Θέμα 1^{ο}

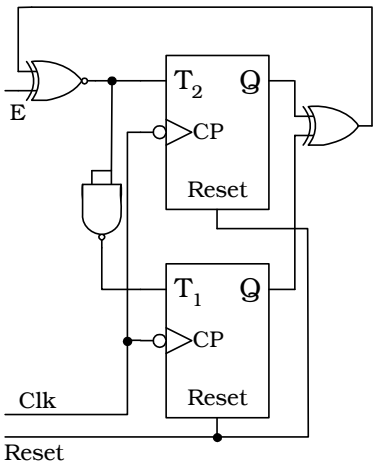
Με αρνητικά ακμοπυροδότητα T FFs να σχεδιάσετε ένα σύγχρονο ακολουθιακό κύκλωμα με μία είσοδο E, το οποίο για E=0 να διατρέχει διαδοχικά τις καταστάσεις **0, 2, 3, 1,** 0, 2, 3, 1, 0, ... ενώ για E=1 να διατρέχει διαδοχικά τις καταστάσεις **0, 1, 3, 2,** 0, 1, 3, 2, 0, ... Ζητείται να σχεδιάσετε το **απλούστερο** δυνατό κύκλωμα, με τον **μικρότερο** αριθμό πυλών.

- Δώστε τη Verilog περιγραφή του στοχευόμενου κυκλώματος σα Moore FSM.
- Δώστε τη περιγραφή σε behavioral Verilog για ένα T FF. Χρησιμοποιώντας το ως δομικό στοιχείο, περιγράψτε σε structural Verilog, το κύκλωμα στο οποίο καταλήξατε στο 1^{ο} υποερώτημα.

Αφού οι διαφορετικές καταστάσεις που έχουμε είναι 4, θα χρειαστούμε 2 T ffs. Θεωρούμε δυαδική κωδικοποίηση των δεδομένων καταστάσεων. Καταλήγουμε συνεπώς στον εξής πίνακα μετάβασης καταστάσεων και εισόδων των ακολουθιακών στοιχείων:

	Παρούσα Κατάσταση		**Επόμενη Κατάσταση**		**Είσοδες**	
Είσοδος E	*Q2(t)*	*Q1(t)*	*Q2(t+1)*	*Q1(t+1)*	*T2(t+1)*	*T1(t+1)*
0	0	0	1	0	1	0
0	0	1	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	0	1
1	0	1	1	1	1	0
1	1	0	0	0	1	0
1	1	1	1	0	0	1

Από το πίνακα φαίνεται ότι T2 = ~T1 και συνεπώς αρκεί να απλοποιήσουμε μία συνάρτηση μόνο. Για την T2 είναι T2 = E'Q2'Q1' + E'Q2Q1 + EQ2'Q1 + EQ2Q1' = E'(Q2 ⊗ Q1) + E (Q2 ⊕ Q1). Αντικαθιστώντας (Q2 ⊕ Q1) = Δ, παίρνω T2 = E'Δ' + EΔ = E ⊗ Δ = E ⊗ (Q2 ⊕ Q1). Αρα το ζητούμενο κύκλωμα είναι :




```
module FSMdescription (Clk, Reset, E, State);
input          Clk, Reset, E;
output [1:0] State;
reg       [1:0] State;
parameter S0=0, S1=1, S2=2, S3=3;
always @(negedge Clk or posedge Reset)
  if (Reset) State <= S0;
   else if (E)
          case (State)
             S0 : State <= S1;
             S1 : State <= S3;
             S2 : State <= S0;
             S3 : State <= S2;
          endcase
         else
          case (State)
             S0 : State <= S2;
             S1 : State <= S0;
             S2 : State <= S3;
             S3 : State <= S1;
          endcase
endmodule
```

