

# FPGA dice game

Ομαδική εργασία μαθήματος Σχεδίαση Υπολογιστικών Συστημάτων

Π16090: Andi Besimi | [besimiandi@gmail.com](mailto:besimiandi@gmail.com)

Π16126: Δημήτρης Σερπάνος | [dimitris\\_serpanos@yahoo.gr](mailto:dimitris_serpanos@yahoo.gr)

Π16040: Μιχάλης Καλλιάφας | [mikekalliafas2@gmail.com](mailto:mikekalliafas2@gmail.com)

## Περιεχόμενα

State Machine .....	2
Σύντομη περιγραφή του κώδικα και της σχεδίασης .....	5
Πως κρατάμε τις τιμές των ζαριών την χρονική στιγμή που θέλουμε; .....	5
Αποτελέσματα από simulation .....	6

## State Machine

Η μηχανή καταστάσεων συνολικά αποτελείται από 14 καταστάσεις. Οι λειτουργίες της κάθε κατάστασης είναι οι εξής:

**Phase 1:** Οι έξοδοι είναι κενές. Αναμέται το πάτημα του roll κουμπιού. Όταν το roll γίνει 1 τότε πηγαίνουμε στην κατάσταση Rolling 1.

**Rolling1:** Όσο είναι πατημένο το roll παραμένουμε σ αυτήν την κατάσταση. Στις εξόδους βλέπουμε τυχαίους αριθμούς να εναλλάσσονται. Όταν το roll γίνει 0 τότε περνάμε στην κατάσταση StopRolling1.

**StopRolling1:** Το roll κουμπί έγινε 0. Ένα εσωτερικό σήμα στέλνεται για να ξεκινήσει ο μετρητής την αντιστροφή μέτρηση ώστε να σταματήσουν τα ζάρια να αλλάζουν τιμές. Όταν ο μετρητής γίνει 0 τότε οι τιμές των ζαριών “κλειδώνουν”, το άθροισμα τους υπολογίζεται και συνεχίζουμε στην κατάσταση Result1.

**Result1:** Τα ζάρια έχουν σταματήσει και το άθροισμα έχει υπολογισθεί. Ακόμη ένα εσωτερικό σήμα στέλνεται για να αρχίσει ο μετρητής ο οποίος είναι υπεύθυνος για το πόση ώρα θα προβάλλονται οι τιμές των ζαριών στον χρήστη. Όταν ο μετρητής γίνει 0 τότε προχωράμε στην κατάσταση Phase2.

**Phase2:** Σε αυτήν την κατάσταση προβάλλεται το άθροισμα των ζαριών και γίνεται ο έλεγχος για το αν ο χρήστης κέρδισε ή έχασε. Αν κέρδισε θα μεταβούμε στην κατάσταση win1, αν έχασε στην κατάσταση lose1 και αν δεν γίνει τίποτα από τα δυο θα πάμε στην κατάσταση save.

**Win1:** Το άθροισμα των ζαριών είναι νικητήριο. Το λαμπάκι της νίκης έχει ανάψει και ακόμα προβάλλεται το άθροισμα των ζαριών. Ο χρήστης μπορεί να πατήσει μόνο newgame για να μεταβεί στην κατάσταση phase1.

**Lose1:** Το άθροισμα των ζαριών δεν είναι νικητήριο. Το λαμπάκι της ήττας έχει ανάψει και ακόμα προβάλλεται το άθροισμα των ζαριών. Ο χρήστης μπορεί να πατήσει μόνο newgame για να μεταβεί στην κατάσταση phase1.

**Save:** Το άθροισμα των ζαριών δεν κερδίζει ούτε χάνει. Ένα εσωτερικό σήμα στέλνεται για να γίνει η αποθήκευση του αθροίσματος των ζαριών και κατευθείαν θα συνεχίσουμε στην κατάσταση Rolling2.

**Rolling2:** Όσο είναι πατημένο το roll παραμένουμε σ αυτήν την κατάσταση. Στις εξόδους βλέπουμε τυχαίους αριθμούς να εναλλάσσονται. Όταν το roll γίνει 0 τότε περνάμε στην κατάσταση StopRolling2.

**StopRolling2:** Το roll κουμπί έγινε 0. Ένα εσωτερικό σήμα στέλνεται για να ξεκινήσει ο μετρητής την αντιστροφή μέτρηση ώστε να σταματήσουν τα ζάρια να αλλάζουν τιμές. Όταν ο μετρητής γίνει 0 τότε οι τιμές των ζαριών “κλειδώνουν”, το άθροισμα τους υπολογίζεται και συνεχίζουμε στην κατάσταση Result2.

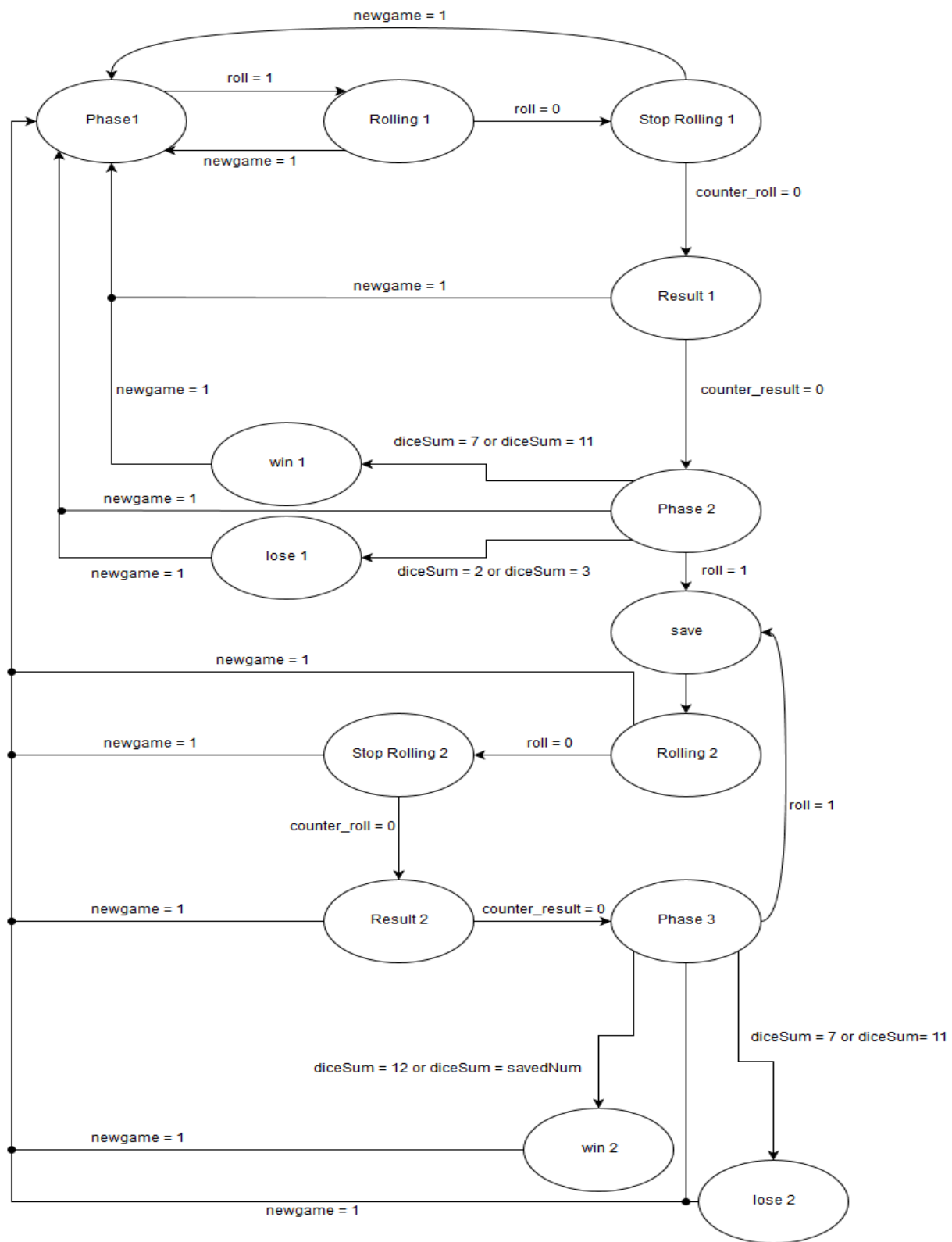
**Result2:** Τα ζάρια έχουν σταματήσει και το άθροισμα έχει υπολογισθεί. Ακόμη ένα εσωτερικό σήμα στέλνεται για να αρχίσει ο μετρητής ο οποίος είναι υπεύθυνος για το πόση ώρα θα προβάλλονται οι τιμές των ζαριών στον χρήστη. Όταν ο μετρητής γίνει 0 τότε προχωράμε στην κατάσταση Phase3.

