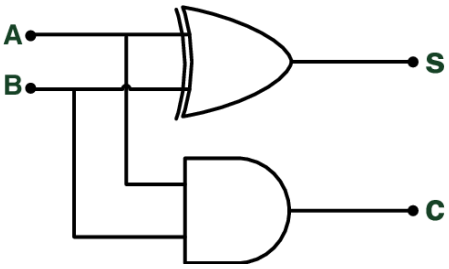
Εισαγωγή στη Σχεδίαση Συστημάτων VLSI

5η εργαστηριακή άσκηση

Ερώτηση 1:

Σε αυτό το ερώτημα θα γίνει με χρήση του compiler μιας γραμμής ή/και του compiler αρχείου του microwind η υλοποίηση των ακόλουθων layout:

Ημιαθροιστή (H-A):



Πίνακας αληθείας H-A:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Αυτή η υλοποίηση του ημιαθροιστή, που αποτελείται από μια πύλη XOR δύο εισόδων και μια πύλη AND δύο εισόδων, παρουσιάζει τη μικρότερη καθυστέρηση τόσο για το μερικό άθροισμα, όσο και για το κρατούμενο εξόδου. Αυτό συμβαίνει διότι και οι δύο αυτές συναρτήσεις Boole S(A,B) και C_{out}(A,B) υλοποιούνται με μια πύλη η καθεμία, οι οποίες μάλιστα βρίσκονται στο ίδιο επίπεδο.

Κώδικας Verilog:

Η υλοποίηση του κώδικα έγινε με διάφορες δοκιμές, αξιοποιώντας τις διάφορες μεθόδους υλοποιήσεις που δίνει το Verilog, δηλαδή υλοποίηση μέθοδο gate level και data flow.

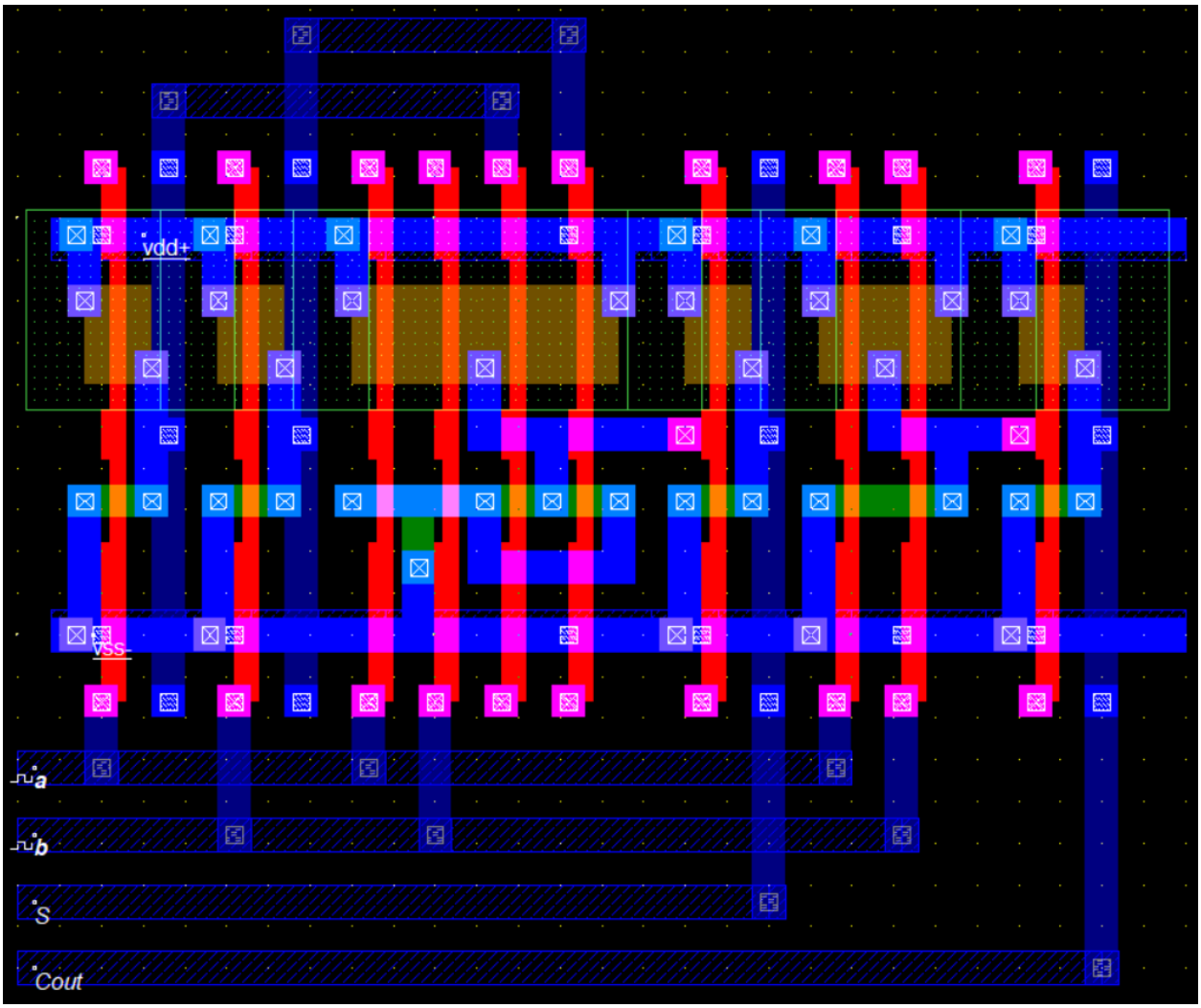
Η υλοποίηση με gate level παρατηρήθηκε ότι έβγαζε διάφορα προβλήματα στην γραφική παράσταση του Sum που γινόταν με την εντολή xor(Sum,a,b); ή not(na,a); not(nb,b); xnor(Sum,na,nb); αφού η βιβλιοθήκη της Verilog για τις πύλες xor και xnor δεν είναι καλά δομημένη.

Από την άλλη, η υλοποίηση με data flow παρατηρήθηκε ότι η υλοποίηση του ημιαθροιστή βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

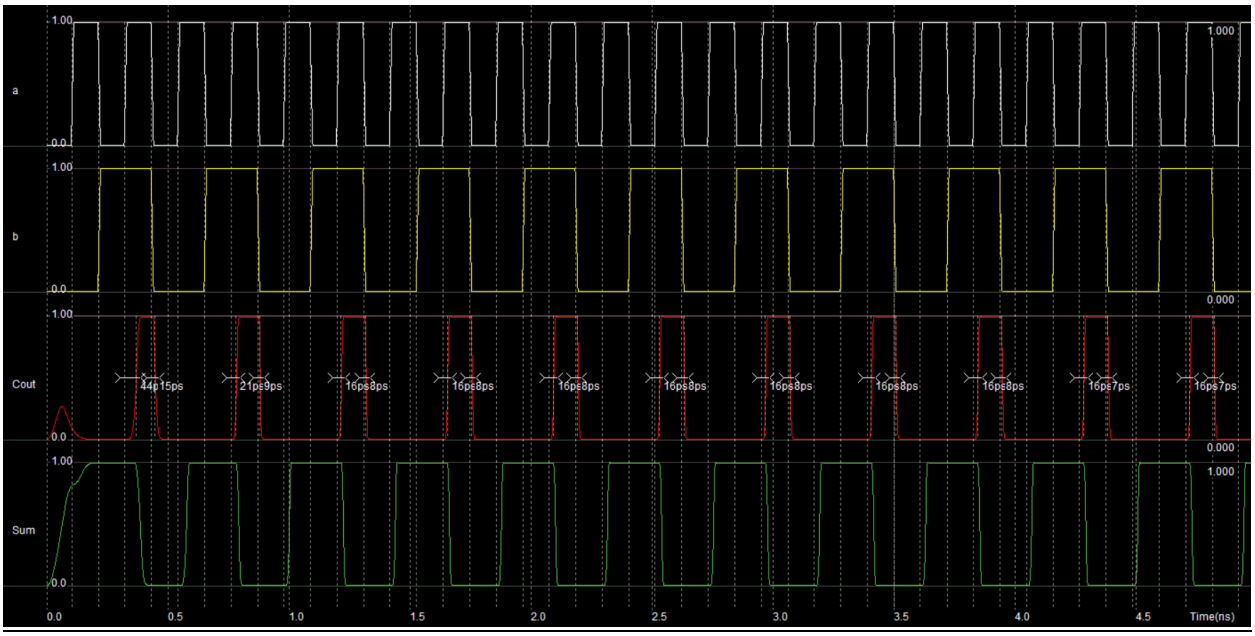
```
module hadd(Sum,Cout,a,b)

        input a,b;

        output Sum,Cout;

        assign anot = ~a
```

```
    assign bnot = ~b
    assign Sum = (a|b)&(anot|bnot)
    assign Cout = a&b
endmodule
```

Κυκλωματική υλοποίηση half-adder:



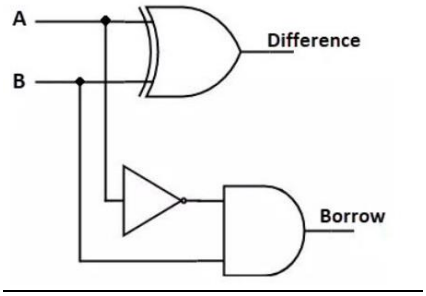
Γραφική παράσταση:



