**Εικόνα που περιέχει κείμενο, δωμάτιο, καζίνο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Πανεπιστήμιο Ιωαννίνων**

**Πολυτεχνική Σχολή**

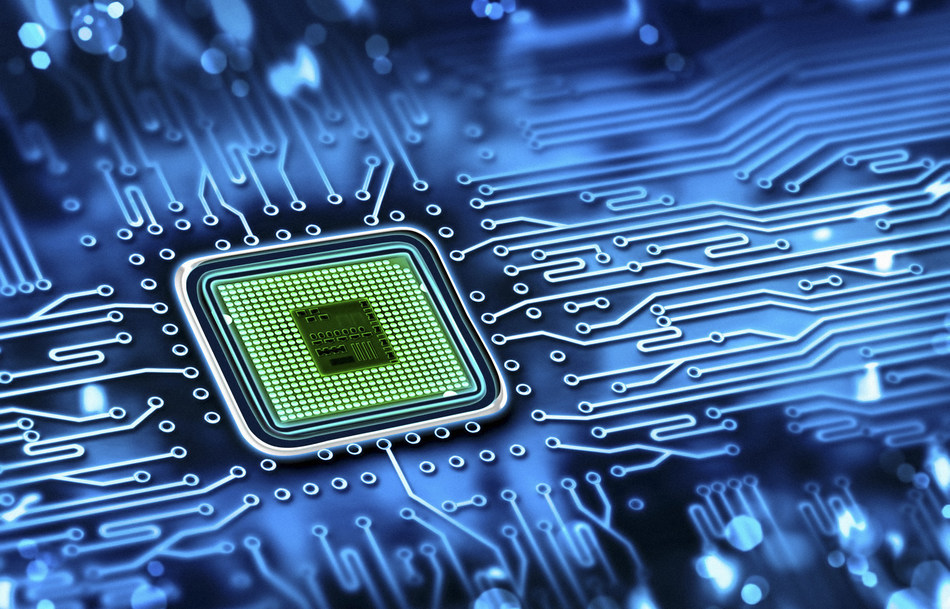
**Τμήμα Μηχανικών Η/Υ και Πληροφορικής**

**Προπτυχιακό Μάθημα: «Δοκιμή και Αξιοπιστία Ηλεκτρονικών Συστημάτων»**

**Πρώτη Εργαστηριακή Άσκηση**

**Όνομα Φοιτητή – Α.Μ.:**

**Γεώργιος Κρομμύδας – 3260**

****

*ΙΩΑΝΝΙΝΑ,*

*2021*

Πίνακας περιεχομένων

[Μέρος – Α: Περιγραφή Εργασίας: 3](#_Toc68179561)

[Μέρος – Β: Υλοποίηση Κυκλώματος: 3](#_Toc68179562)

[Μέρος – Γ: Θεωρητική Άσκηση: 11](#_Toc68179563)

[APPENDIX: 13](#_Toc68179564)

[Άσκηση-1.1: 13](#_Toc68179565)

[Άσκηση-1.2: 18](#_Toc68179566)

# Μέρος – Α: Περιγραφή Εργασίας:

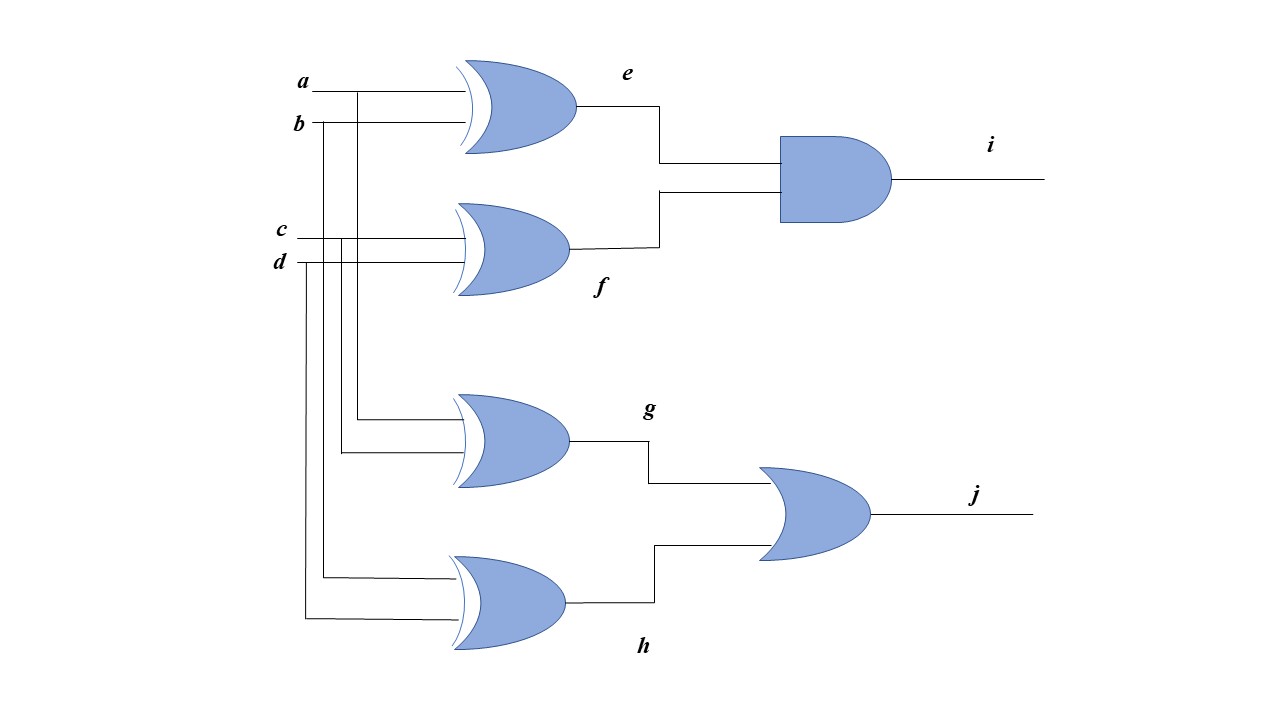
Σε αυτή την εργασία είχαμε να υλοποιήσουμε ένα κύκλωμα **TRCUT(Testable-Ready Circuit Under Test)** με την μέθοδο της σειριακής σάρωσης. Με διάφορα διανύσματα δοκιμής τα οποία εισάγονται σειριακά μέσω του κυκλώματος **Scan Chain**, προσπαθούμε να ελέγξουμε την λειτουργία του συνδυαστικού κυκλώματος. Ο έλεγχος γίνεται βάσει του πίνακα αλήθειας και τις σωστές αποκρίσεις του.