**Phase3:** Σε αυτήν την κατάσταση προβάλλεται το άθροισμα των ζαριών και γίνεται ο έλεγχος για το αν ο χρήστης κέρδισε ή έχασε. Αν κέρδισε θα μεταβούμε στην κατάσταση win2, αν έχασε στην κατάσταση lose2 και αν δεν γίνει τίποτα από τα δυο θα πάμε στην κατάσταση save. Η μόνη διαφορά αυτής της κατάστασης από την phase 2 είναι ότι σε αυτήν την κατάσταση οι συνθήκες νίκης και ήττας είναι διαφορετικές.

**Win2:** Το άθροισμα των ζαριών είναι νικητήριο. Το λαμπάκι της νίκης έχει ανάψει και ακόμα προβάλλεται το άθροισμα των ζαριών. Ο χρήστης μπορεί να πατήσει μόνο newgame για να μεταβεί στην κατάσταση phase1.

**Lose2:** Το άθροισμα των ζαριών δεν είναι νικητήριο. Το λαμπάκι της ήττας έχει ανάψει και ακόμα προβάλλεται το άθροισμα των ζαριών. Ο χρήστης μπορεί να πατήσει μόνο newgame για να μεταβεί στην κατάσταση phase1.

Στις καταστάσεις Rolling1, Result1, Phase2, Win1, Lose1, Rolling2, Result2, Phase3, Win2 και Lose2 όταν ο χρήστης πατήσει το κουμπί newgame τότε θα μεταβούμε στην κατάσταση Phase1. Στην Εικόνα 1 απεικονίζεται το State Machine.



Εικόνα 1: State machine diagram

## Σύντομη περιγραφή του κώδικα και της σχεδίασης

Για την λειτουργία του παιχνιδιού χρησιμοποιούμε 2 υπό-units το random\_bit\_generator και το clock divider και το βασικό unit που περιέχει αυτά τα δυο υπό-units.

Η λειτουργία του random\_bit\_generator είναι απλή, υπάρχει μια μεταβλητή count η οποία είναι 16 bits και σε κάθε χτύπο του ρολογιού(125MHz) αυξάνεται κατά 1. Σε κάθε χτύπο του διαιρεμένου ρολογιού(2Hz) περνάμε τυχαία bits (τα οποία έχουν περάσει από πύλες XOR) στις εξόδους dice1 και dice2 τις οποίες τις συνδέουμε στο βασικό μας unit. Σε περίπτωση που τα τυχαία bits που θα πάρουμε βγάλουν τον αριθμό 0 ή 7, τότε αλλάζουμε τις τιμές σε 1 και 6 αντίστοιχα.

Το βασικό unit, που περιέχει και το state machine, έχει 7 processes. Τα 3 από αυτά αφορούν το state machine. Από τα υπόλοιπα 4, τα 2 υλοποιούν τους counters (display\_count και rolling\_after\_stop), το ένα αποθηκεύει το άθροισμα των ζαριών όταν χρειάζεται και το τελευταίο “κλειδώνει” την τιμή στα ζάρια όταν σταματάνε.

### Πως κρατάμε τις τιμές των ζαριών την χρονική στιγμή που θέλουμε;

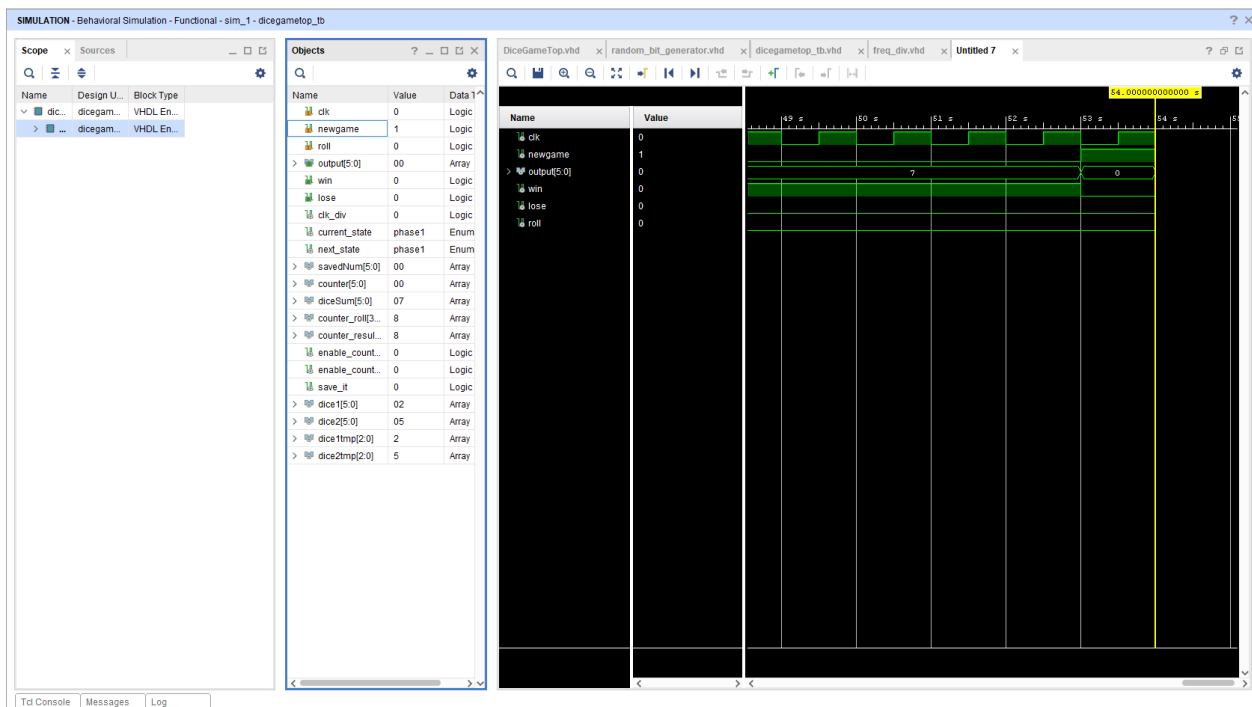
Για να μπορέσουμε να κρατήσουμε τις τιμές των ζαριών την στιγμή που θα σταματήσει ο μετρητής αντίστροφης μέτρησης, χρησιμοποιήσαμε 2 βοηθητικές μεταβλητές (τις dice1tmp και dice2tmp) οι οποίες παίρνουν τις εξόδους του random\_bit\_generator unit. Όταν από την κατάσταση stoprolling σταλθεί το εσωτερικό σήμα(enable\_counter\_roll) για να αρχίσει ο μετρητής παύσης των ζαριών, τότε εκτελείται και το lock\_dice process το οποίο γράφει τις τιμές των dice1tmp και dice2tmp στις dice1 και dice2, και ταυτόχρονα υπολογίζει και το άθροισμα τους. Όταν σταματήσει ο μετρητής αντίστροφης μέτρησης τότε το εσωτερικό σήμα γίνεται 0 και το lock\_dice process “σταματάει”(Λόγω εσωτερικής if, δεν μπορεί να εκτελέσει το κομμάτι που αναθέτει νέες τιμές στα dice1 dice2). Έτσι στα dice1 και dice2 έχουμε τις τελευταίες τιμές που προβλήθηκαν στον χρήστη.

```
lock_dice:process (enable_counter_roll)
begin
    if enable_counter_roll = '1' then
        dice1 <= "000"&dice1tmp;
        dice2 <= "000"&dice2tmp;
        diceSum <= dice1 + dice2;
    end if;
end process lock_dice;
```