Ημιαφαιρέτη (H-S):

Για τον ημιαφαιρέτη ισχύουν οι λογικές εκφράσεις:

- Difference = A'B + AB' =A(+)B
- Borrow = A'B



Πίνακας αληθείας H-S:

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

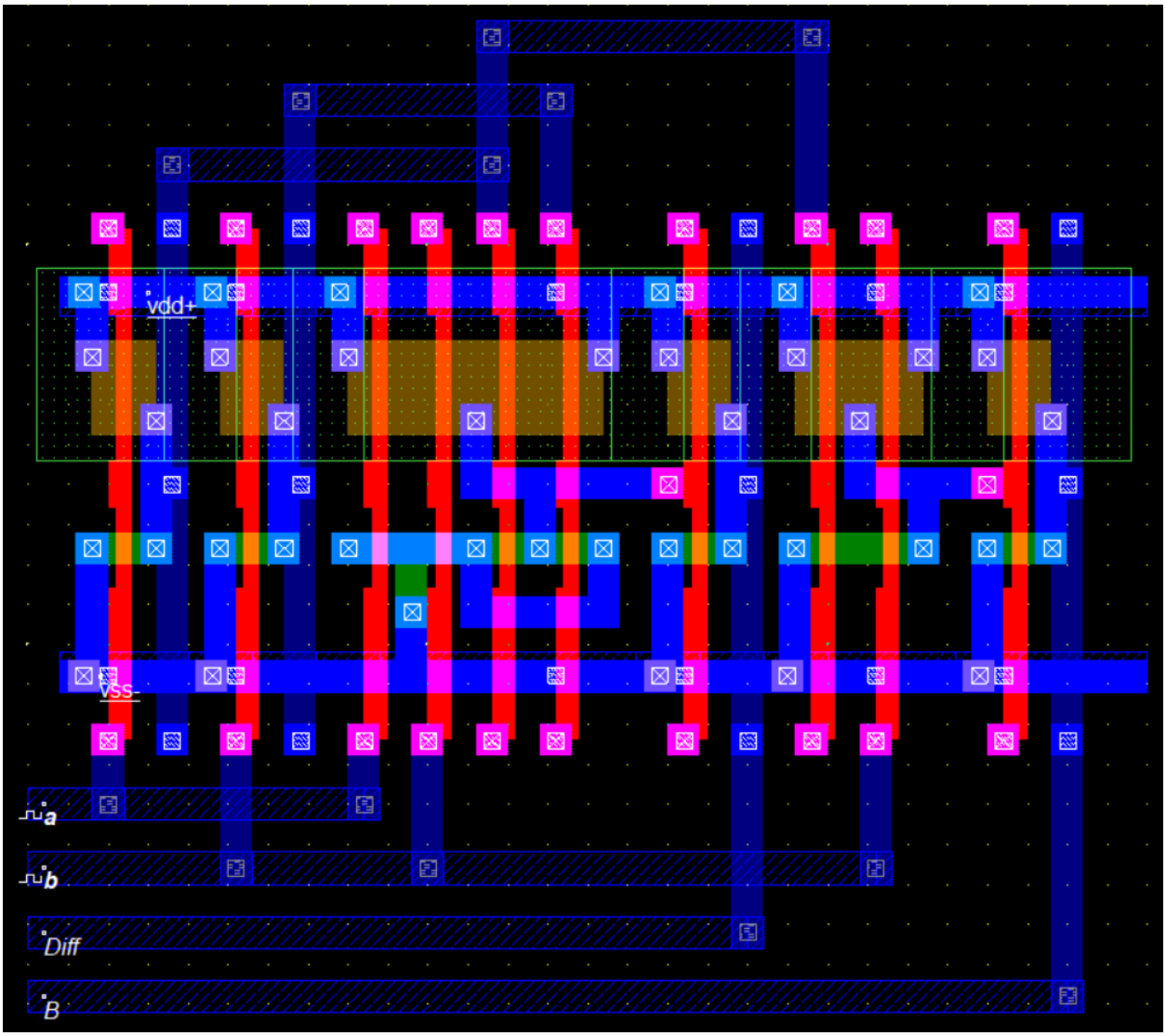
Κώδικας Verilog:

Η υλοποίηση με data flow του ημιαφαιρέτη παρατηρήθηκε ότι βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

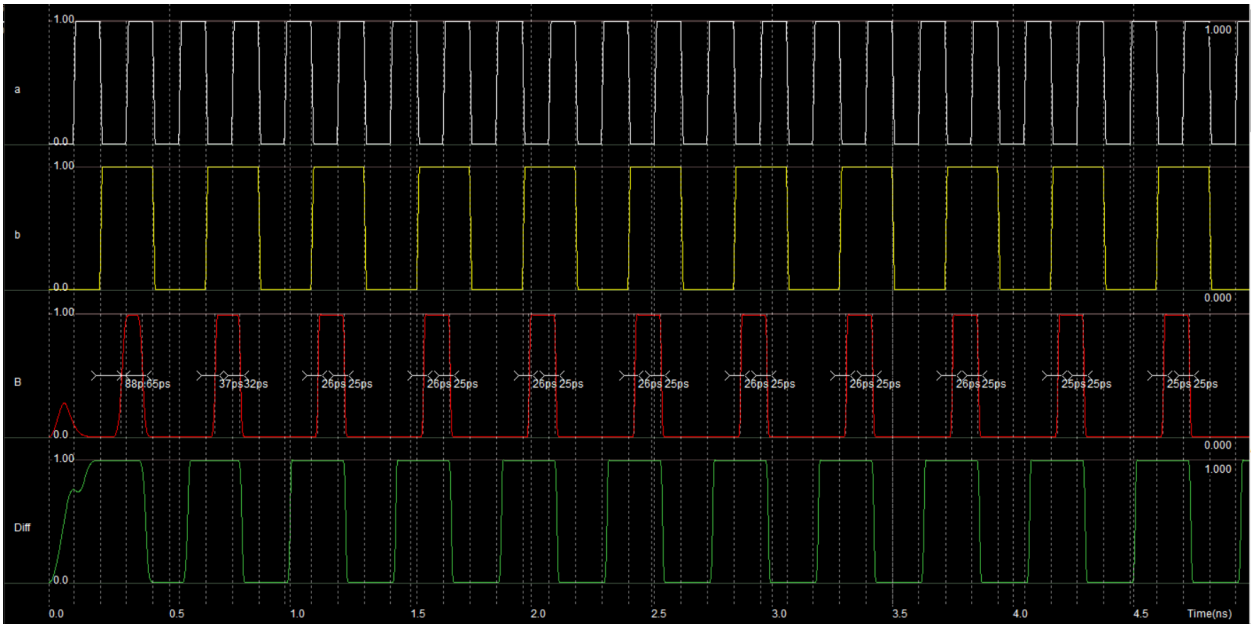
```
module hasub(Diff,B,a,b)
        input a,b;
        output Diff,B;
        assign anot = ~a
        assign bnot = ~b
        assign Diff = (a|b)&(anot|bnot)
```

```
    assign B = anot&b
endmodule
```

Κυκλωματική υλοποίηση Half-Subtractor:



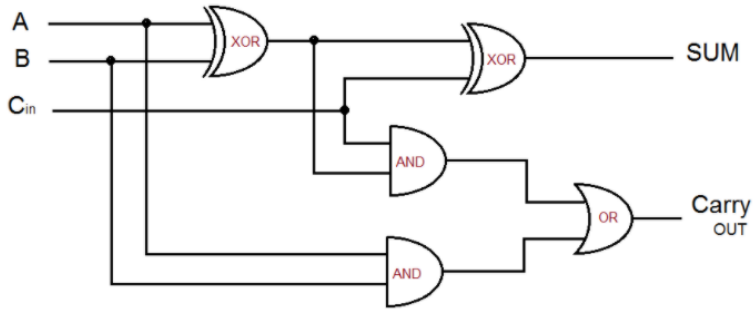
Γραφική Παράσταση:



Πλήρη αθροιστή (F-A):

Η υλοποίηση του πλήρους αθροιστή, που αποτελείται από δύο πύλες XOR, για το μερικό άθροισμα AND OR ισχύουν οι λογικές εκφράσεις:

- Sum= A(+)B(+)Cin
- Carry_out = AB+(A(+)B)Cin



Πίνακας Αληθείας F-A:

INPUTS			OUTPUTS	
A	B	Cin	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1

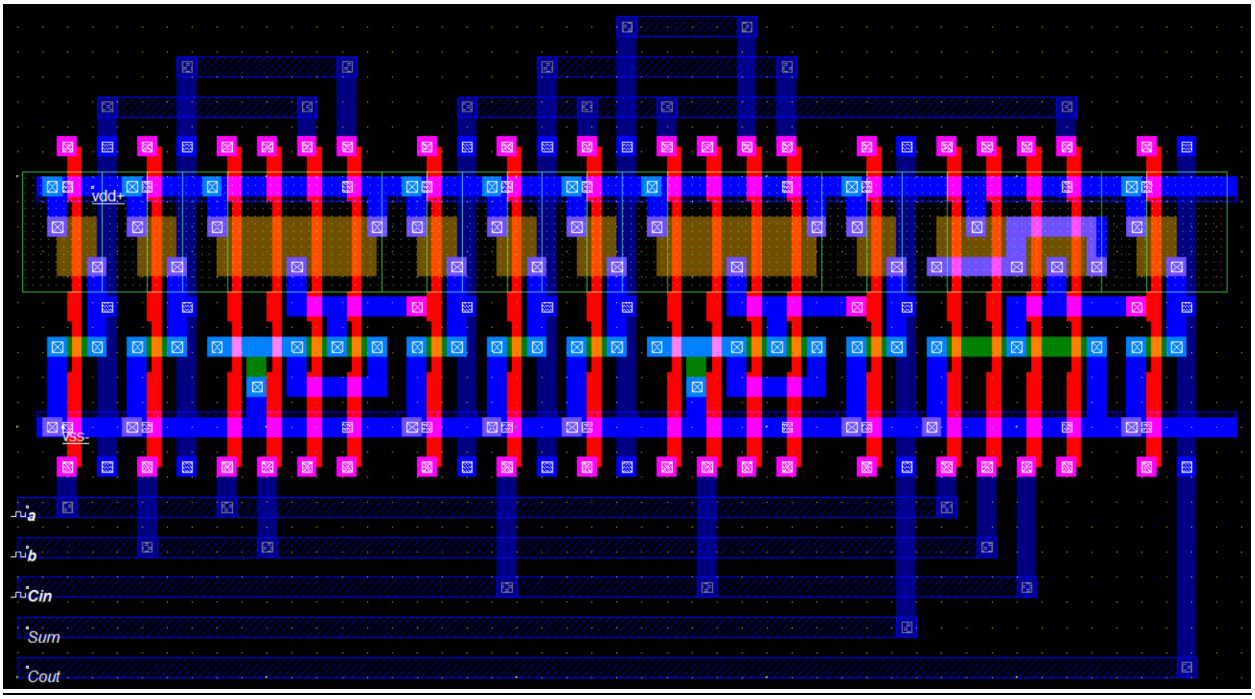
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Κώδικας Verilog:

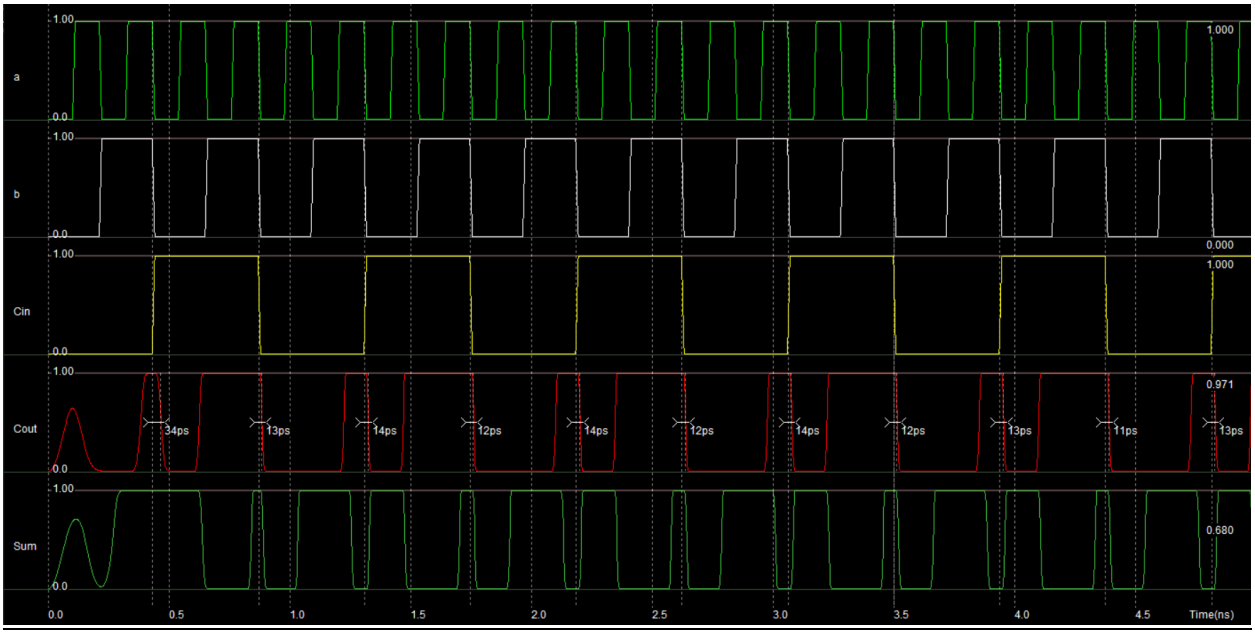
Η υλοποίηση με data flow του πλήρους αθροιστή παρατηρήθηκε ότι βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

```
module fulad(Sum,Cout,a,b,Cin)
        input a,b,Cin;
        output Sum,Cout;
        wire w1;
        assign anot = ~a
        assign bnot = ~b
        assign w1 = (a|b)&(anot|bnot)
        assign cnot = ~Cin
        assign w1not = ~w1
        assign Sum = (w1|Cin)&(w1not|cnot)
        assign Cout = (a&b)|(Cin&w1)
endmodule
```

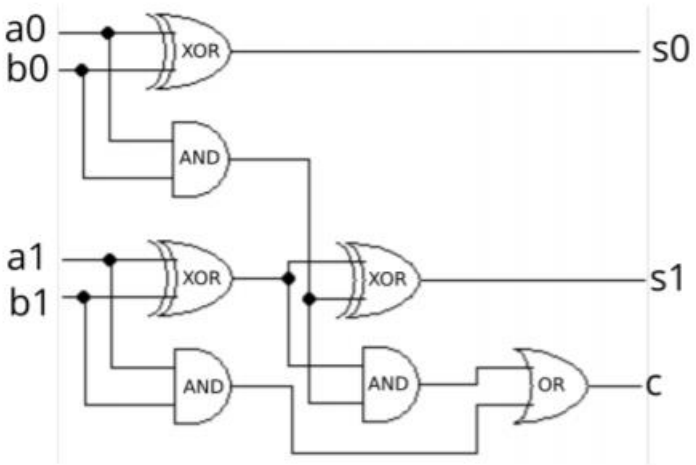
Κυκλωματική υλοποίηση Full-Adder:



Γραφική παράσταση:



Αθροιστή δυο αριθμών των δυο bit που παρέχει στην έξοδό του τις μεταβλητές S0, S1 και S2:



Πίνακας Αληθείας 2bit-Adder:

INPUTS				OUTPUTS		
A1	A0	B1	B0	S2(Carryout)	S1	S0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

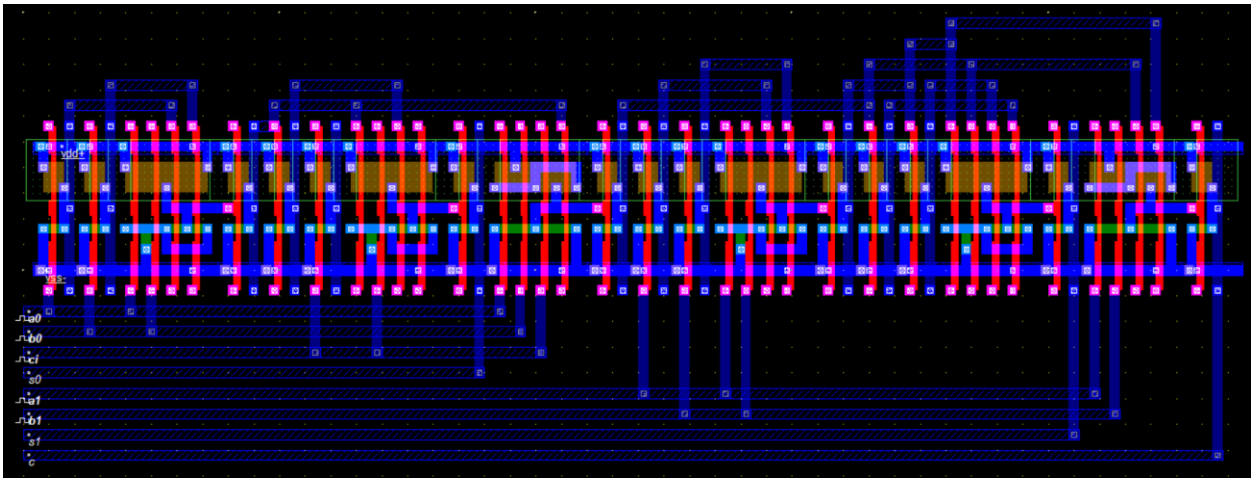
Κώδικας Verilog:

Η υλοποίηση με data flow του αθροιστή δυο αριθμών των δυο bit παρατηρήθηκε ότι βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