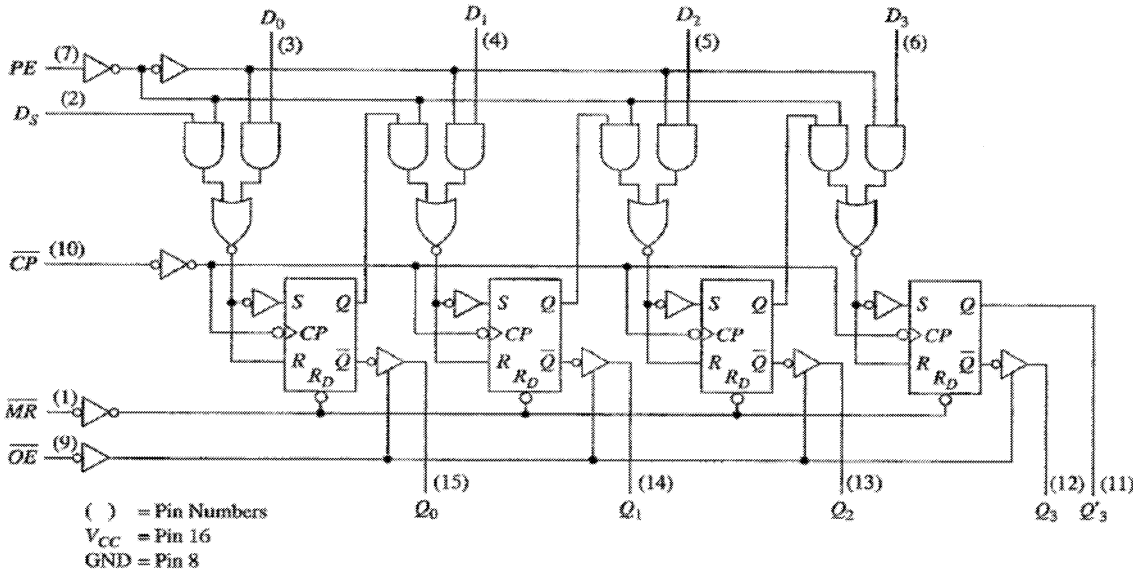
```
modul
e TFFbeh (Clk, Reset, T, Q);
  input   Clk, Reset, T;
  output Q;
  reg Q;

  always @(negedge Clk or posedge Reset)
    if (Reset) Q<=0;
      else if (T) Q<=~Q;
endmodule

module structcir (Clk, Reset, E, State);
input          Clk, Reset, E;
output [1:0] State;

xor (inx, State[0], State[1]);
xor (T1, E, inx);
TFFbeh FF1 (Clk, Reset,   T1, State[0]);
TFFbeh FF2 (Clk, Reset, ~T1, State[1]);
endmodule
```

Θέμα 2ο



- Στη παραπάνω εικόνα φαίνεται το κύκλωμα που υλοποιείται εντός του ολοκληρωμένου 74LS395A. Για τι κύκλωμα πρόκειται ? Εξηγείστε πλήρως τη λειτουργία του και κατασκευάστε τον πίνακα αληθείας του.
- Περιγράψτε σε behavioral Verilog το πιο πάνω κύκλωμα.
- Χρησιμοποιώντας το module του προηγούμενου υποερωτήματος ως δομικό στοιχείο, περιγράψτε ένα κύκλωμα των 8 δυαδικών ψηφίων με τα ίδια ακριβώς χαρακτηριστικά.

Από την εικόνα είναι προφανές ότι :

