μεγαλύτερο εύρος που μπορεί να λάβουν οι ακολουθίες από bit για τον LFSR, δεδομένου ότι ο αριθμός των taps θα είναι ο ελάχιστος (βέλτιστη επιλογή).

Επιπλέον χρησιμοποιούμε το διαιρεμένο ρολόι των 2Hz και το σήμα *enable*, ούτως ώστε όταν ο χρήστης ανοίξει τον διακόπτη Roll, τα ζάρια να δέχονται δύο διακριτές τιμές το δευτερόλεπτο. Ακόμη χρησιμοποιούνται δύο εσωτερικά σήματα dice1_tmp και dice2_tmp που λαμβάνουν 3 bit το καθένα από τον LFSR. Αυτά τα σήματα περνούν προς την έξοδο μονάχα εάν έχουν έγκυρες τιμές για τα ζάρια, δηλαδή διάφορες του 0 και του 7. Σε αντίθετη περίπτωση, η έξοδος δέχεται μια μικρή διόρθωση.



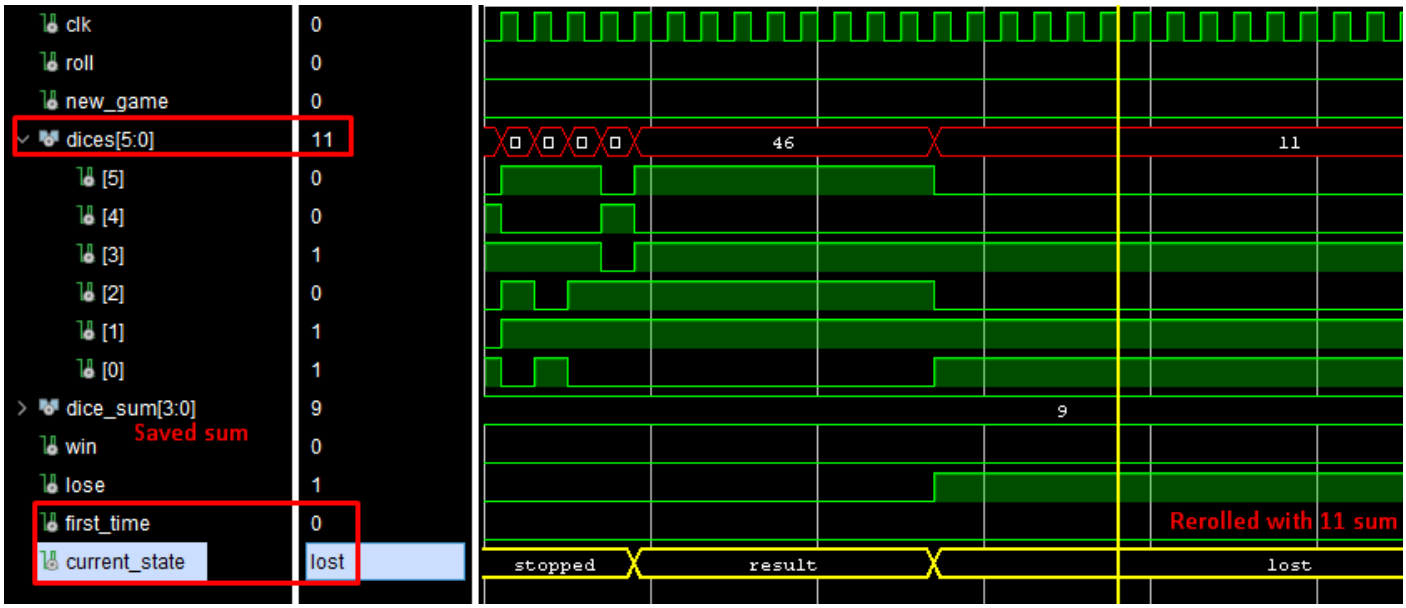
Εικόνα 4: Προσεγγιστική αναπαράσταση του LFSR