# Μέρος – Β: Υλοποίηση Κυκλώματος:

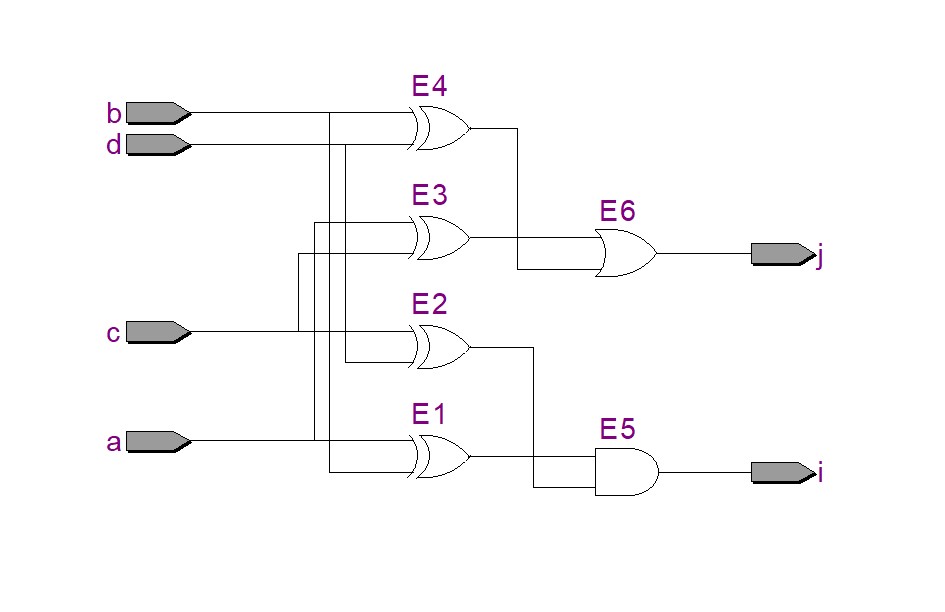
Το πρώτο κομμάτι του κυκλώματος είναι το συνδυαστικό μέρος **CUT**(**Circuit Under Test**) το οποίο αποτελείται από έξι λογικές πύλες. Έχει τέσσερις **xor**, μία **and** και μία **or** και ο τύποι του είναι οι εξής:

.

Το κύκλωμα φαίνεται παρακάτω φαίνεται παρακάτω:



Χρησιμοποιώντας **RTL** περιγραφή η οποία φαίνεται στο κεφάλαιο **APPENDIX**, με τον κώδικα **Κ1** σχεδιάστηκε το παραπάνω συνδυαστικό κύκλωμα το οποίο αποτελείται από τέσσερις εισόδους **a**, **b**, **c** και **d**, καθώς και με δύο εξόδους τις **i** και **j**. Οι είσοδοι του κυκλώματος αρχικά ορίζονται και ως **wires** για να σχηματιστούν και τα καλώδια μεταξύ των πυλών. Αρχικά, ορίζονται οι πύλες **xor**.Με την εντολή η οποία αρχικοποιεί την πρώτη πύλη με έξοδο **e** και εισόδους **a** και **b**, όπου το **E1(Element 1)** αποτελεί το πρώτο στοιχείο του κυκλώματος. Αντίστοιχα ορίζονται και οι υπόλοιπες τρεις πύλες **xor** και δημιουργείται το ενδιάμεσο επίπεδο που αποτελείται από τις τέσσερις εξόδους των πυλών **e**, **f**, **g** και **h**. Αυτές ανά δύο συνδέονται σε δύο πύλες **and** και **or** αντίστοιχα, οι οποίες παράγουν την τελική έξοδο του κυκλώματος που αποτελείται από τα **i**,**j**. Παρακάτω φαίνεται και η επαλήθευση σχεδιασμού από το quartus.



Το επόμενο στάδιο ήταν η επαλήθευση του παραπάνω σχεδιασμού, χρησιμοποιώντας ένα testbench. Ο κώδικας, του testbench είναι ο κώδικας **Κ2**, ο οποίος επαληθεύει τον πίνακα αλήθειας του παραπάνω κυκλώματος. Αρχικά, δημιουργήθηκαν τέσσερις καταχωρητές οι οποίες είναι οι είσοδοι του κυκλώματος και παίρνουν τιμές.Η εγγραφή των εισόδων γίνεται μέσω της εντολής **wire [3:0] tb\_cut\_inputs**, όπου θα αναπαρίστανται οι τιμές του πίνακα αλήθειας. Στη συνέχεια, καλείται το συνδυαστικό κύκλωμα με την εντολή