- Χρησιμοποιούνται 4 αρνητικά ακμοπυροδότητα SR FF. Αφού στις εισόδους S και R φορτώνονται συμπληρωματικές τιμές χρησιμοποιούνται στην ουσία σα D FF με αντιστραμμένη είσοδο.
- Τα FF έχουν ασύγχρονη είσοδο καθαρισμού ενεργή με χαμηλό δυναμικό.
- Στην έξοδό τους υπάρχουν στοιχεία 3ών καταστάσεων ελεγχόμενα από τον ακροδέκτη ~OE.
- Στην είσοδό τους υπάρχουν πολυπλέκτες 2 σε 1, που ελέγχονται από το σήμα PE. Οταν αυτό είναι 1, επιτρέπει τη φόρτωση από τις παράλληλες γραμμές δεδομένων D3 – D0. Οταν είναι 0, στο αριστερότερο FF φορτώνεται τιμή από την είσοδο Ds (σειριακή είσοδο δεδομένων) ενώ στα υπόλοιπα FFs φορτώνεται στο καθένα η τιμή του αμέσως αριστερότερου.
- Η έξοδος Q3΄είναι η σειριακή έξοδος, που μπορεί να χρησιμοποιηθεί για επέκταση.

Συνεπώς, πρόκειται για ένα κύκλωμα καταχωρητή / ολισθητή, με παράλληλη φόρτωση, ασύγχρονο καθαρισμό και έξοδο τριών καταστάσεων. Μπορεί να χρησιμοποιηθεί σαν απλός καταχωρητής, ή για μετατροπές από σειριακή σε παράλληλη και αντίστροφα. Ο πίνακας αληθείας του κυκλώματος έχει ως ακολούθως :

~OE	~MR	PE	CP	Q3(t+1)	Q2(t+1)	Q1(t+1)	Q0(t+1)
1	X	X	X	Hi-Z	Hi-Z	Hi-Z	Hi-Z
0	0	X	X	0	0	0	0
0	1	0	↓	Q2(t)	Q1(t)	Q0(t)	Ds
0	1	1	↓	D3	D2	D1	D0
0	1	X	≠ ↓	No change	No change	No change	No change

```
module parser4 (MRn, OEn, CP, Ds, PE, D, Q, Q3);
  input         MRn, OEn, CP, Ds, PE;
  input   [3:0] D;
  output [3:0] Q;
  output        Q3;
  reg     [3:0] tmp;

  assign Q   = (OEn) ? 4'bz : tmp;
  assign Q3 = tmp[3];

 always @(negedge CP or negedge MRn)
  if (~MRn) tmp <= 4'h0;
   else if (PE) tmp <= D;
         else tmp <= {tmp[2:0], Ds};
endmodule
```

```
module parser8 (MRn, OEn, CP, Ds, PE, D, Q, Q8);
  input         MRn, OEn, CP, Ds, PE;
  input   [7:0] D;
  output [7:0] Q;
  output        Q8;
  wire          Q3;

parser4 low  (MRn, OEn, CP, Ds, PE, D[3:0], Q[3:0], Q3);
parser4 high (MRn, OEn, CP, Q3, PE, D[7:4], Q[7:4], Q8);
endmodule
```

Θέμα 3ο

Ζητείται να υλοποιήσετε τις :

- f_1 (a) = ~a (NOT a)
- f_2 (b,c) = ~b | c (NOT b OR c) και
- f_3 (a,d) = a ^ d (a XOR d)

χρησιμοποιώντας μια και μοναδική EEPROM. Η απάντησή σας θα πρέπει οπωσδήποτε να περιλαμβάνει :

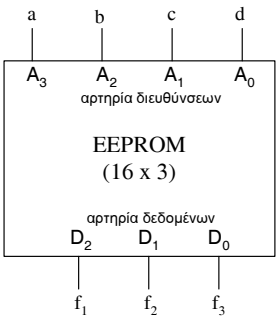
- Το μέγεθος της EEPROM που θα χρειαστείτε.

- Τα περιεχόμενα κάθε θέσης της EEPROM.
- Ενα σχηματικό διάγραμμα, που θα δείχνει τι θα συνδεθεί σε ποια γραμμή διευθύνσεων της EEPROM και σε ποιες γραμμές δεδομένων θα προκύψουν οι ζητούμενες συναρτήσεις.