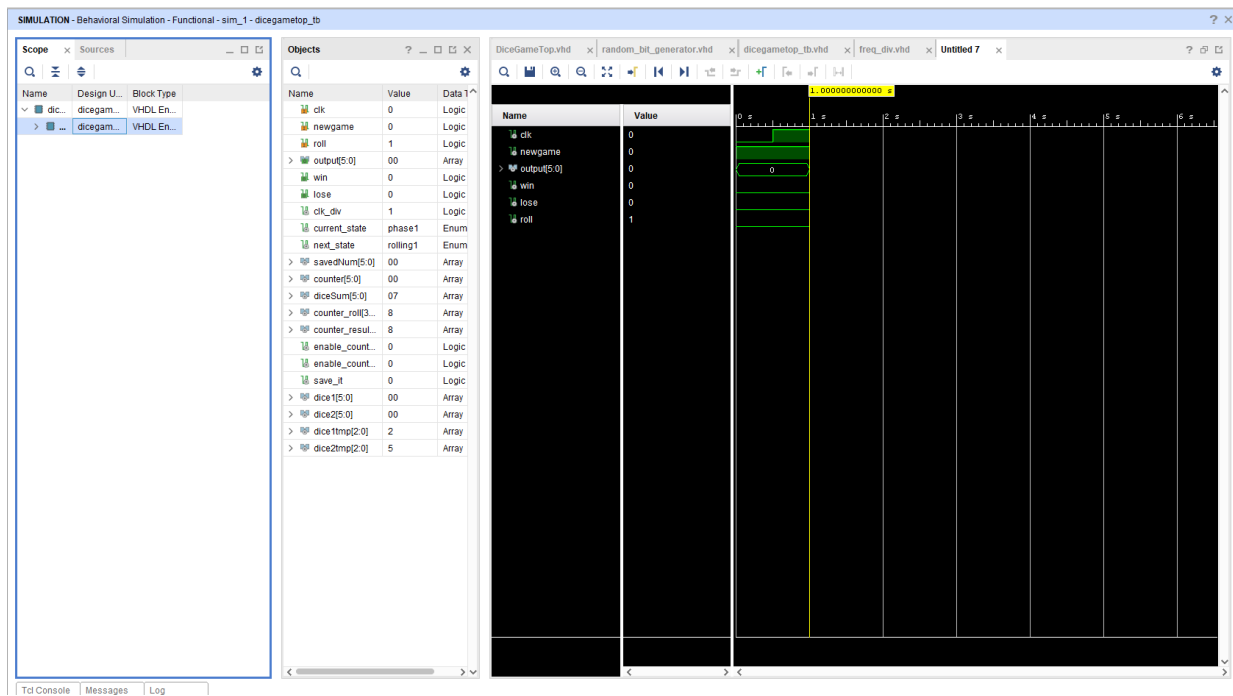
Εικόνα 2: lock\_dice process

## Αποτελέσματα από simulation

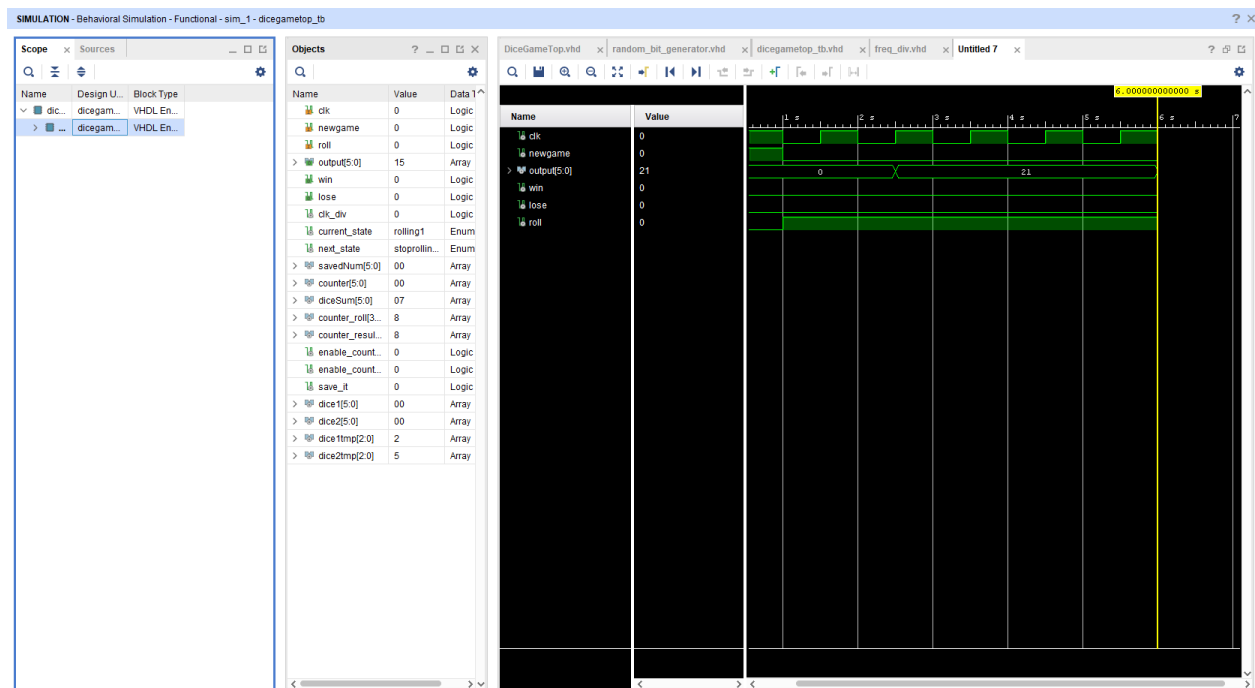
Παρακάτω απεικονίζονται κομμάτια του simulation από διάφορες φάσης και καταστάσεις του προγράμματος(Όλες οι παρακάτω εικόνες βρίσκονται και στον φάκελο Images).