Παραδείγματα εκτέλεσης



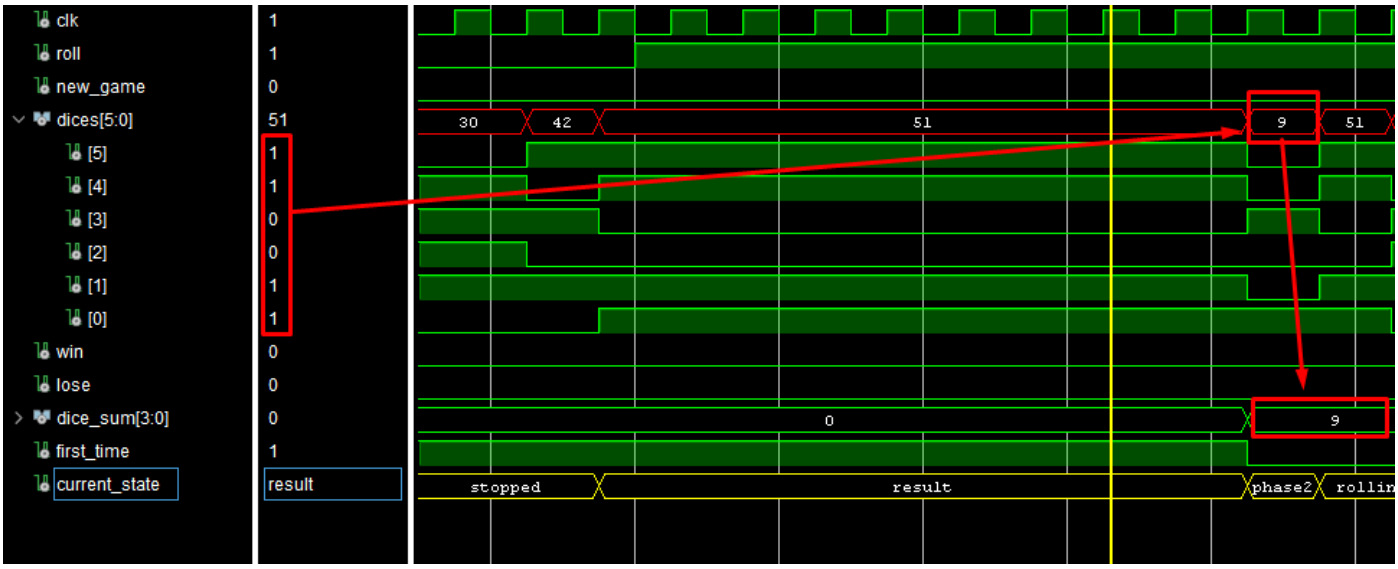
Εικόνα 5: Νίκη παιχνιδιού κατά την πρώτη φορά ρίψης των ζαριών

Στην εικόνα 5 βλέπουμε την νίκη ενός παιχνιδιού ρίχνοντας για πρώτη φορά τα ζάρια. Μπορούμε εύκολα να καταλάβουμε εάν είναι η πρώτη ρίψη καθώς το σήμα first_time έχει ακόμη την τιμή 1. Ο παίκτης κέρδισε το παιχνίδι εφόσον το άθροισμα των ζαριών είναι 11. Η έξοδος dices στην κατάσταση Won δείχνει το άθροισμα των ζαριών, ενώ προηγουμένως φαίνονται απλά οι τιμές των ζαριών ενωμένες (concatenated '&'), επομένως σαν νούμερο δεν έχει κάποιο ιδιαίτερο νόημα, το αγνοούμε.