Θέλουμε την υλοποίηση 3ών συναρτήσεων. Αφού συνολικά οι συναρτήσεις μας χρησιμοποιούν 4 διαφορετικές μεταβλητές, θα μετατρέψουμε αυτές τις συναρτήσεις σε συναρτήσεις των 4ων μεταβλητών. Τότε θα χρειαστούμε μια EEPROM των 2^4 θέσεων. Κάθε συνδυασμός τιμών των μεταβλητών, δημιουργεί μια διαφορετική διεύθυνση της EEPROM και συνεπώς επιλέγει μία εκ των πιθανών θέσεών της. Σε κάθε θέση θα είναι αποθηκευμένα 3 δυαδικά ψηφία, οι τιμές δηλαδή των 3ών συναρτήσεων γι' αυτό το συνδυασμό τιμών των εισόδων. Η μετατροπή μπορεί να γίνει αλγεβρικά ή με ένα πίνακα αληθείας όπως ο παρακάτω :

Πιθανές τιμές των μεταβλητών (ισοδύναμα πιθανές διευθύνσεις της EEPROM)				**Τιμές των συναρτήσεων (Ισοδύναμα περιεχόμενα της EEPROM στη συγκεκριμένη θέση)**		
a	*b*	*c*	*d*	*f1*	*f2*	*f3*
0	0	0	0	1	1	0
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	1	1	1
0	1	0	0	1	0	0
0	1	0	1	1	0	1
0	1	1	0	1	1	0
0	1	1	1	1	1	1
1	0	0	0	0	1	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	0
1	1	1	0	0	1	1
1	1	1	1	0	1	0

Αρα συνολικά θα χρειαστούμε μια EEPROM 16x3 δυαδικών ψηφίων. Οι μεταβλητές θα χρησιμοποιηθούν στις γραμμές διευθύνσεων της EEPROM. Εδώ θα πρέπει να προσέξουμε και να φαίνεται καθαρά στο σχήμα ότι έτσι όπως καταστρώσαμε το πίνακα, το a θα πρέπει να συνδεθεί στη γραμμή διεύθυνσης υψηλότερης σημαντικότητας. Υποθέτοντας ότι οι γραμμές διεύθυνσης είναι A3 έως A0, το a θα πρέπει να συνδεθεί στο A3, το β στο A2 κ.ο.κ. Επίσης θα πρέπει να είναι καθαρό ότι αν οι γραμμές δεδομένων της EEPROM είναι D2 έως D0, η f1 θα προκύψει στην D2 κ.ο.κ. Αρα το ζητούμενο σχηματικό είναι :

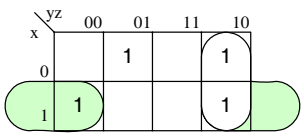


Θέμα 4ο

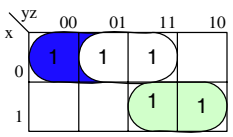
Στο σχήμα φαίνεται μια PAL με 3 εισόδους και 4 εξόδους. Στην PAL αυτή θέλουμε να εμφωλεύσουμε τις συναρτήσεις των οποίων ο πίνακας αληθείας φαίνεται παρακάτω.

Είσοδοι			Εξοδοι			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

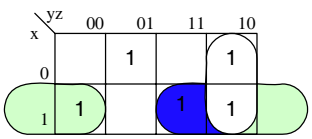
Εξάγετε τον πίνακα προγραμματισμού της PAL και σημειώστε **μόνο** τους συνδέσμους που πρέπει να μείνουν άθικτοι στο σχηματικό της PAL.