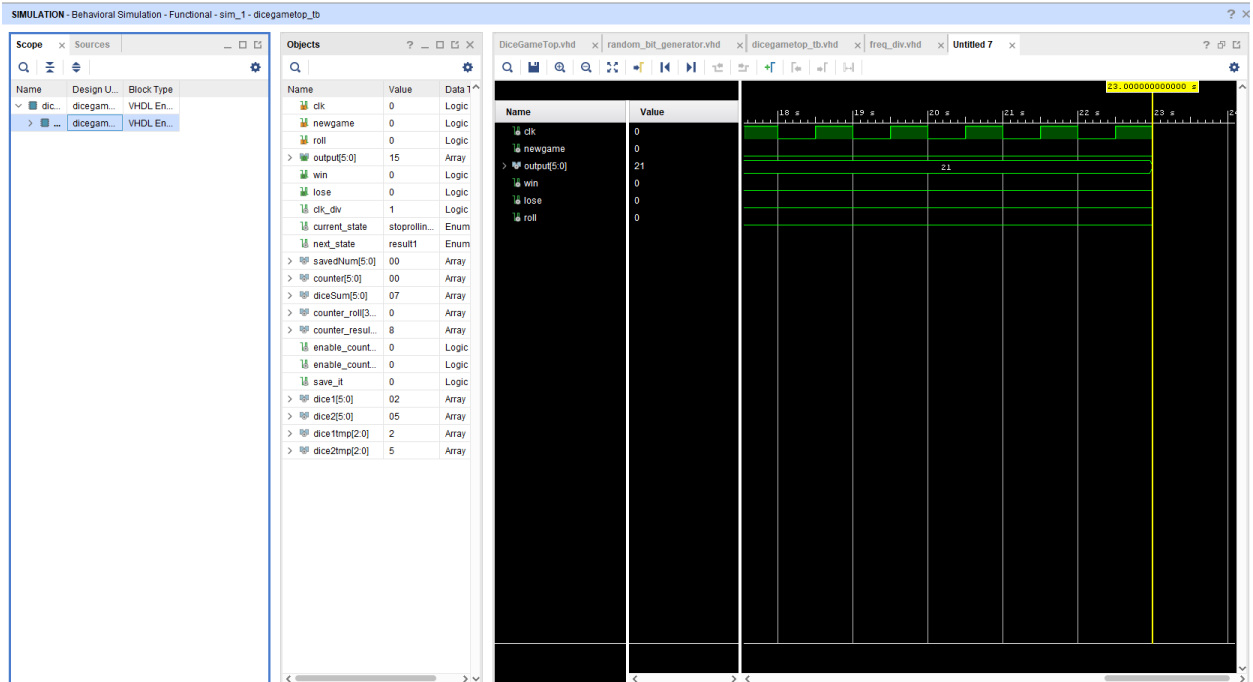
α. Phase 1. Αναμένεται είσοδος



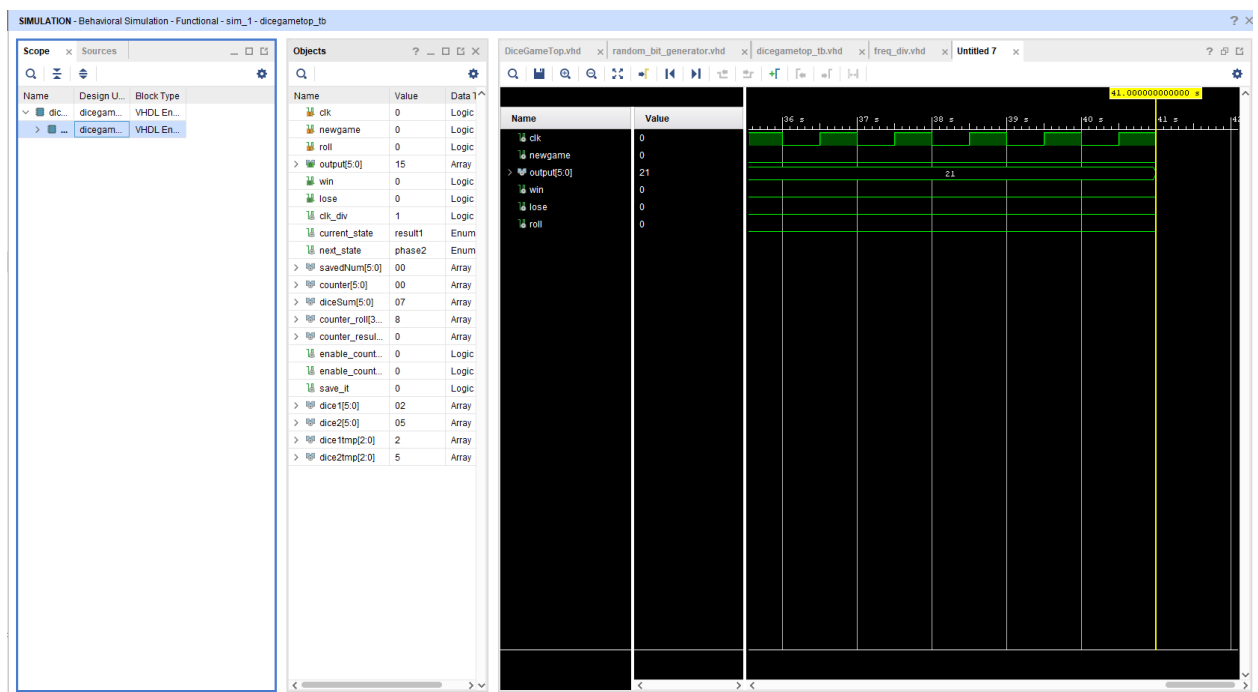
b. Phase 1 γίνεται η μετάβαση στην κατάσταση rolling