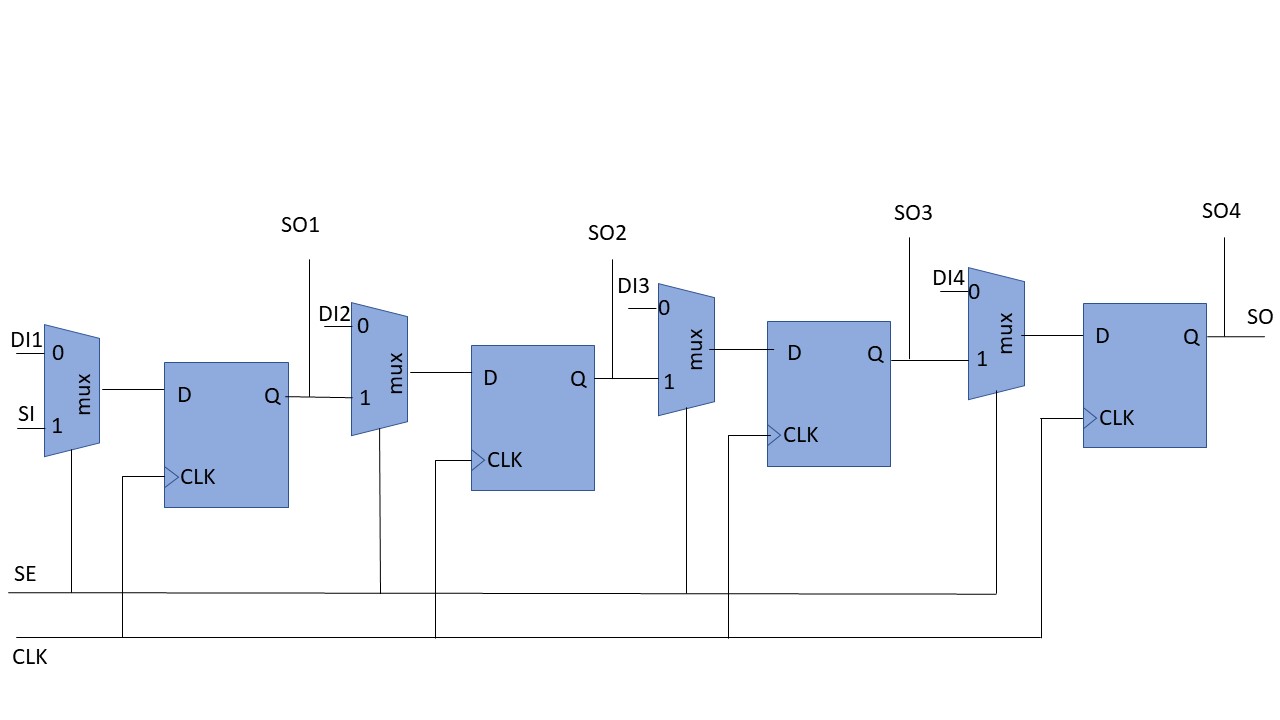
όπου θα πάρει ως είσοδο τους καταχωρητές και ως έξοδο τα **wires tb\_i**, **tb\_j**. Έπειτα, αρχικοποιούμε με εντολή **assign** τον δίαυλο των εισόδων **tb\_cut\_inputs** με τους καταχωρητές εισόδου. Στη συνέχεια, δημιουργείται ένα **forever loop** στο οποίο αυξάνεται η τιμή του **tb\_cut\_inputs** κατά ένα, έτσι ώστε να πάρουμε όλες τις τιμές του πίνακα αλήθειας. Η τιμή θα αυξάνεται κάθε είκοσι μονάδες χρόνου και θα έχουμε την σωστή απόκριση από το κύκλωμα που περιμένουμε. Παρακάτω φαίνεται η κυματομορφή:

Εικόνα που περιέχει κείμενο, πίνακας αποτελεσμάτων

Περιγραφή που δημιουργήθηκε αυτόματα

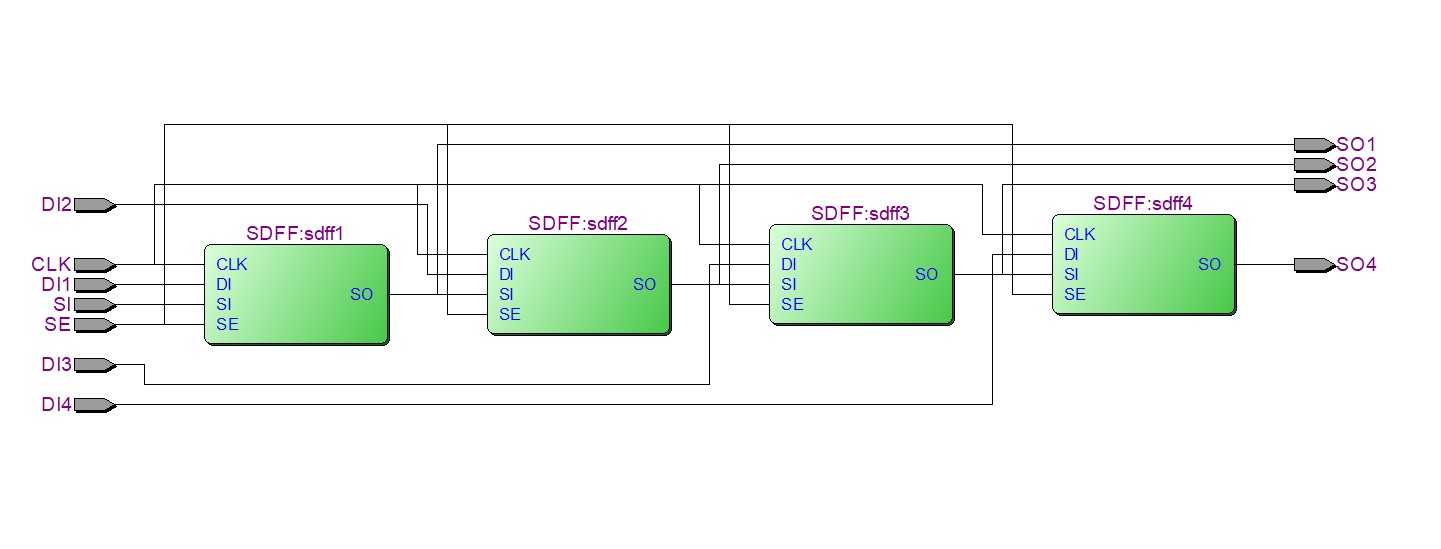
Παρατηρούμε ότι οι αποκρίσεις του κυκλώματος είναι σωστές και αυτές που περιμένουμε κατά την εκτέλεση. Για παράδειγμα, εάν έχουμε ως είσοδο την τιμή **<0110>**, τότε ως έξοδο θα πρέπει να έχουμε την τιμή **<11>**, η οποία φαίνεται καθαρά στην κυματομορφή.