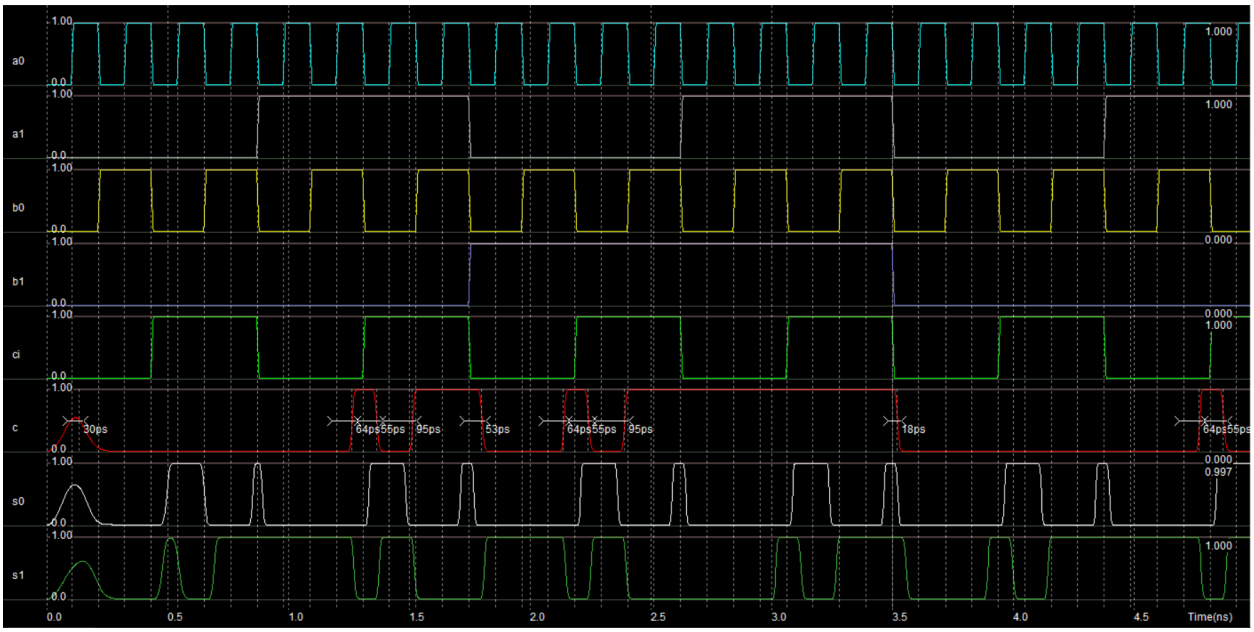
```
module add2bit (s0,s1,c,ci,a1,b1,a0,b0);
```

```
    input a1, b1,a0,b0,ci;
    output s0,s1,c;
    wire w1,w2,w3;
    assign a0not = ~a0
    assign b0not = ~b0
    assign w1 = (a0|b0)&(a0not|b0not)
    assign w1not = ~w1
    assign cinot = ~ci
    assign s0 = (w1|ci)&(w1not|cnot)
    assign w2 = (a0&b0)|(ci&w1)
    assign a1not = ~a1
    assign b1not = ~b1
    assign w3 = (a1|b1)&(a1not|b1not)
    assign w2not = ~w2
    assign w3not = ~w3
    assign s1 = (w3|w2)&(w3not|w2not)
    assign c = (a1&b1)|(w2&w3)
endmodule
```

Κυκλωματική υλοποίηση 2bit-Adder:



Γραφική παράσταση:

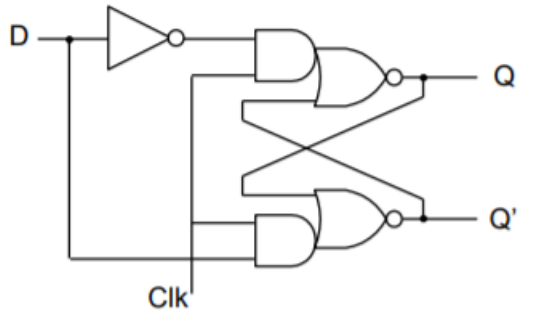


Ερώτηση 2:

Σε αυτό το ερώτημα θα γίνει με χρήση του compiler Verilog είτε με τον layout editor οι τρεις παρακάτω τύποι καταχωρητών:

Μανδαλωτής D:

Ένας μανδαλωτής D εξασφαλίζει ότι οι είσοδοι του flip-flop δεν θα πάνε ποτέ ταυτόχρονα στο 1. Ταυτόχρονα, όμως, δεν πάνε ποτέ ούτε στο 0 με αποτέλεσμα χάνουμε την διατήρηση της προηγούμενης κατάστασης. Όταν το clk είναι 1 τότε η έξοδος ακολουθεί την είσοδο. Όταν το clk είναι 0 τότε η έξοδος έχει την τιμή που είχε η είσοδος όταν το clk άλλαξε από 1 σε 0.



Πίνακας αληθείας D-Latch:

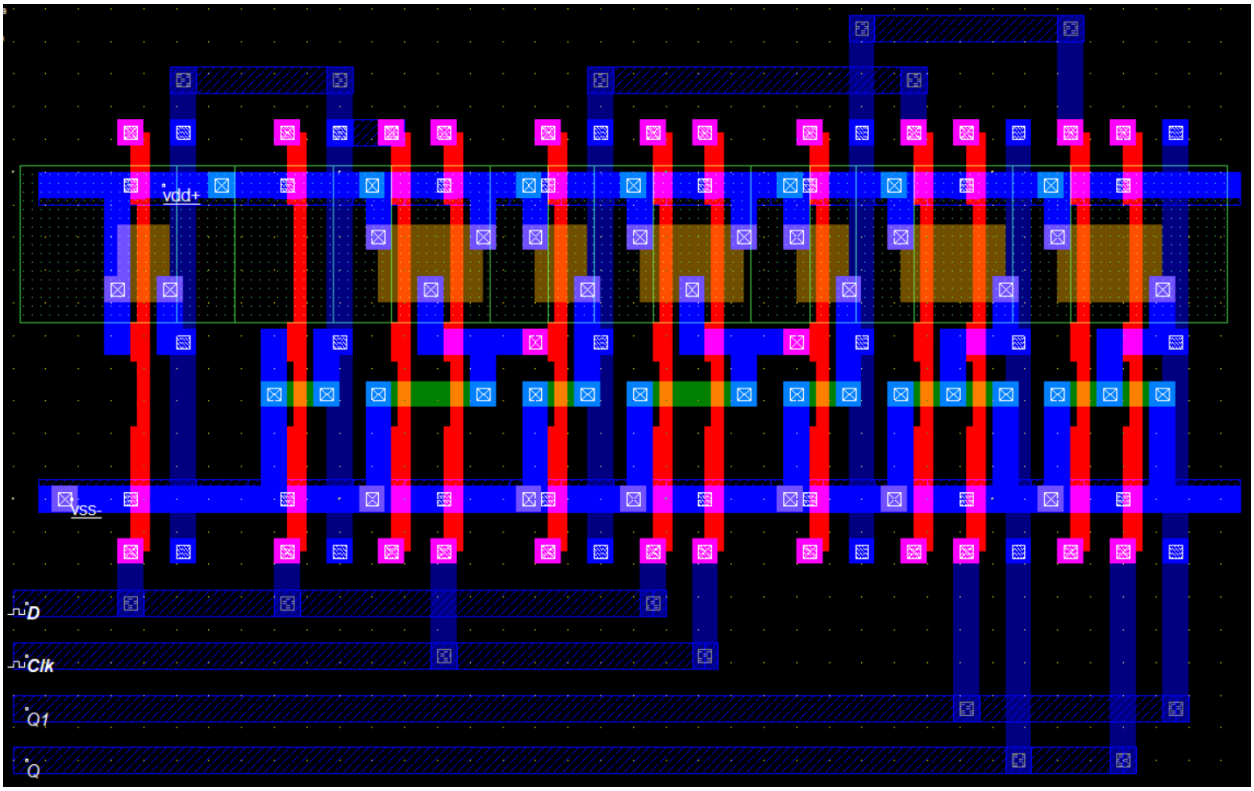
Clk	D	Q	Q’
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

Κώδικας Verilog:

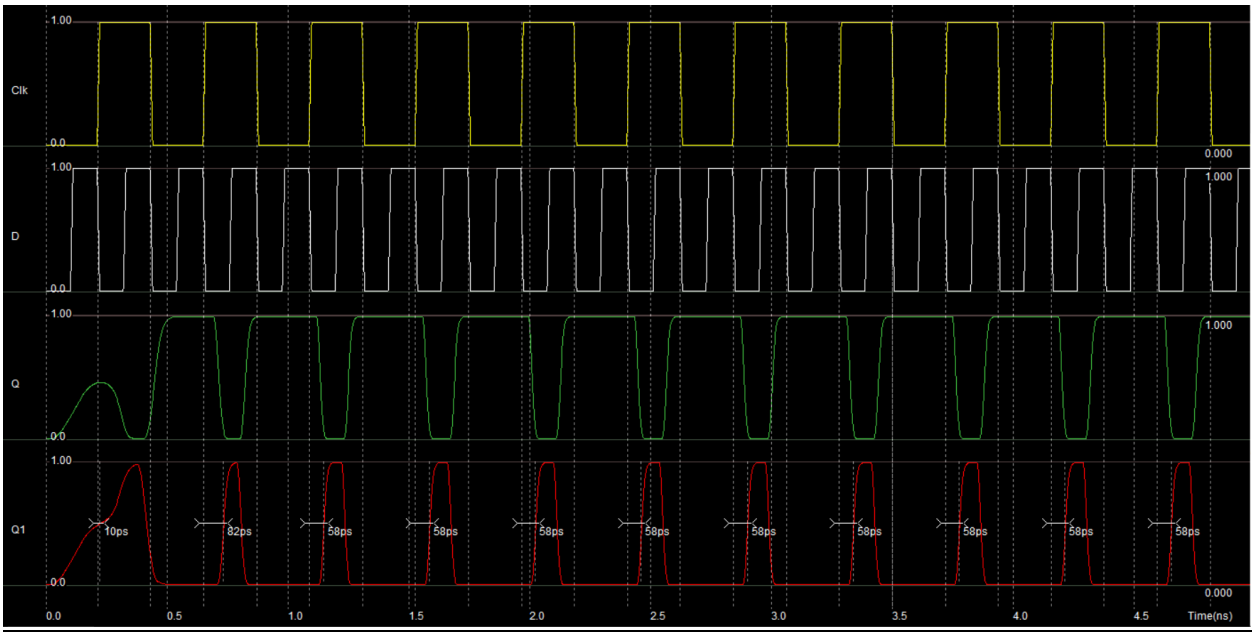
Η υλοποίηση με gate level του μανδαλωτή παρατηρήθηκε ότι βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

```
module Latch(Q,Q1,D,Clk);
   input D,Clk;
        output Q,Q1;
        wire w1,w2,w3;
        pmos p1(w1,VDD,D);
        nmos n1(w1,VSS,D);
        and(w2,w1,Clk);
        and(w3,D,Clk);
        nor(Q,w2,Q1);
   nor(Q1,w3,Q);
endmodule


//Simulate parameters in Verilog Format
always
#200 D=~D;
#400 Clk=~Clk;
```