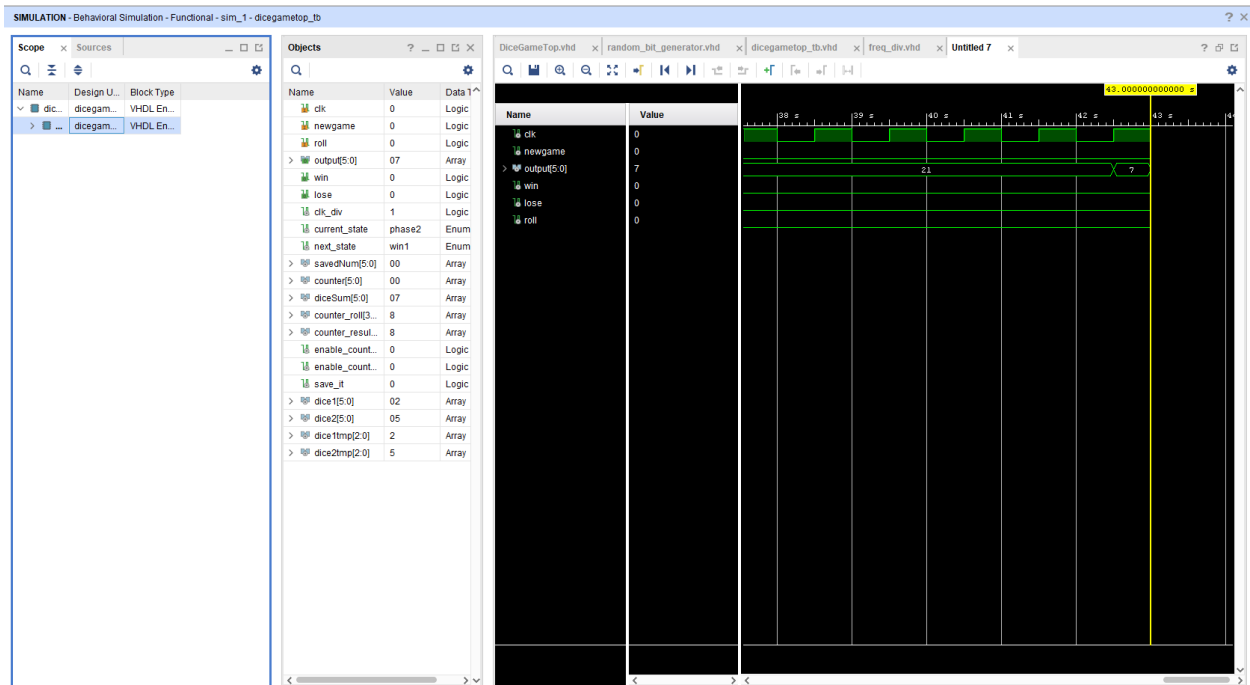
c. Rolling 1, το roll είναι πατημένο



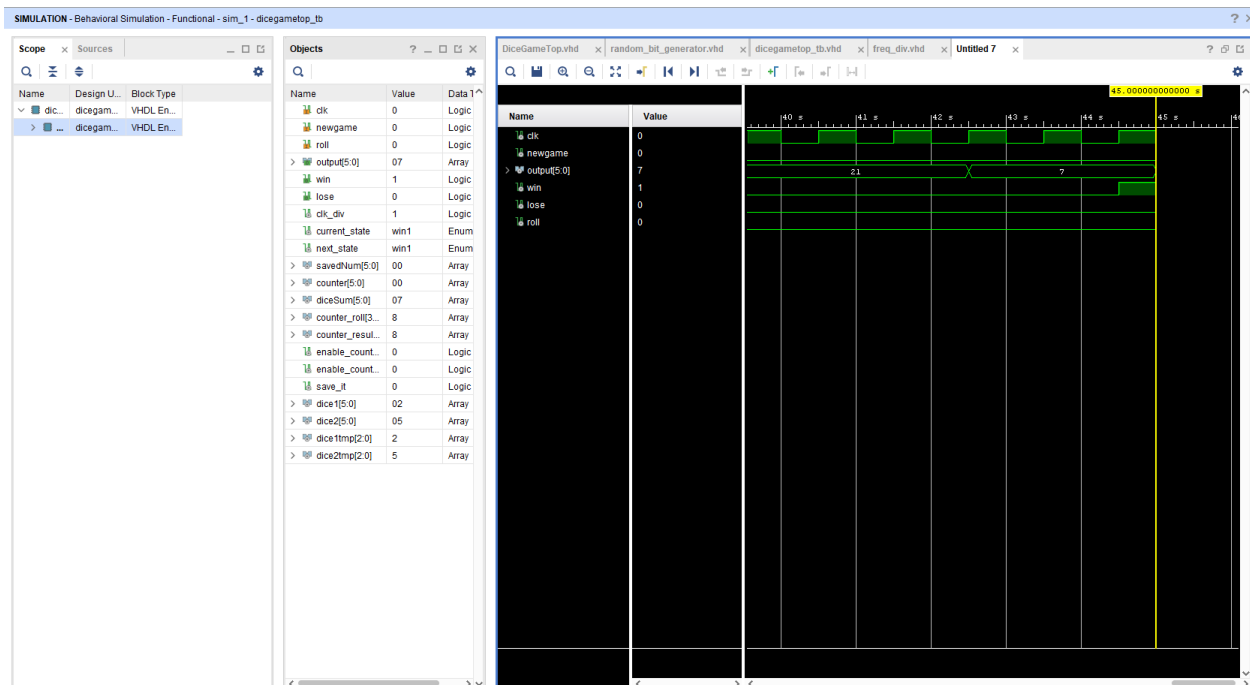
#### d. StopRolling1



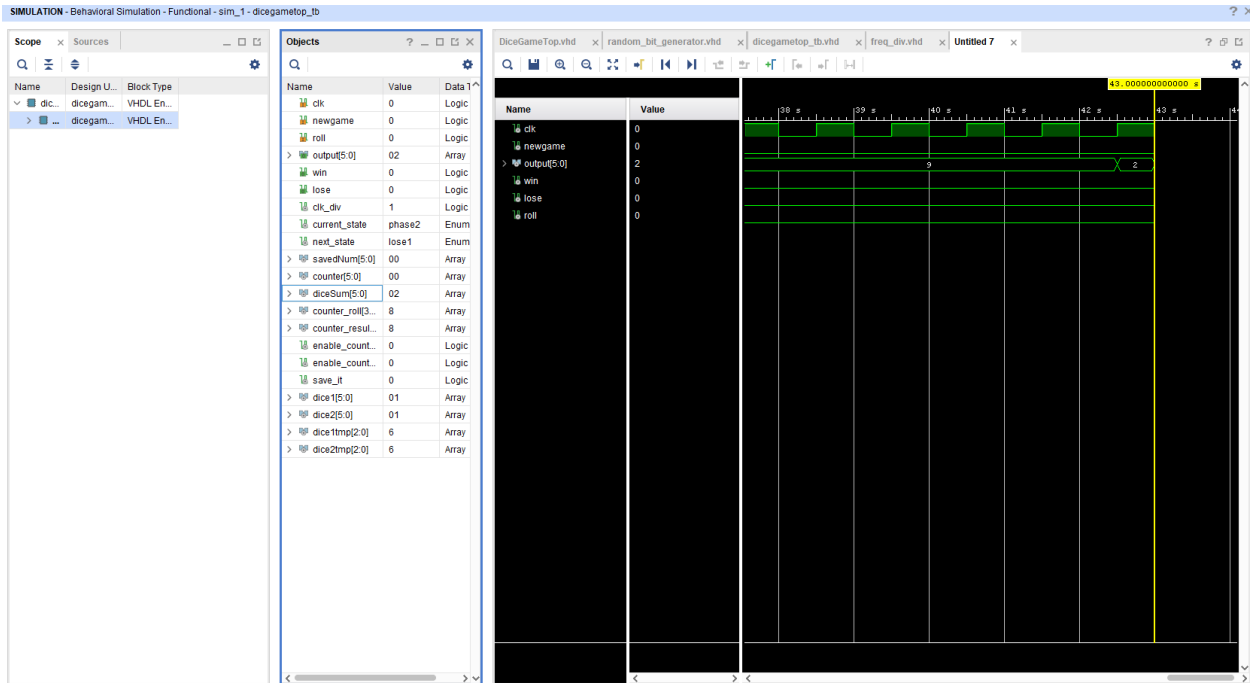
e. Result1



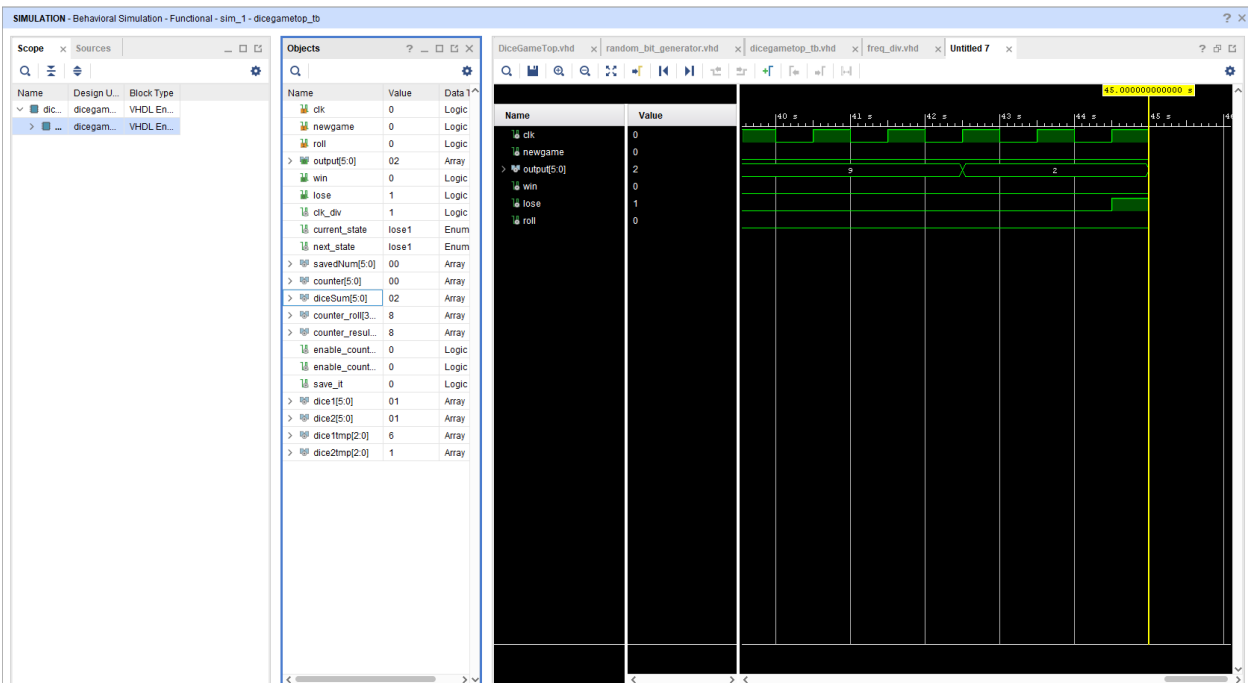
f. Phase 2, το άθροισμα των ζαριών είναι 7. Επόμενη κατάσταση η win 1.