Το δεύτερο κομμάτι του κυκλώματος είναι η ακολουθιακή αλυσίδα **Scan Chain**, η οποία αποτελείται από τέσσερα **Scan D Flip-Flop**, που είναι συνδεδεμένα σειριακά όπως φαίνεται στο κύκλωμα παρακάτω:



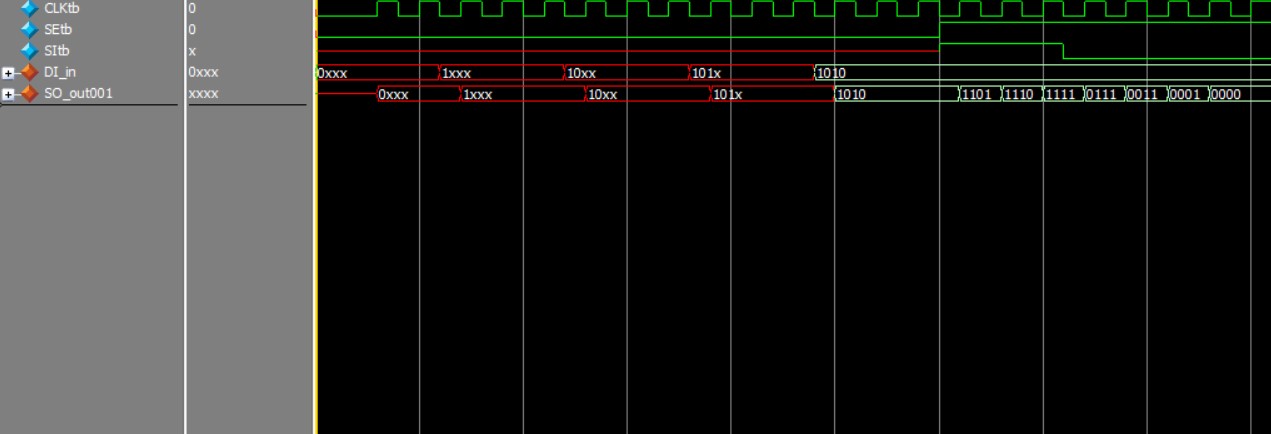
Γενικά, το **Scan D Flip-Flop** αποτελείται από έναν πολυπλέκτη **MUX 2-1** και από ένα  **D Flip-Flop**. Ο πολυπλέκτης έχει ως είσοδο είτε τα δεδομένα από την έξοδο του κυκλώματος **DI(Data Input)** είτε δεδομένα από την είσοδο **SI(Serial Input)** όπου εισάγεται ακολουθία από **bit** σειριακά. Για να αποφασιστεί ποια είσοδο θα επιλέξει ο πολυπλέκτης, ελέγχει την τιμή του **SE(Scan Enable)**. Αν η τιμή του είναι μηδέν τότε ως είσοδο παίρνει τα δεδομένα από την συνδυαστική λογική. Σε αντίθετη περίπτωση εισάγει τα δεδομένα της σειριακής εισόδου. Η έξοδος του πολυπλέκτη, στη συνέχεια αποτελεί είσοδο του **D Flip-Flop**, όπου αποθηκεύεται η τιμή που επιλέχθηκε σύμφωνα με την θετική ακμή του ρολογιού **CLK**. Η έξοδος του **D Flip-Flop** θα συνδεθεί με μία είσοδο του συνδυαστικού τμήματος. Επίσης, θα συνδεθεί και με το επόμενο **Scan D Flip-Flop**. Έτσι, θα δημιουργηθεί η σειριακή αλυσίδα(**Scan Chain**) η οποία έχει μήκος **L = 4**. Το παραπάνω κύκλωμα φαίνεται σε **RTL** περιγραφή στο κεφάλαιο **APPENDIX** ο κώδικας **Κ3**.

Το module που περιγράφει το κύκλωμα έχει ως εισόδους τα σήματα **SE, SI** και **CLK** τα οποία είναι τα σήματα ελέγχου λειτουργίας του κυκλώματος. Επιπρόσθετα, έχει και ακόμα τέσσερις εισόδους τα **DI1 DI2 DI3** και **DI4**, όπου είναι οι είσοδοι στους πολυπλέκτες στην θέση 0 (**SE=0**). Κάθε έξοδος του **D Flip-Flop** αποτελεί ως είσοδο του πολυπλέκτη στην θέση 1 (**SE=1**) και αποτελούν επίσης είσοδοι ως προς το συνδυαστικό κύκλωμα. Η έξοδος **SO4** αποτελεί και την τελική έξοδο της σειριακής αλυσίδας. Παρακάτω φαίνεται και η επαλήθευση σχεδιασμού από το quartus.



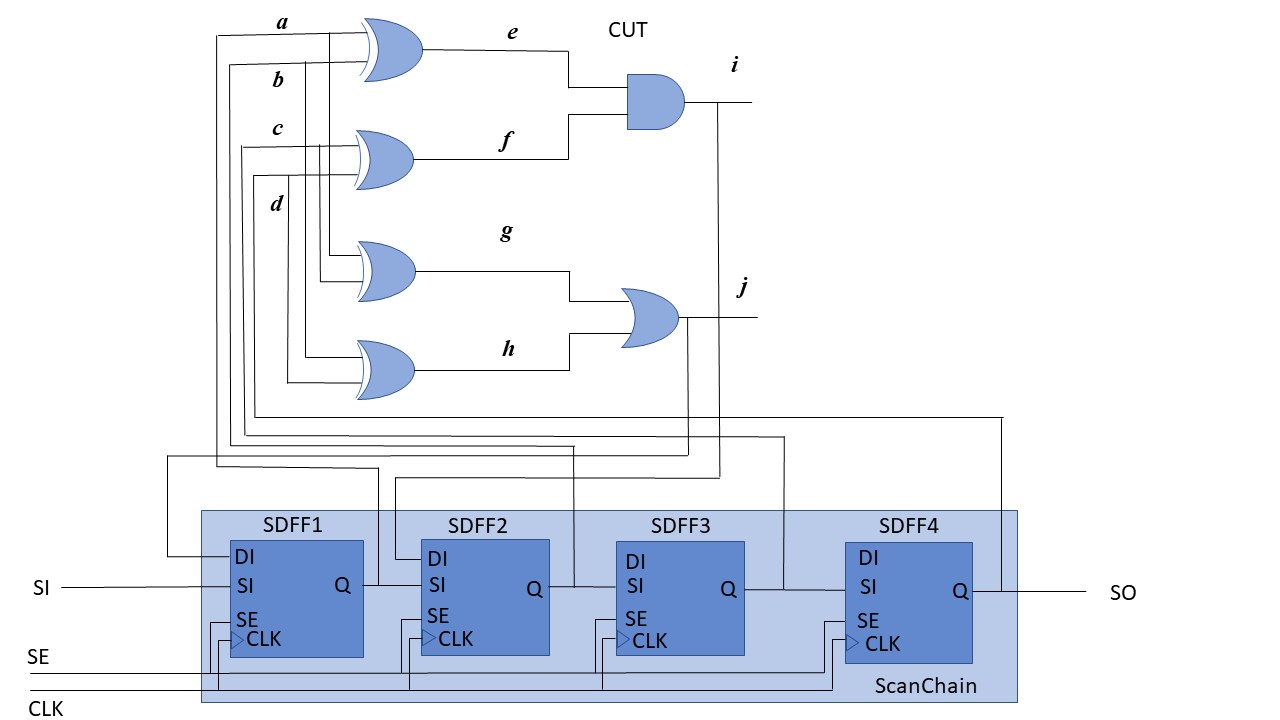
Το επόμενο στάδιο είναι η επαλήθευση του σχεδιασμού του παραπάνω κυκλώματος, όπου θα γίνει με την χρήση ενός testbench. Το testbench περιγράφεται από τον κώδικα **Κ4** του κεφαλαίου **APPENDIX**. Αρχικά, τα σήματα τοποθετούνται ως καταχωρητές **SEtb, SItb** και **CLKtb** για να μπορούν να δίνουν τιμές στο κύκλωμα. Επιπλέον, δημιουργήθηκαν ακόμα επιπλέον σήματα **CUT\_i(i=1,2,3,4)**, τα οποία δίνουν τιμές στο κύκλωμα μέσω της θέσης 0 του πολυπλέκτη ανάλογα με την λειτουργία του κυκλώματος βάσει του σήματος **SEtb**. Για να προσομοιώσουμε την λειτουργία του κυκλώματος, καλούμε το instance:

και ως εισόδους βάζουμε τους καταχωρητές που δηλώθηκαν και ως έξοδο τα **wires SOitb(i=1,2,3,4)**. Στη συνέχεια δημιουργούμε ένα loop το οποίο παράγει τους παλμούς του ρολογιού για να τροφοδοτηθεί το παραπάνω κύκλωμα με 20 μονάδες χρόνου κύκλου ρολογιού. Η αλλαγή ακμής γίνεται κάθε 10 μονάδες χρόνου από θετική σε αρνητική και από αρνητική σε θετική αντίστοιχα. Τέλος, έχουμε το block ελέγχου λειτουργίας του κυκλώματος, στο οποίο αναθέτονται τιμές στους καταχωρητές εισόδου. Επιπλέον ελέγχεται και η σωστή λειτουργία του κυκλώματος βάσει του σήματος **SEtb**. Παρακάτω φαίνεται η κυματομορφή:



Αρχικά, το σήμα **SEtb** παίρνει την τιμή 0. Έτσι, μέσα στην αλυσίδα θα εισάγονται τα δεδομένα από την είσοδο 0 του πολυπλέκτη. Παρατηρούμε ότι μετά από ένα χρονικό διάστημα έχουν μπει και οι τέσσερις τιμές **<DI1,DI2,DI3,DI4>=<1010>**. Επίσης, παρατηρείται και στην έξοδο των **flip-flop** η συγκεκριμένη τιμή. Στη συνέχεια, το σήμα **SEtb** παίρνει την τιμή 1 και έπειτα ξεκινά η είσοδος δεδομένων από το σήμα **SItb**. Όπως φαίνεται στην κυματομορφή παίρνει για λίγο την μονάδα και έπειτα ξανά πάει στην τιμή 0, δημιουργώντας το διάνυσμα εισόδου **<V01>=<01>**. Έτσι, ολισθαίνει τα **bit** ένα ένα σειριακά μέσα στην αλυσίδα μέχρι να μεταδοθούν οι τιμές στις εξόδους και δημιουργείται η ακολουθία δεδομένων στις εξόδους που φαίνεται παραπάνω.

Το τελευταίο κομμάτι της υλοποίησης είναι ο συνδυασμός των δύο παραπάνω οντοτήτων(instances) για την δημιουργία του τελικού κυκλώματος **TRCUT(Testable-Ready Circuit Under Test)**. Το συνολικό κύκλωμα αποτελείται από τρεις εισόδους τα **SI**, **SE** και **CLK**. Επίσης, έχει μία έξοδο **SO**, η οποία ως αποτέλεσμα βγάζει την απόκριση του συστήματος και ελέγχει εάν η λειτουργεία του κυκλώματος είναι σωστή. Ο συνδυασμός των κυκλωμάτων σχεδιάζεται ως εξής:



**c: controllability points**

**c**

**o**

**c**

**o**

**c**

**o**

**c**

**o**

**o**

**o**

**o**

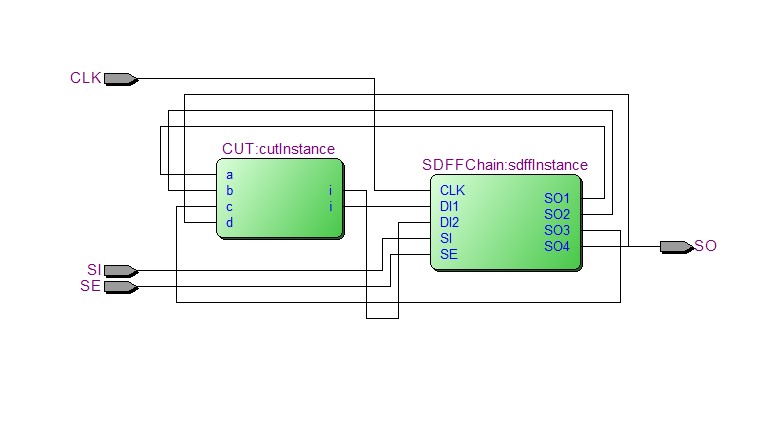
**o**

**o: observability points**

Έχουμε ότι οι έξοδοι των **flip-flops** θα αποτελούν τις εισόδους του συνδυαστικού κυκλώματος. Αυτά ονομάζονται και **controllability points**. Δηλαδή, έχουν τον πλήρη έλεγχο του κυκλώματος. Παράγουν τις αναγκαίες εισόδους για τον ορθό έλεγχο του κυκλώματος ως προς δοκιμή. Επίσης, οι έξοδοι του συνδυαστικού κυκλώματος **i** και **j** θα αποτελούν ως είσοδο της σειριακής αλυσίδας στα πρώτα δύο **Scan D Flip-Flop**. Από τις προδιαγραφές του κυκλώματος έχουμε ότι η έξοδος **i** θα μπει στο δεύτερο **Scan D Flip-Flop** για να σχηματιστεί το **Rbi** και η έξοδος **j** θα μπει στο πρώτο **Scan D Flip-Flop** για να σχηματιστεί το **Raj**. Έτσι, θα δημιουργηθούν τα **observability points**, τα οποία εισάγουν τις αποκρίσεις του κυκλώματος μέσα στην σειριακή αλυσίδα. Στόχος είναι η παρατήρηση του αποτελέσματος(έξοδος από την συνδυαστική λογική) και την σωστή λειτουργία του κυκλώματος.

Το παραπάνω κύκλωμα έχει σχεδιαστεί με βάσει την **RTL** περιγραφή του κώδικα **Κ5** που βρίσκεται στο κεφάλαιο **APPENDIX**. Αρχικοποιείται το **module TRCUT** με τις εισόδους και εξόδους που περιεγράφηκαν προηγουμένως. Επίσης, ορίζονται και τα **wires** των εξόδων του κυκλώματος για να μπουν στους καταχωρητές της **Scan Chain**. Η σύνδεση των δύο κυκλωμάτων γίνεται μέσω των instances

όπου οι έξοδοι τις αλυσίδας θα αποτελούν την είσοδο του συνδυαστικού κυκλώματος και οι έξοδοι του κυκλώματος θα είναι είσοδοι στους δύο πρώτους καταχωρητές, στις θέσεις 0 του πολυπλέκτη, που είναι μέσα στην **Scan Chain**. Επιπλέον, οι είσοδοι των δύο επόμενων καταχωρητών της αλυσίδας είναι κενοί, διότι θα υπάρχει κάποια έξοδος του κυκλώματος που να συνδέονται με αυτές. Παρακάτω φαίνεται και η επαλήθευση σχεδιασμού από το quartus.



Παρατηρούμε στον σχεδιασμό πως στην **Scan Chain** δεν υπάρχουν οι είσοδοι **DI3** και **DI4**, καθώς δεν τις χρειαζόμαστε για να δούμε εάν γίνεται σωστά η απόκριδη του κυκλώματος.

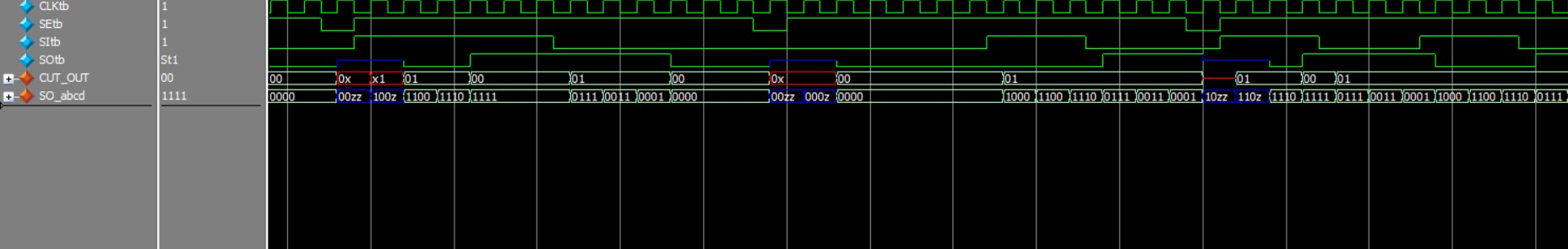
Το επόμενο στάδιο είναι η επαλήθευση του παραπάνω σχεδιασμού βάσει διανυσμάτων δοκιμής από τον πίνακα αλήθειας. Αυτό θα γίνει μέσω του testbench που περιγράφεται στον κώδικα **Κ6** του κεφαλαίου **APPENDIX**. Αρχικά, τα σήματα τοποθετούνται ως καταχωρητές **SEtb, SItb** και **CLKtb** για να μπορούν να δίνουν τιμές στο κύκλωμα. Επίσης, κρατάμε και ως **wire** την έξοδο **SOtb**, για να μπορέσει να εμφανιστεί το αποτέλεσμα του κυκλώματος στην κυματομορφή. Στη συνέχεια, καλείται το instance

το οποίο είναι και το τελικό μας κύκλωμα. Επιπρόσθετα, δημιουργούμε ένα loop το οποίο παράγει τους παλμούς του ρολογιού για να τροφοδοτηθεί το παραπάνω κύκλωμα με 20 μονάδες χρόνου κύκλου ρολογιού, με αλλαγή ακμής κάθε 10 μονάδες χρόνου από θετική σε αρνητική και από αρνητική σε θετική αντίστοιχα. Τέλος, έχουμε το block ελέγχου λειτουργίας του κυκλώματος, στο οποίο αναθέτονται τιμές στους καταχωρητές εισόδου. Αρχικά, τροφοδοτούμε με την τιμή 1 το σήμα **SEtb** για να ξεκινήσει η είσοδος των διανυσμάτων μέσα στο σύστημα. Καθώς έχουμε τέσσερις καταχωρητές, τότε το μήκος των διανυσμάτων δοκιμής είναι επίσης τέσσερα. Συνεπώς, για να πάρουν όλοι οι καταχωρητές τιμή, τότε ανά τρεις κύκλους ρολογιού εισάγουμε ένα **bit**, έτσι ώστε να δημιουργηθεί το πρώτο διάνυσμα.