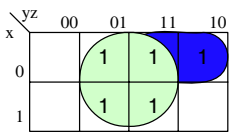
A = x΄ y΄z + yz΄+ xz΄



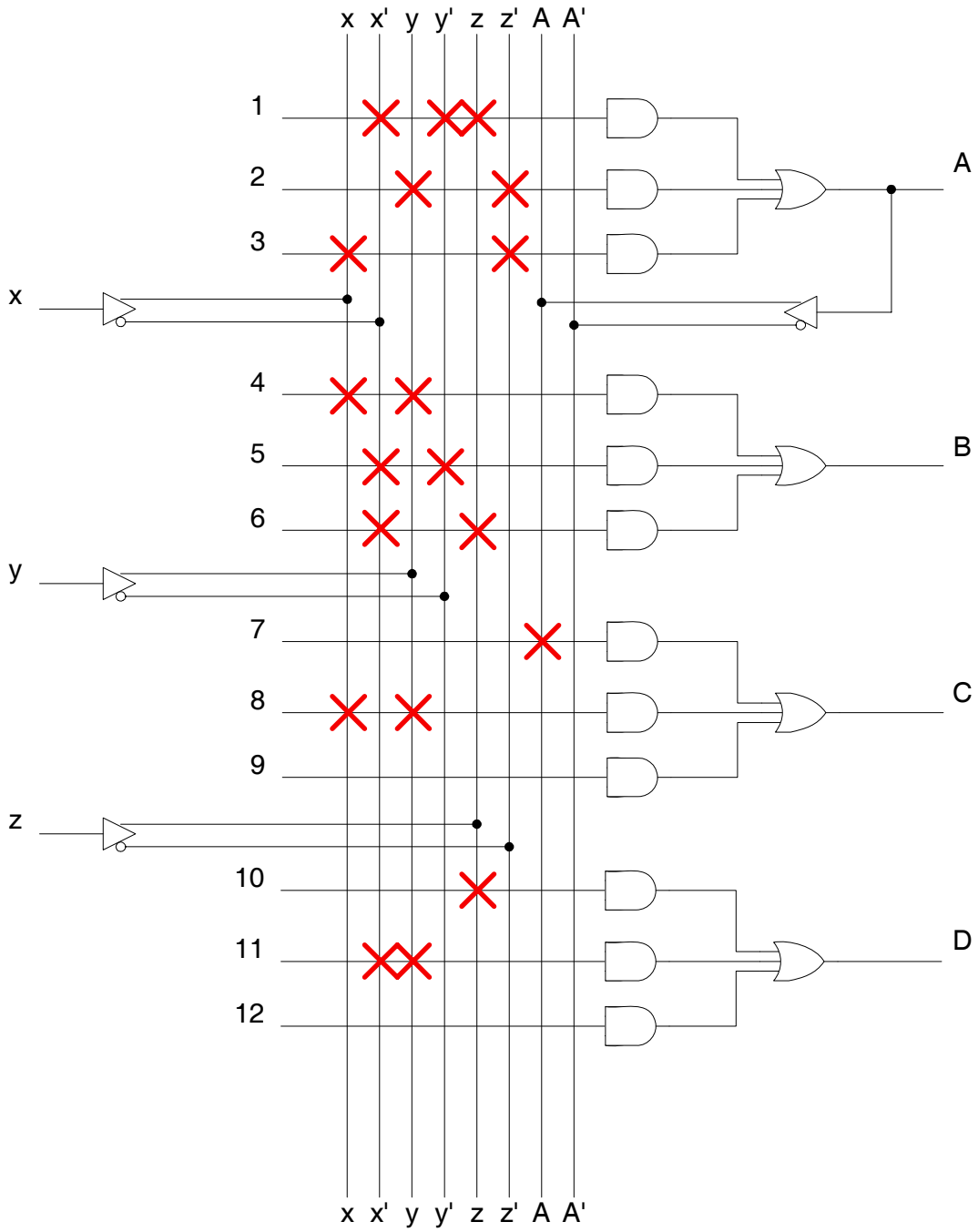
B = x y + x΄z+ x΄y΄



C= x΄y΄z + yz΄ + xz΄ + xy = A+xy



D= z + yx΄


x x' y y' z z' A A'
1
2
3
x
4
5
6
y
7
8
9
z
10
11
12
A
B
C
D
x x' y y' z z' A A'