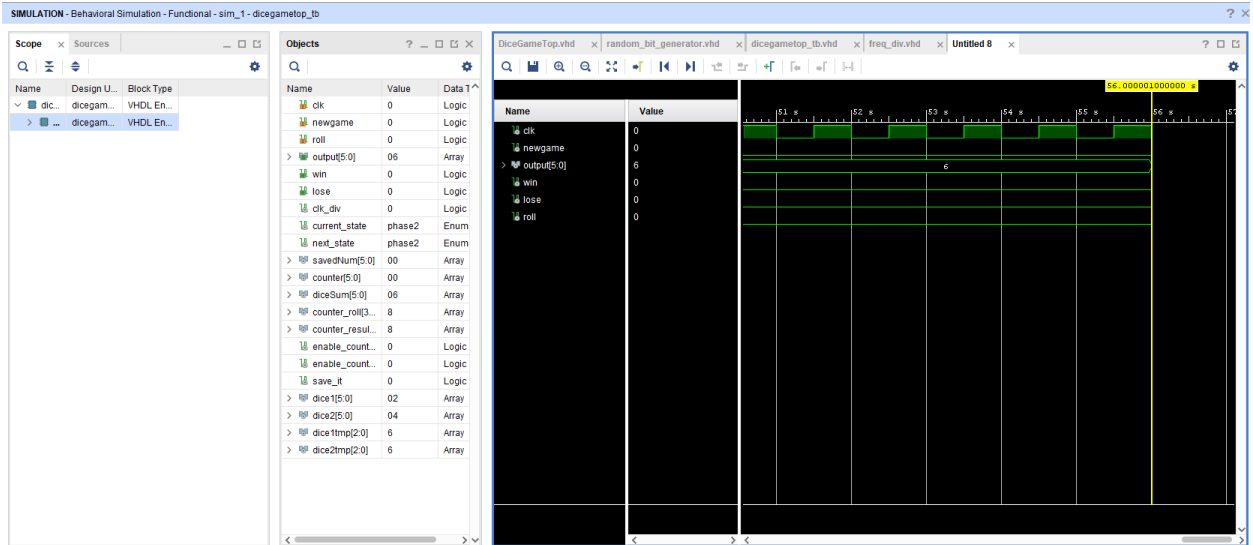
g. Win 1, παραμένουμε σε αυτήν την κατάσταση μέχρι να πατηθεί το newgame.



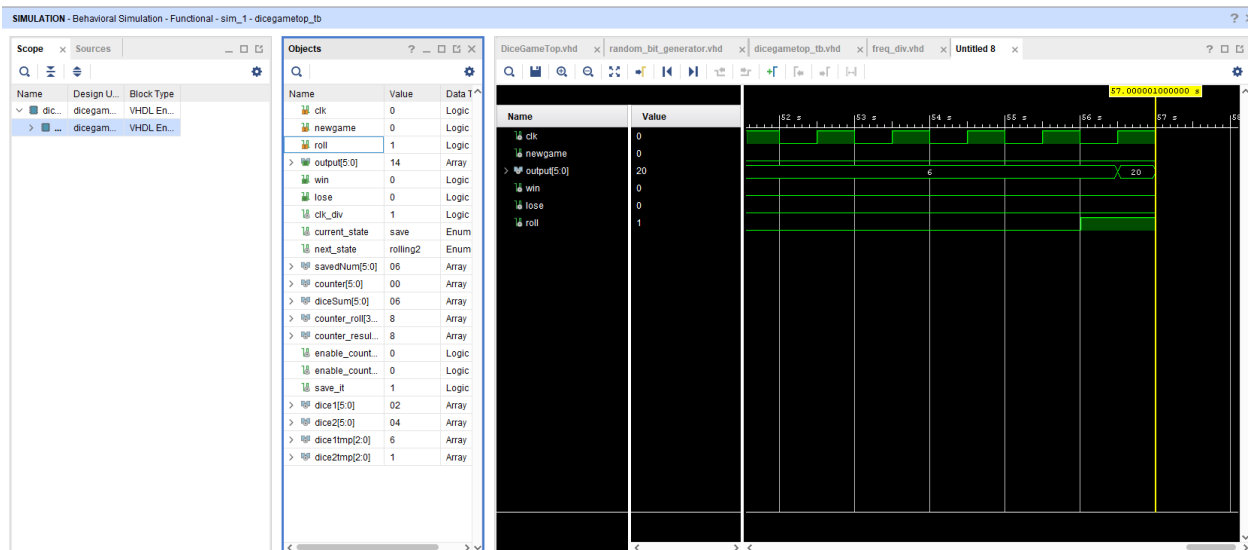
h. Phase2, το άθροισμα είναι 2. Επόμενη κατάσταση lose1



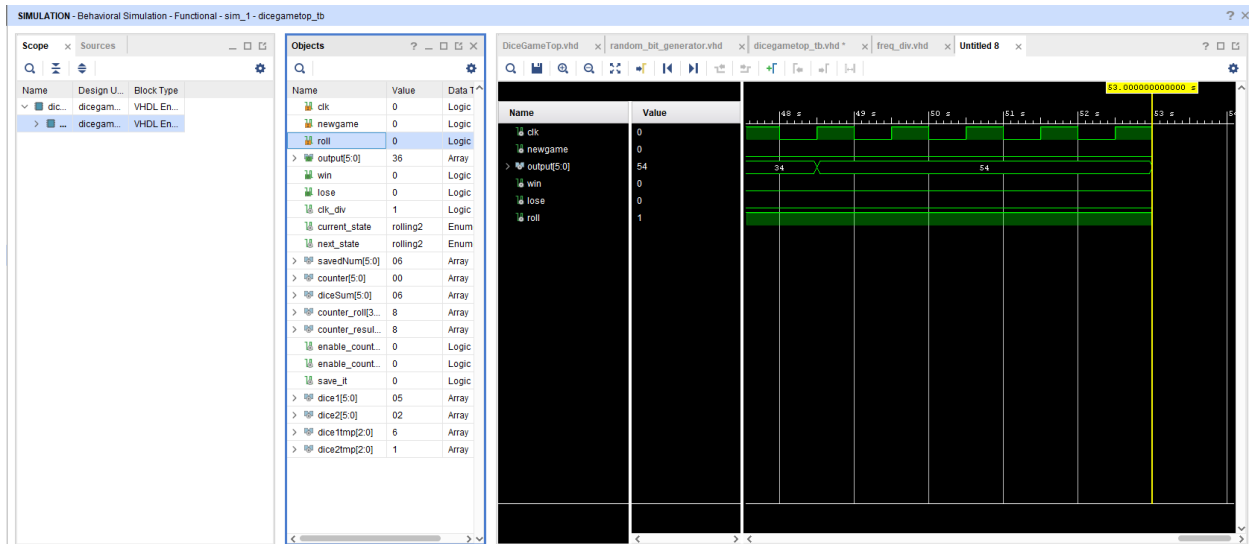
i. Lose 1, παραμένουμε σε αυτήν την κατάσταση μέχρι να πατηθεί το newgame.