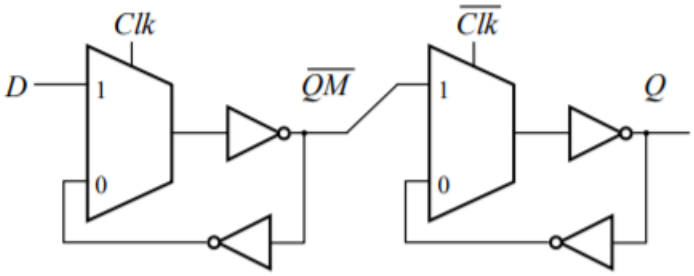
Κυκλωματική υλοποίηση D-Latch:



Γραφική Παράσταση:



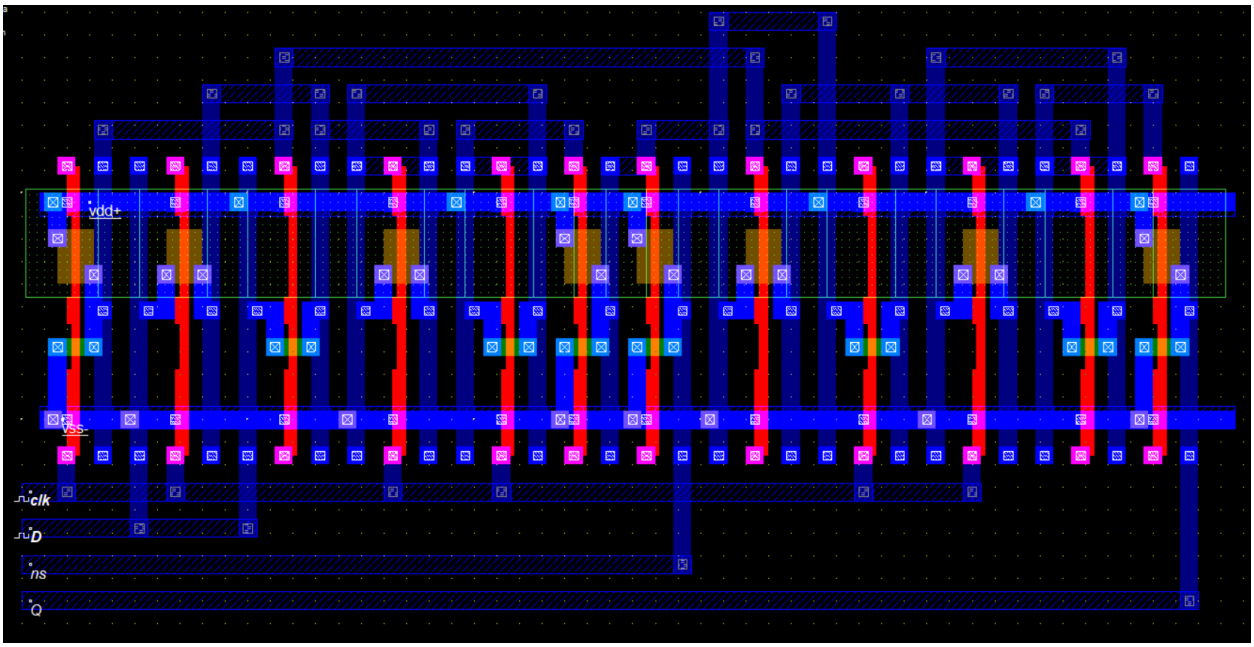
Ακμοπυροδότητος καταχωρητής:



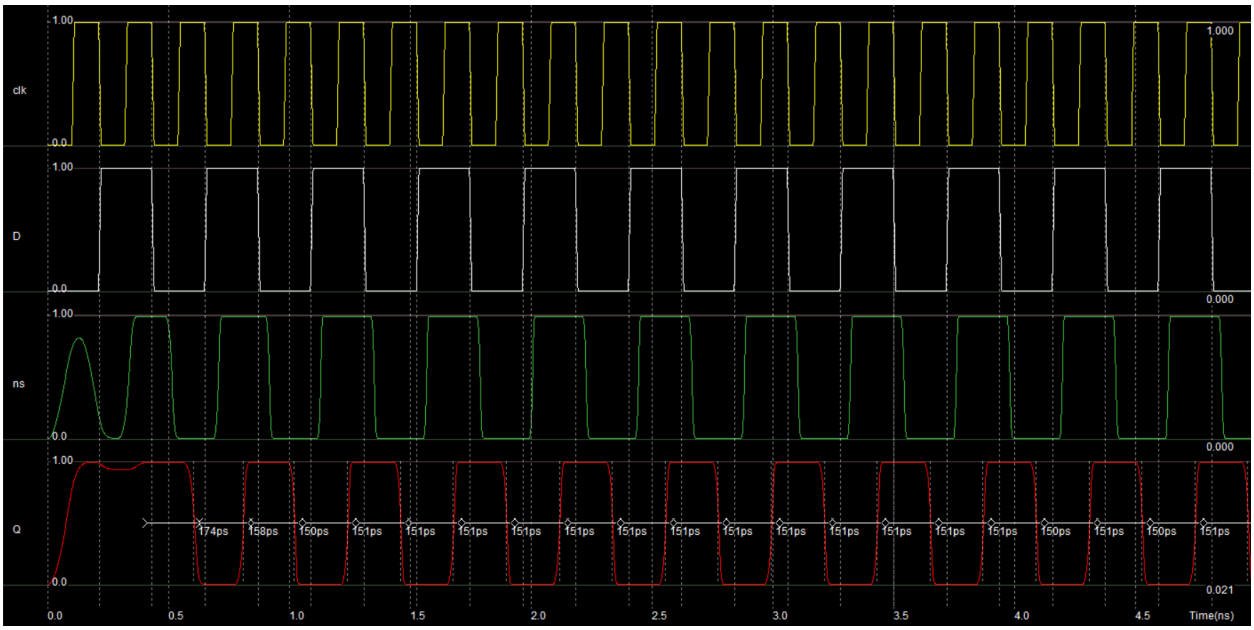
Κώδικας Verilog:

```
module Register(Q,ns,D,clk);
   input D,clk;
   output Q,ns;
   wire w1,w3;
   not(nclk,clk);
   pmos(w1,D,clk);
   nmos(w1,D,nclk);
   pmos(w1,w1,clk);
   nmos(w1,w1,clk);
   not(s,w1);
   not(ns,s);
   pmos(w3,s,nclk);
   nmos(w3,s,clk);
   pmos(w3,w3,clk);
   nmos(w3,w3,nclk);
   not(Q,w3);
 endmodule
```

Κυκλωματική υλοποίηση του ακμοπυροδότητου καταχωρητή:

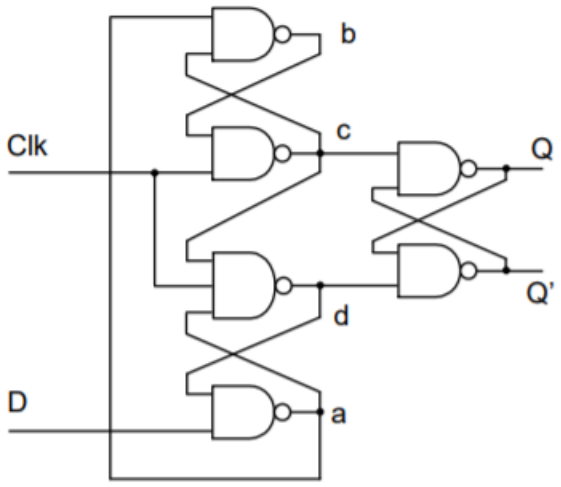


Γραφική Παράσταση:



Θετικά ακμοπυροδότητος καταχωρητής:

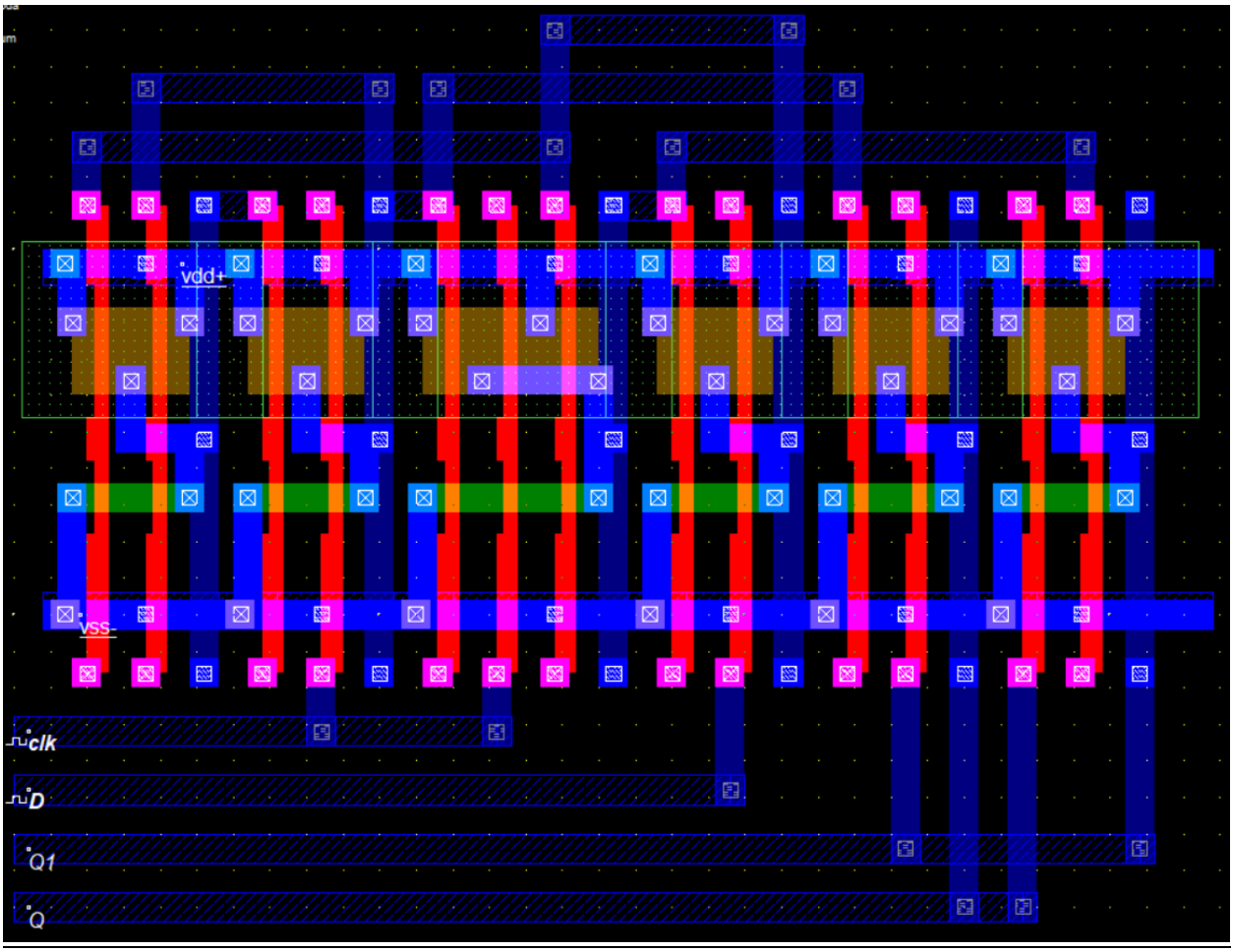
Γενικά, το ακμοπυροδότητο flip-flop αποτελείται ουσιαστικά από τρία βασικά flip-flops και όλες οι αλλαγές στις εξόδους συμβαίνουν σε μία ακμή. Μερικά ακμοπυροδότητα flip-flops αντιδρούν στην αρνητική ακμή του ρολογιού και άλλα στη θετική ακμή, όπου και αντιστοίχως χωρίζονται σε θετικά και αρνητικά.



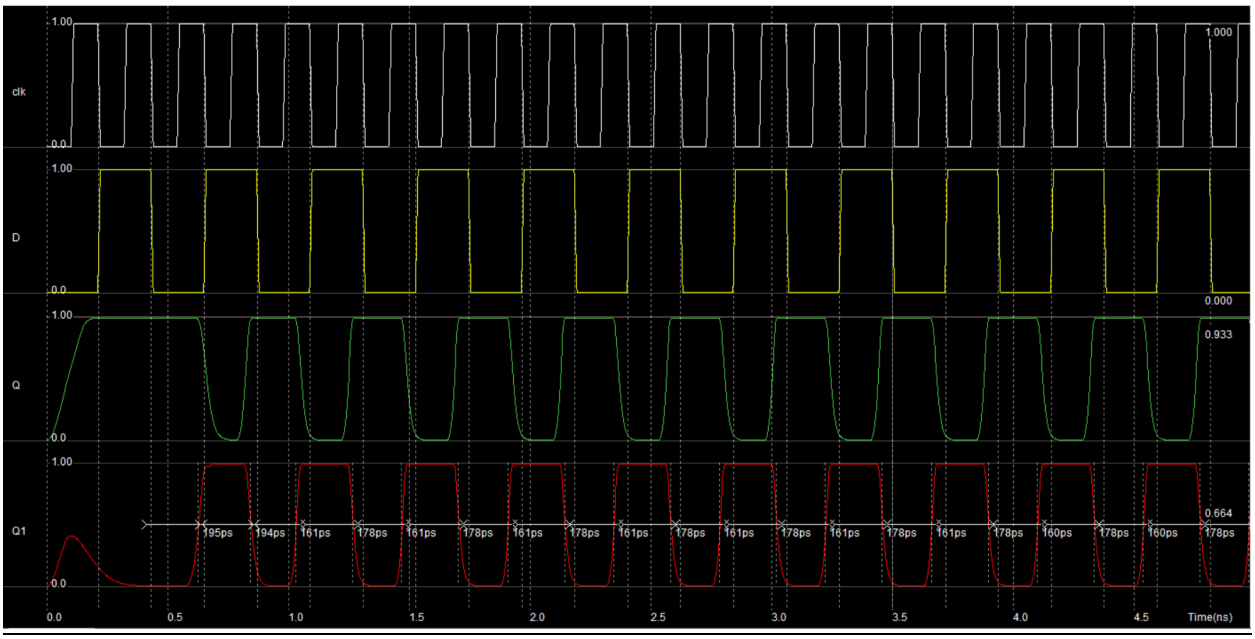
Κώδικας Verilog:

```
module Register2(Q,Q1,D,clk);
   output Q,Q1;
   input D,clk;
   wire a,b,c,d;
   nand(b,a,c);
   nand(c,b,clk);
   nand(d,c,clk,a);
   nand(a,d,D);
   nand(Q,c,Q1);
   nand(Q1,Q,d);
endmodule
```

Κυκλωματική υλοποίηση του θετικά ακμοπυροδότητου καταχωρητή:



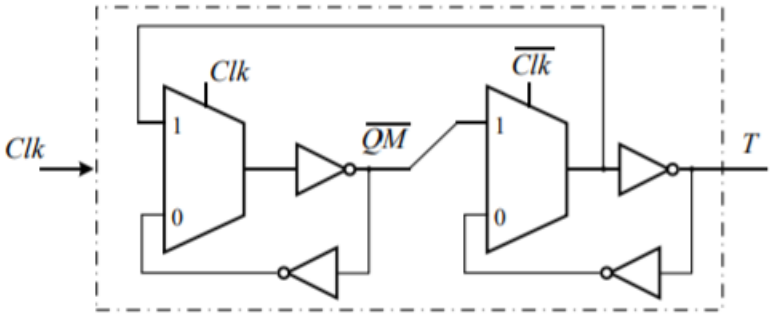
Γραφική Παράσταση:



Ερώτηση 3:

T flip-flop:

Σε αυτό το ερώτημα θα γίνει με τη βοήθεια του compiler Verilog είτε με το layout editor ένα T flip-flop, όπως φαίνεται στο παρακάτω σχήμα.



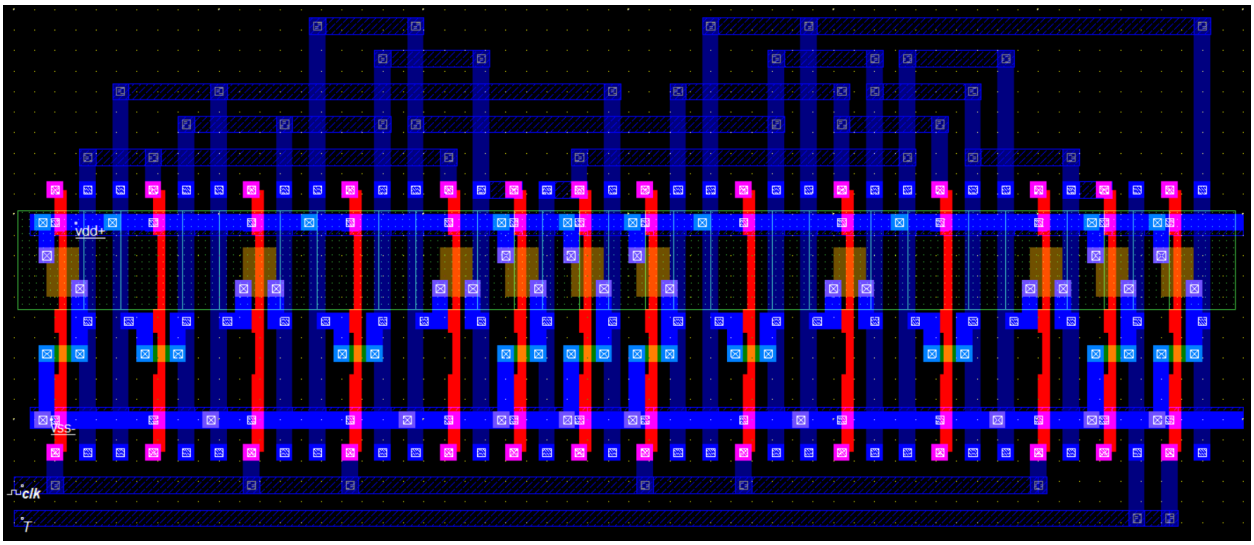
Σε ένα T flip-flop, ως T συμβολίζεται ο κοινός αγωγός των J και K, και λειτουργεί στη θετική ακμή του ρολογιού κάθε φορά συμπληρώνεται η είσοδος.