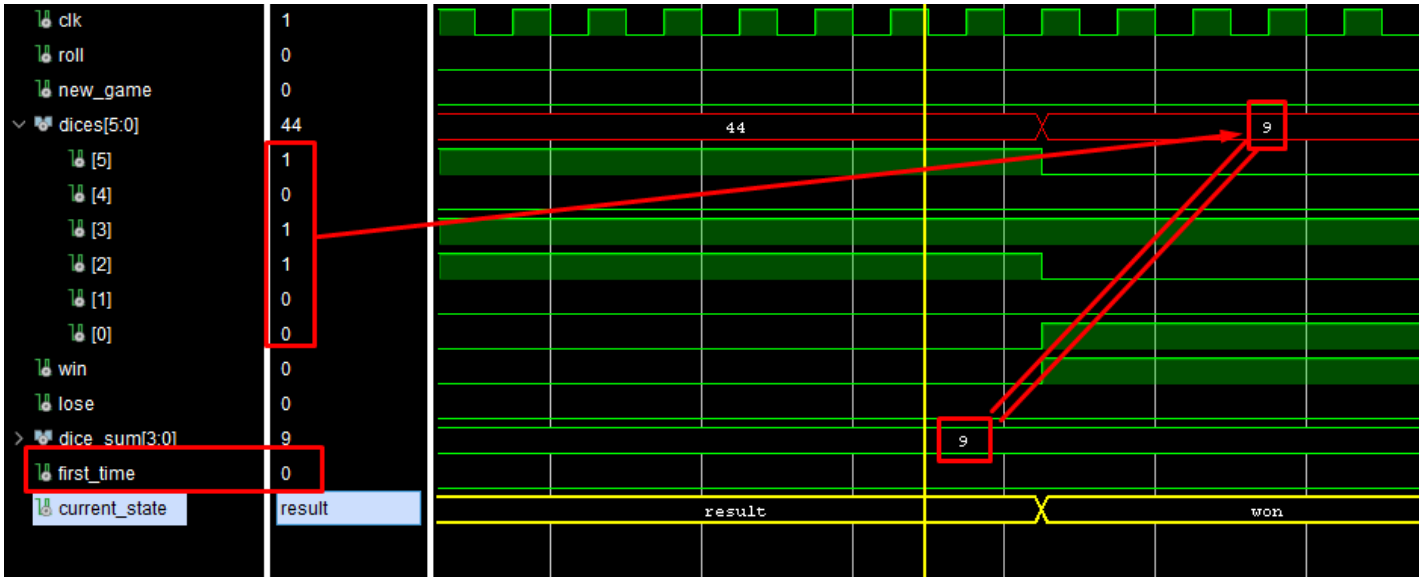
Εικόνα 6: Ήττα κατά την δεύτερη φορά ρίψης των ζαριών

Στην εικόνα 6 βλέπουμε την ήττα ενός παιχνιδιού ρίχνοντας για δεύτερη φορά τα ζάρια. Γνωρίζουμε ότι ο παίκτης δεν ρίχνει για πρώτη φορά τα ζάρια, επειδή το σήμα first_time έχει την τιμή 0. Ο παίκτης έχασε το παιχνίδι εφόσον το άθροισμα των ζαριών είναι 11. Επιπλέον παρατηρούμε ότι ο καταχωρητής dice_sum έχει λάβει την τιμή 9. Αυτό σημαίνει ότι κατά την προηγούμενη ρίψη των ζαριών, ο παίκτης είχε άθροισμα 9.



Εικόνα 7: Αποθήκευση του αθροίσματος των ζαριών κατά την πρώτη ρίψη

Στην εικόνα 7 το άθροισμα των ζαριών αποθηκεύεται στον καταχωρητή dice_sum για την προετοιμασία της επόμενης ρίψης. Ο παίκτης έριξε για πρώτη φορά τα ζάρια. Παρατηρούμε ότι ο παίκτης μεταβαίνει από την κατάσταση Result στην Phase2 λόγω του ότι το άθροισμα των ζαριών δεν κατέστησε κανένα αποτέλεσμα δυνατό.



Εικόνα 8: Νίκη παιχνιδιού εξαιτίας της ισότητας του παρόντος και του αποθηκευμένου αθροίσματος

Σε συνέχεια του παιχνιδιού της εικόνας 7, στην εικόνα 8 βλέπουμε τον παίκτη να νικά. Αυτό συμβαίνει καθώς το άθροισμα που είχε αποθηκευτεί στον dice_sum ισούται με το άθροισμα που έτυχε ο παίκτης σε αυτή την προσπάθεια ρίψης.

Βιβλιογραφία

(1) http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/1999f/Drivers_Ed/lfsr.html

(2) https://www.engineersgarage.com/feed-back-register-in-vhdl/

(3) https://www.nandland.com/vhdl/modules/lfsr-linear-feedback-shift-register.html

ΣΗΜΕΙΩΣΗ

Λόγω του ότι στο top level μοντέλο dice_game χρησιμοποιείται και ο διαιρέτης συχνότητας, θα χρειαστεί να γίνουν κάποιες αλλαγές για να τρέξει ορθά το test bench. Οι αλλαγές αυτές είναι: i) αλλαγή του div_clk σε clk στα *counter1*, *counter2* και *res_counter* processes και ii) αλλαγή του div_clk σε clk στο *dices_output* process του dices_generator μοντέλου.