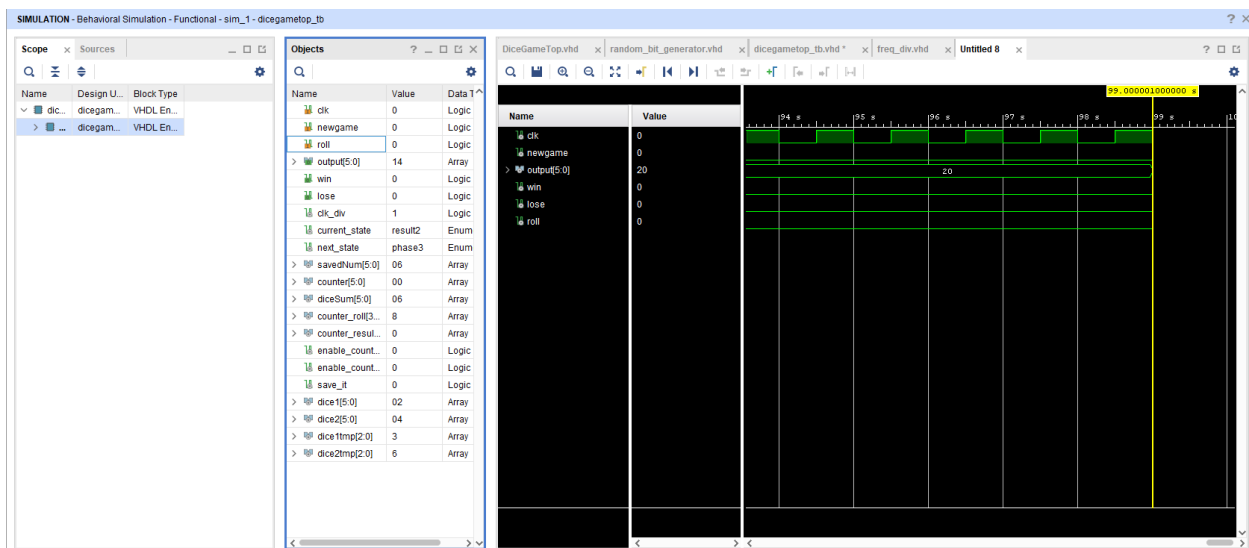
j. Phase 2, το άθροισμα δεν κερδίζει ούτε χάνει. Αναμένεται κάποια είσοδος.



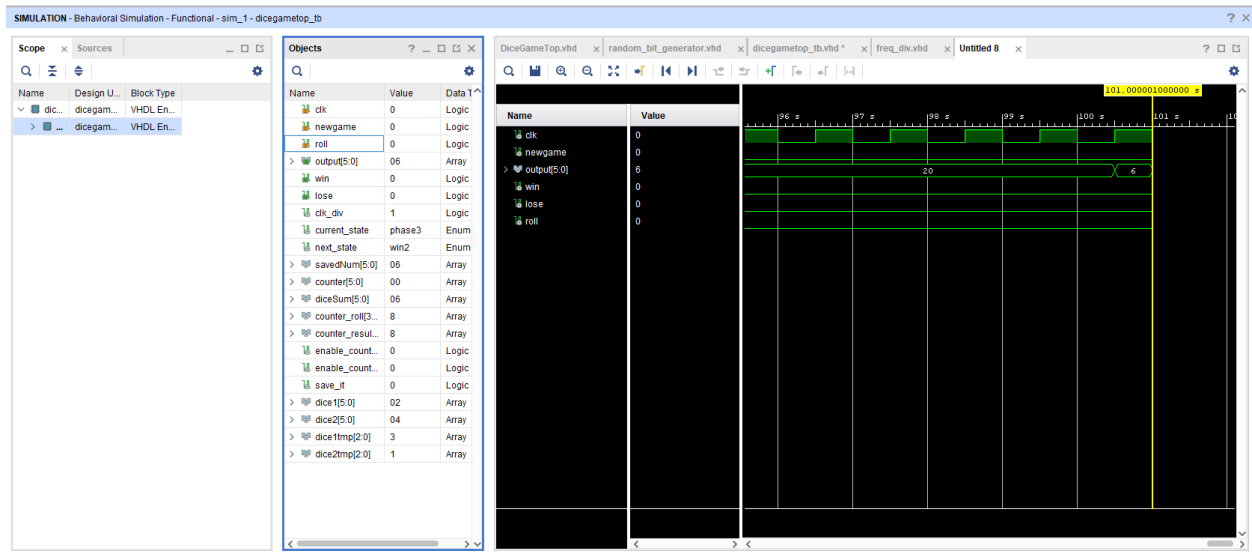
k. Save, αποθηκεύουμε την τιμή 6 από την κατάσταση phase 2