Capture Mode

Παρατηρούμε ότι μπαίνουν τα δεδομένα σειριακά. Όταν πάρουν και οι τέσσερις καταχωρητές τιμές τότε θα ξεκινήσει ουσιαστικά ο έλεγχος του συνδυαστικού κυκλώματος. Για διάνυσμα δοκιμής **<0000>** βλέπουμε πως η έξοδος του κυκλώματος είναι και αυτή **<00>** όπως περιμέναμε. Για να δούμε την απόκριση του κυκλώματος στους καταχωρητές και στην έξοδο, θα πρέπει το κύκλωμα να μπει σε **Capture Mode** το οποίο συνεπάγεται ότι το σήμα **SEtb** θα πάρει την τιμή 0. Θα διαρκέσει έναν παλμό ρολογιού, ώστε να έχουμε την απόκριση του κυκλώματος. Παράλληλα, μετά το πέρας του παλμού θα ξανά γίνει μονάδα το σήμα **SEtb** για να μπει το επόμενο διάνυσμα δοκιμής στο κύκλωμα. Αυτή η διαδικασία επαναλαμβάνεται μέχρις ότου να εκτελεστούν όλα τα διανύσματα δοκιμής που παράγει ο πίνακας αλήθειας.



Κατά την εισαγωγή των διανυσμάτων δοκιμής στις εισόδους του κυκλώματος, παρατηρούμε επίσης, ότι γίνεται και ολίσθηση των προηγούμενων **bit** ως προς την έξοδο. Στόχος κατά την σειριακή σάρωση είναι να ολισθαίνονται όλες οι παλιές τιμές των εξόδων έξω από την **Serial Chain** για να μπει το επόμενο διάνυσμα δοκιμής. Επιπλέον, όταν το κύκλωμα εισέλθει σε **Capture Mode** παρατηρούμε πως στις εξόδους των καταχωρητών έχουμε δύο τιμές που εμφανίζονται κανονικά(απόκριση από το κύκλωμα) και δύο τιμές **z**(τιμές εμπέδησης). Αυτό συμβαίνει καθώς, οι είσοδοι **DI3** και **DI4** δεν είναι συνδεδεμένοι με κάποια έξοδο και δεν τροφοδοτούνται με τιμές. Τέλος, παρατηρούμε την έξοδο **SOtb** και περιμένουμε να πάρουμε τα **bit** που μπήκαν από την είσοδο **SItb**.

# Μέρος – Γ: Θεωρητική Άσκηση:

Έχουμε ότι η διαδικασία ψηφιακής σάρωσης εκτελείται με συχνότητα ρολογιού . Για να γίνει σωστή διεξαγωγή του πίνακα αλήθειας σε μία συσκευή πρέπει να δοκιμάσουμε όλες της τιμές του(διανύσματα). Αν υποθέσουμε ότι έχουμε **N** εισόδους στο κύκλωμα υπό δοκιμή, τότε τα διανύσματα δοκιμής θα είναι . Επίσης θα πρέπει να ξέρουμε και το μήκος της σειριακής αλυσίδας **L** για να βρούμε τον χρόνο που χρειαζόμαστε. Γενικά, θα χρειαστούμε **L** κύκλους ρολογιού για κάθε διάνυσμα δοκιμής συν έναν κύκλο ρολογιού για το **Capture Mode**. Οπότε, συνολικά θέλουμε **L+1** κύκλους για κάθε διάνυσμα. Άρα, για όλα τα διανύσματα θέλουμε συνολικ***ά***  κύκλους ρολογιού. Το επόμενο βήμα είναι να ολισθήσουν όλα τα **bit** εκτός της σειριακής αλυσίδας για να έχουμε το τελικό διάνυσμα απόκρισης, το οποίο θα απαιτεί όσο και η αλυσίδα. Συνεπώς χρειαζόμαστε επιπλέον **L** κύκλους ρολογιού. Όποτε, ο τελικός τύπος υπολογισμού κύκλων ρολογιού θα είναι ο εξής:

και επιπροσθέτως, για τον υπολογισμό του χρόνου διαιρούμε με την συχνότητα του ρολογιού. Άρα:

Συνεπώς, τώρα ας δούμε τον χρόνο που χρειάζεται για τις παρακάτω τιμές εισόδου. Υποθέτουμε ότι το μήκος της αλυσίδας θα είναι **L = 4**. Αν το **N = 10**, τότε θα έχουμε εισόδους. Αντικαθιστούμε στον τύπο και έχουμε:

και επομένως ο χρόνος λειτουργίας θα είναι

Αν το **N = 20**, τότε θα έχουμε εισόδους. Αντικαθιστούμε στον τύπο και έχουμε:

και επομένως ο χρόνος λειτουργίας θα είναι

Αν το **N = 30**, τότε θα έχουμε εισόδους. Αντικαθιστούμε στον τύπο και έχουμε:

και επομένως ο χρόνος λειτουργίας θα είναι

Αν το **N = 40**, τότε θα έχουμε εισόδους. Αντικαθιστούμε στον τύπο και έχουμε:

και επομένως ο χρόνος λειτουργίας θα είναι

Παρατηρούμε πως όσα περισσότερες εισόδους έχουμε σε ένα κύκλωμα παράλληλα με το μήκος της αλυσίδας, τόσο περισσότερο χρόνο ελέγχου χρειαζόμαστε.

# APPENDIX:

Παρακάτω εμφανίζονται οι RTL περιγραφές των κυκλωμάτων μαζί με τα αντίστοιχα testbench:

## Άσκηση-1.1:

**Κ1:**Κώδικας κυκλώματος CUT:

**`timescale 1ns/1ps**

**module CUT(a,b,c,d,i,j);**

**input a,b,c,d;**

**output i,j;**

**wire a,b,c,d; // Input wires**

**wire e,f,g,h; // Mid-level wires**

**wire i,j; // Output wires**

**//Structure of the circuit**

**//xor elements**

**//input elements**

**xor E1(e,a,b);**

**xor E2(f,c,d);**

**xor E3(g,a,c);**

**xor E4(h,b,d);**

**//and elements**

**//output elements**

**and E5(i,e,f);**

**and E6(j,g,h);**

**endmodule**

**Κ2**: Κώδικας testbench κυκλώματος CUT:

**`timescale 1ns/1ps**

**module CUTtb();**

**//input registers to our instantiated module**

**reg tb\_a, tb\_b, tb\_c, tb\_d;**

**//bus for writing data to the inputs**

**wire [3:0] tb\_cut\_inputs;**

**//wires for reading the output of the instantiated module**

**wire tb\_i, tb\_j;**

**//this is the instantiated module circuitModule.**

**//the name of the instance is cutCircuitInstance**

**CUT cutCircuitInstance(tb\_a,tb\_b,tb\_c,tb\_d,tb\_i,tb\_j);**

**//Functional testing of the logic circuit**

**//now we create the bus that consists of the four input registers values**

**assign tb\_cut\_inputs = {tb\_a, tb\_b, tb\_c, tb\_d};**

**//this block is running only at the beginning of the simulation**

**initial begin**

**{tb\_a, tb\_b, tb\_c, tb\_d} = 4'b0000;**

**//the following line runs forever every 5 time units**

**forever #20 {tb\_a, tb\_b, tb\_c, tb\_d} = tb\_cut\_inputs + 1;**

**end**

**endmodule**

**Κ3**: Κώδικας κυκλώματος SDFFChain:

**`timescale 1ns/1p**

**module SDFFChain(CLK, DI1, DI2, DI3, DI4, SI, SE, SO1, SO2, SO3, SO4);**

**input CLK, DI1, DI2, DI3, DI4, SI, SE;**

**output SO1, SO2, SO3, SO4;**

**//Input wires for each element**

**wire CLK, DI1, DI2, DI3, DI4, SI, SE;**

**//Output wires for each element**

**wire SO1, SO2, SO3, SO4;**

**// Four scan d flip-flop instances**

**//Scan chain**

**SDFF sdff1(CLK, DI1, SI, SE, SO1); //Inputs: SI, DI1 | Output: SO1**

**SDFF sdff2(CLK, DI2, SO1, SE, SO2); //Inputs: SO1, DI2 | Output: SO2**

**SDFF sdff3(CLK, DI3, SO2, SE, SO3); //Inputs: SO2, DI3 | Output: SO3**

**SDFF sdff4(CLK, DI4, SO3, SE, SO4); //Inputs: SO3, DI4 | Output: SO4**

**endmodule**

**Κ4**: Κώδικας testbench κυκλώματος SDFFChain:

**`timescale 1ns/1ps**

**module SDFFChaintb();**

**reg CLKtb, SEtb, SItb;**

**//input wires**

**wire DI1\_SI,DI\_2,DI\_3,DI\_4;**

**//output wires**

**wire SO1tb, SO2tb, SO3tb, SO4tb;**

**reg CUT\_1,CUT\_2,CUT\_3,CUT\_4;**

**assign DI\_1 = CUT\_1;**

**assign DI\_2 = CUT\_2;**

**assign DI\_3 = CUT\_3;**

**assign DI\_4 = CUT\_4;**

**//the name of the instance is sdffInstance**

**SDFFChain sdffInstance(CLKtb,DI\_1, DI\_2, DI\_3, DI\_4, SItb, SEtb, SO1tb, SO2tb, SO3tb, SO4tb);**

**//Block for clock generation**

**initial begin**

**CLKtb=0;**

**#20**

**forever begin**

**#10 CLKtb=!CLKtb;**

**end**

**end**

**//Block for functional testing**

**initial**

**begin**

**////Testing of SE=0 (input path from DI1)**

**SEtb=0;**

**CUT\_1=0;**

**#60**

**CUT\_1=1;**

**#60**

**CUT\_2=0;**

**#60**

**CUT\_3=1;**

**#60**

**CUT\_4=0;**

**#60**

**////Testing of SE=1 (input path from SI)**

**SEtb=1;**

**SItb=1;**

**#60**

**SItb=0;**

**end**

**endmodule**

**Κ5:**Κώδικας κυκλώματος TRCUT:

**`timescale 1ns/1ps**

**module TRCUT(SI, SE, CLK, SO);**

**input SI, SE, CLK;**

**output SO;**

**//Input wires for each element**

**wire SI, SE, CLK;**

**//Output wires for each element**

**wire i\_out, j\_out;**

**wire SOa, SOb, SOc, SO;**

**//Structure of the test circuit**

**SDFFChain sdffInstance(CLK, j\_out, i\_out, , , SI, SE, SOa, SOb, SOc, SO);**

**CUT cutInstance(SOa, SOb, SOc, SO, i\_out, j\_out);**

**endmodule**

## Άσκηση-1.2:

**Κ6:**Κώδικας testbench κυκλώματος TRCUT:

**`timescale 1ns/1ps**

**module TRCUTtb();**

**reg CLKtb, SEtb, SItb;**

**wire SOtb;**

**TRCUT trcutInstance(SItb, SEtb, CLKtb, SOtb);**

**//Block for clock generation**

**initial begin**

**CLKtb = 0;**

**#20**

**forever begin**

**#10 CLKtb =! CLKtb;**

**end**

**end**

**//Block for functional testing of logic circuit CUT**

**initial**

**begin**

**///Serial Input(SE=1)**

**SEtb = 1;**

**#60 SItb = 0;#60 SItb = 0;#60 SItb = 0;#60 SItb = 0; //SI = <0000>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 0;#60 SItb = 0;#60 SItb = 0;//SI = <0001>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 1;#60 SItb = 0;#60 SItb = 0;//SI = <0010>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 1;#60 SItb = 0;#60 SItb = 0;//SI = <0011>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 0;#60 SItb = 1;#60 SItb = 0;//SI = <0100>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 0;#60 SItb = 1;#60 SItb = 0;//SI = <0101>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 1;#60 SItb = 1;#60 SItb = 0;//SI = <0110>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 1;#60 SItb = 1;#60 SItb = 0;//SI = <0111>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 0;#60 SItb = 0;#60 SItb = 1;//SI = <1000>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 0;#60 SItb = 0;#60 SItb = 1;//SI = <1001>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 1;#60 SItb = 0;#60 SItb = 1;//SI = <1010>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 1;#60 SItb = 0;#60 SItb = 1;//SI = <1011>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 0;#60 SItb = 1;#60 SItb = 1;//SI = <1100>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 0;#60 SItb = 1;#60 SItb = 1;//SI = <1101>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 0;#60 SItb = 1;#60 SItb = 1;#60 SItb = 1;//SI = <1110>**

**#60 SEtb = 0;///Capture Mode for one cycle**

**#20 SEtb = 1;SItb = 1;#60 SItb = 1;#60 SItb = 1;#60 SItb = 1;//SI = <1111>**

**//END**

**end**

**endmodule**