Κώδικας Verilog:

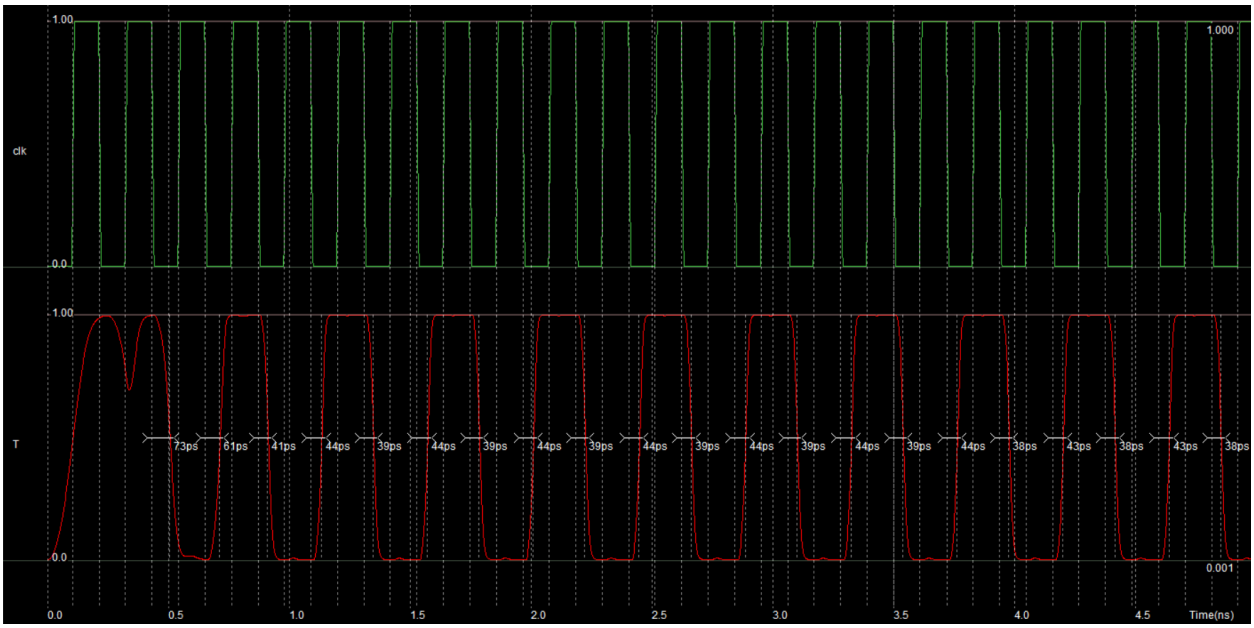
```
module T_flip_flop(T,clk);
  input clk;
  output T;
  wire outm1,outm2,oxiclk,oxiclk2,TM,T1,T2;
  not(oxiclk,clk);
  nmos(outm1,T1,oxiclk);
  pmos(outm1,T1,clk);
  nmos(outm1,outm2,clk);
  pmos(outm1,outm2,oxiclk);
  not(TM,outm1);
  not(T1,TM);
  not(oxiclk2,clk);
  nmos(outm2,T2,clk);
  pmos(outm2,T2,oxiclk2);
  nmos(outm2,TM,oxiclk2);
```

```
    pmos(outm2,TM,clk);
    not(T,outm2);
    not(T2,T);
endmodule
```

Κυκλωματική υλοποίηση T-flip-flop:

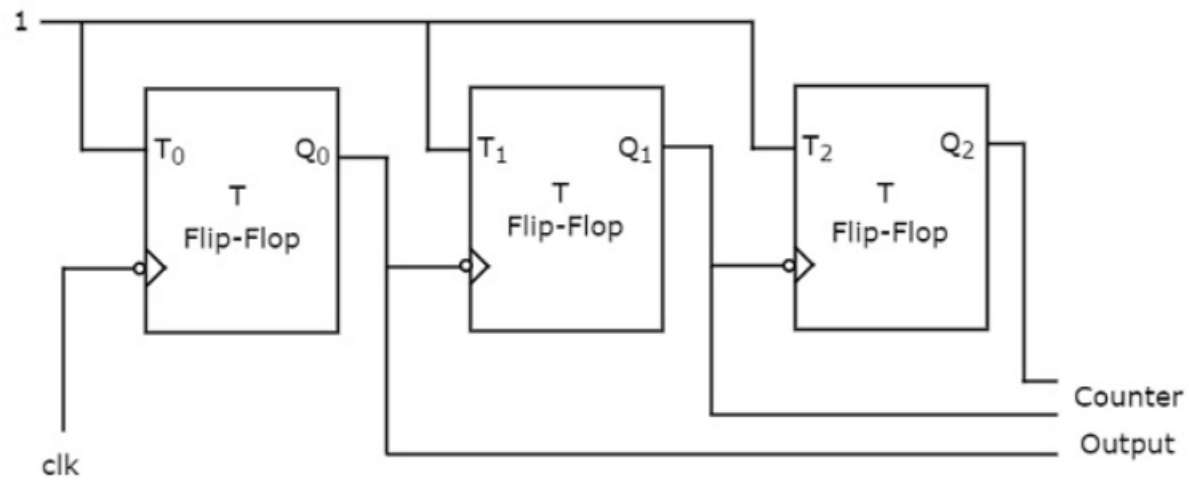


Γραφική παράσταση:



3bit-Counter:

Σε αυτό το ερώτημα θα γίνει με τη βοήθεια του compiler Verilog ένας 3bit-Counter, όπως φαίνεται στο παρακάτω σχήμα.



Ένας 3bit-Counter αποτελείται από 3 T flip-flop, όπως υλοποιήθηκαν στο προηγούμενο ερώτημα, και λειτουργεί σαν μετρητής αριθμών δυαδικού συστήματος δηλαδή:

0 → 000

1 → 001

2 → 010

3 → 011

4 → 100

5 → 101

6 → 110

7 → 111

CLK	T2(MSB)	T1	T0(LSB)
0	0	0	0
1	0	0	1
0	0	1	0
1	0	1	1
0	1	0	0
1	1	0	1
0	1	1	0
1	1	1	1

Κώδικας Verilog:

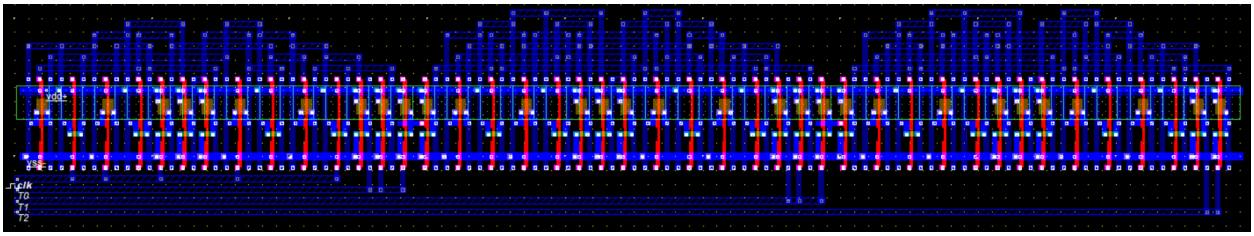
Η υλοποίηση με gate level παρατηρήθηκε ότι η υλοποίηση του 3bit-Counter βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

```
module 3bit_counter(clk,T0,T1,T2);
  input clk;
  output T0,T1,T2;
  wire [2:0]q1,q2,x,y,nclk,qm;
  pmos(q2[0],q1[0],nclk[0]);
  nmos(q2[0],q1[0],clk);
  pmos(x[0],q1[0],clk);
  nmos(x[0],q1[0],nclk[0]);
  not(nclk[0],clk);
  not(qm[0],q1[0]);
  not(x[0],qm[0]);
  pmos(qm[0],q2[0],clk);
  nmos(qm[0],q2[0],nclk[0]);
  pmos(y[0],q2[0],nclk[0]);
  nmos(y[0],q2[0],clk);
  not(T0,q2[0]);
  not(y[0],T0);
```