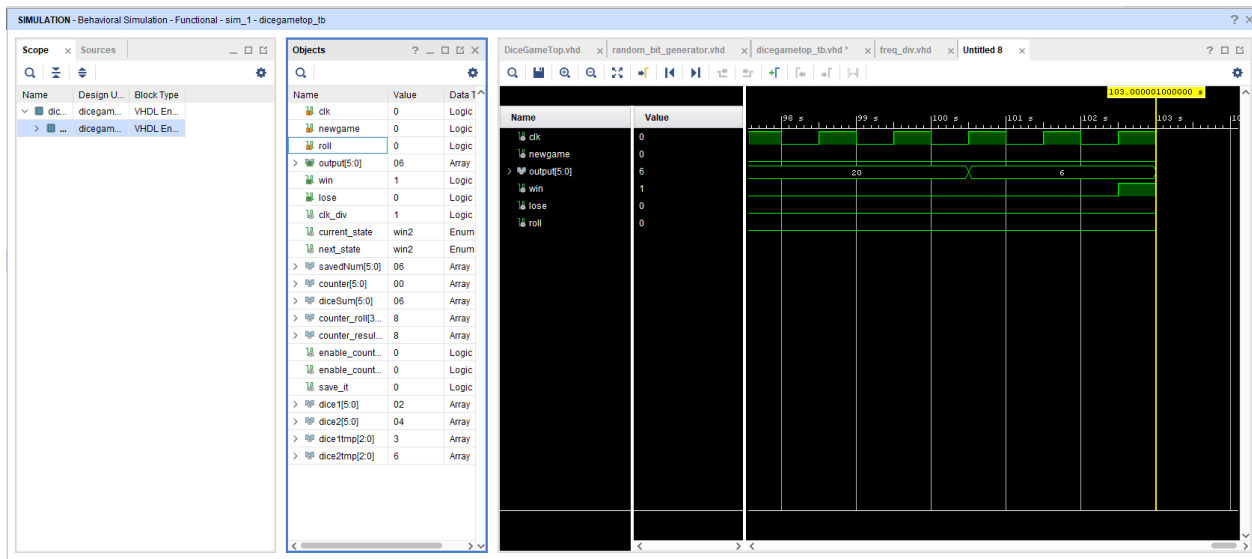
l. Rolling2 με αποθηκευμένο αριθμό το 6



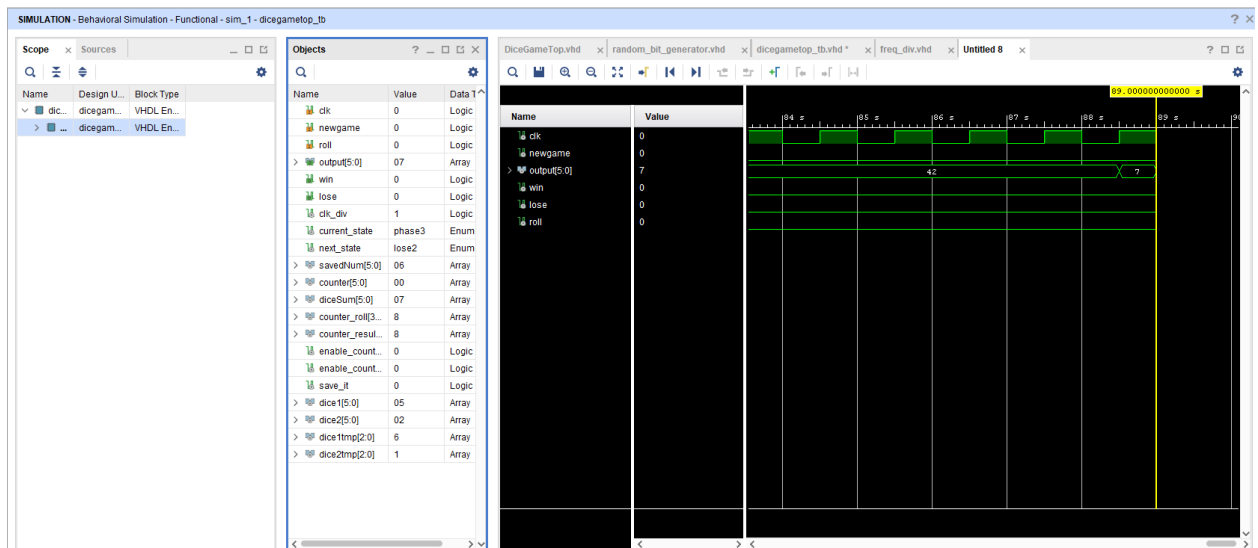
m. Result 2



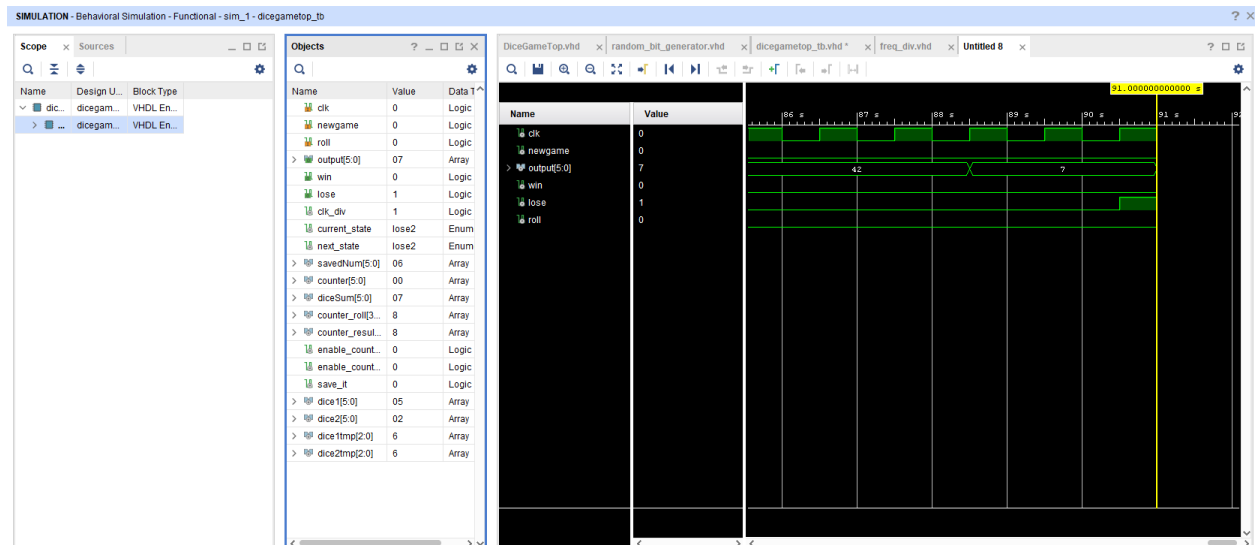
n. Phase 3, το άθροισμα είναι 6 και ο αποθηκευμένος αριθμός είναι 6.



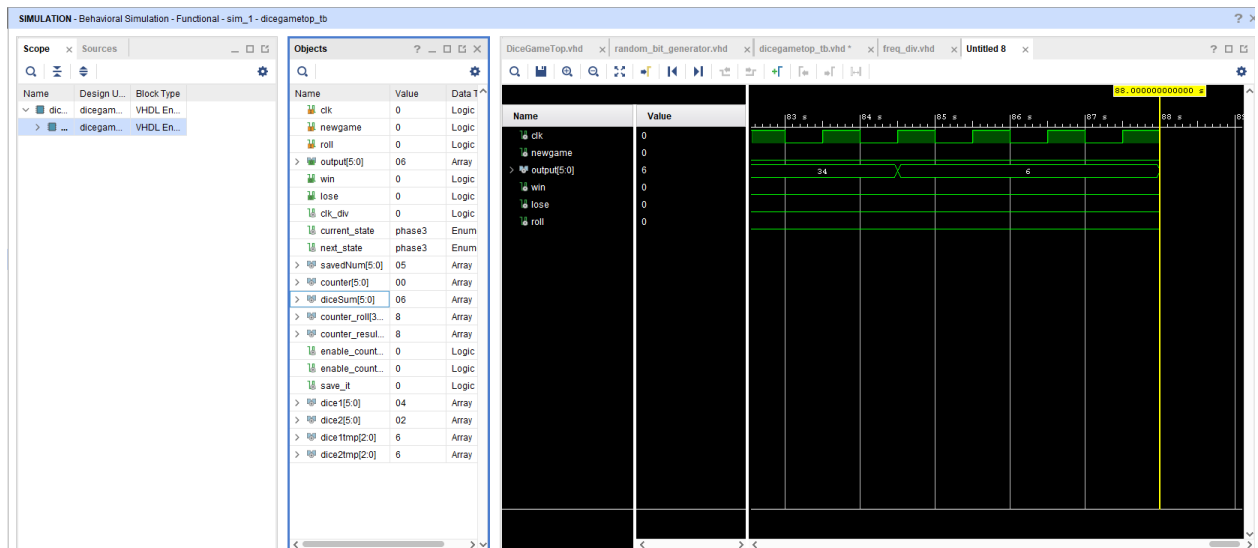
o. Win2



ρ. Phase3, ο αποθηκευμένος αριθμός είναι το 6 αλλά τα ζάρια έφεραν 7.



q. Lose 2



*r.Phase 3, αποθηκευμένος αριθμός το 5. Αναμένεται κάποια είσοδος*