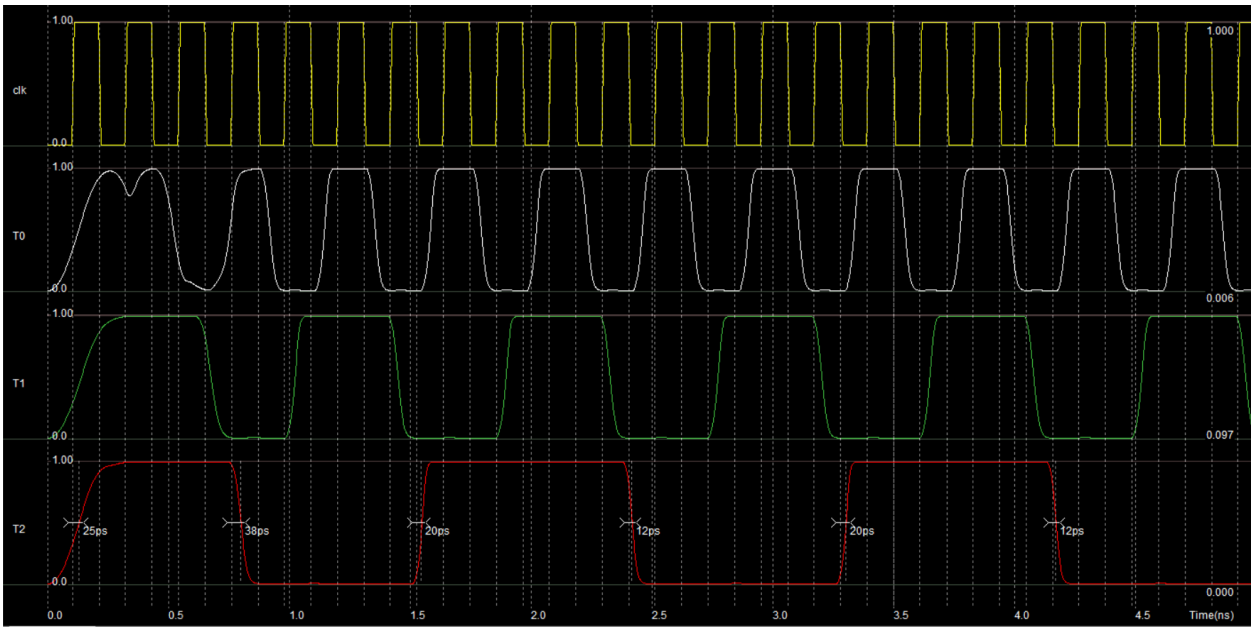
```
  wire clk1;
  assign clk1=T0;
  pmos(q2[1],q1[1],nclk[1]);
  nmos(q2[1],q1[1],clk1);
  pmos(x[1],q1[1],clk1);
  nmos(x[1],q1[1],nclk[1]);
  not(nclk[1],clk1);
  not(qm[1],q1[1]);
  not(x[1],qm[1]);
  pmos(qm[1],q2[1],clk1);
  nmos(qm[1],q2[1],nclk[1]);
  pmos(y[1],q2[1],nclk[1]);
  nmos(y[1],q2[1],clk1);
  not(T1,q2[1]);
  not(y[1],T1);
```

```
  wire clk2;
  assign clk2=T1;
  pmos(q2[2],q1[2],nclk[2]);
  nmos(q2[2],q1[2],clk2);
  pmos(x[2],q1[2],clk2);
  nmos(x[2],q1[2],nclk[2]);
  not(nclk[2],clk2);
  not(qm[2],q1[2]);
  not(x[2],qm[2]);
  pmos(qm[2],q2[2],clk2);
  nmos(qm[2],q2[2],nclk[2]);
  pmos(y[2],q2[2],nclk[2]);
  nmos(y[2],q2[2],clk2);
  not(T2,q2[2]);
  not(y[2],T2);
endmodule
```

Κυκλωματική υλοποίηση 3bit-Counter:

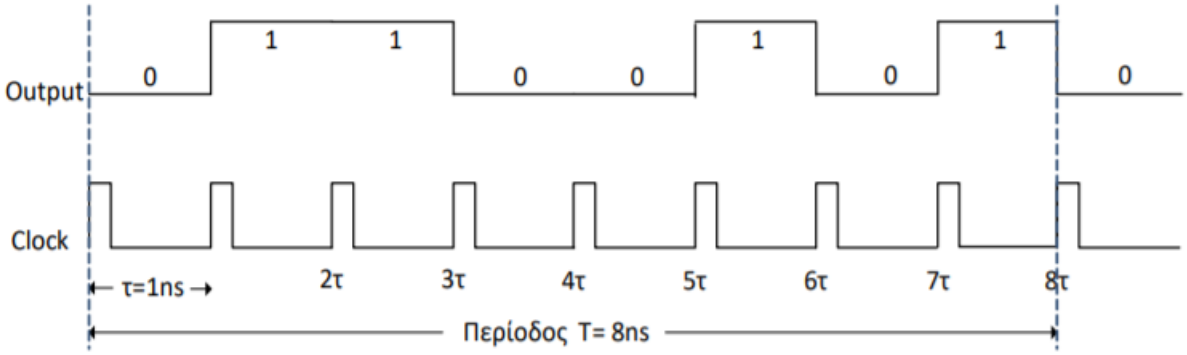


Γραφική Παράσταση:



Ερώτηση 4:

Σε αυτό το ερώτημα θα επιτευχθεί με χρήση κώδικα Verilog, η ακόλουθη έξοδος.



Πίνακας Αληθείας Κυματομορφής:

Q0	Q1	Q2	OUTPUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Χρησιμοποιώντας τον 3bit-counter του προηγούμενου ερωτήματος υλοποιήσαμε για την έξοδο την εξής λογική συνάρτηση: F_{OUT} = Q1'Q2 + Q0Q2, όπου Q0,Q1,Q2 οι έξοδοι του 3bit-counter.

Από τα δεδομένα της εκφώνησης, το σήμα πρέπει να εντοπίζεται μόνο στα θετικά μέτωπα του ρολογιού, για αυτό το λόγο χρησιμοποιούμε το θετικά ακμοπυροδότητο D flip-flop. Σε αυτό αποθηκεύεται η τιμή του start/stop. Η έξοδός του μαζί με τις τιμές των Q0,Q1,Q2 αθροίζονται λογικά (σε δυαδικό σύστημα).
Επιπλέον, πρέπει να εξασφαλίσουμε ότι ακόμα κι αν το σήμα stop έρθει πριν το τέλος της τρέχουσας περιόδου της κυματομορφής, αυτή θα ολοκληρωθεί. Για αυτό το λόγο, το αποτέλεσμα εισάγεται ως είσοδος σε μια πύλη AND με την άλλη είσοδο να είναι το clk. Η έξοδος της AND εισάγεται ως ρολόι στο πρώτο T flip-flop του μετρητή. Μόνο όταν το start/stop είναι 0 και τα Q0,Q1,Q2 είναι στο 0 το αποτέλεσμα της πύλης AND γίνεται μηδέν και ο μετρητής σταματά να μετράει. Έτσι εξασφαλίζεται πως ακόμα και αν το σήμα start/stop φτάσει πριν την λήξη του παλμού αυτός πρώτα θα ολοκληρωθεί και μετά θα τεθεί στο λογικό 1.

Κώδικας Verilog:

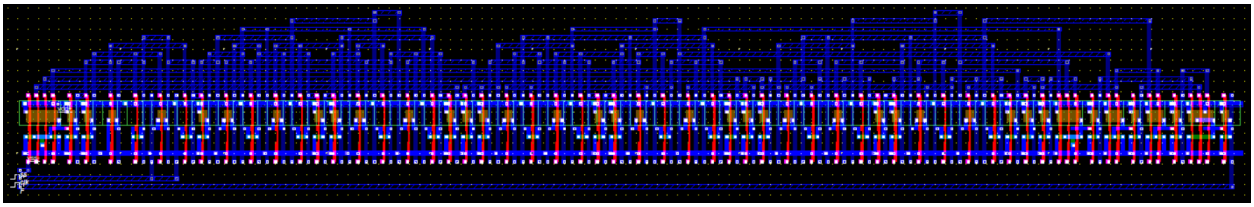
Η υλοποίηση με gate level παρατηρήθηκε ότι η υλοποίηση της κυματομορφής βγήκε ορθή και οι γραφικές παραστάσεις βγάζουν αποτελέσματα, που επαληθεύονται από τον πίνακα αληθείας.

```
module ss counter(F,ss,clk);
input ss,clk;
output F;
wire [2:0]q1,q2,q3,q4,q5;
wire T0,T1,T2,nT0,nT1,nT2;
wire nmux_s,mux_s,Y,nY,or1,or2,or3;
or(mux_s,ss,T0,T1,T2);
not(nmux_s,mux_s);
pmos(Y,VSS,mux_s);
nmos(Y,VSS,nmux_s);
pmos(Y,clk,nmux_s);
nmos(Y,clk,mux_s);
not(nY,Y);
pmos(q1[0],Y,q4[0]);
nmos(q1[0],nY,q4[0]);
pmos(q1[0],nY,q3[0]);
nmos(q1[0],Y,q3[0]);
not(q2[0],q1[0]);
not(q3[0],q2[0]);
pmos(q4[0],nY,q2[0]);
nmos(q4[0],Y,q2[0]);
pmos(q4[0],Y,q5[0]);
nmos(q4[0],nY,q3[0]);
not(T0,q4[0]);
not(q5[0],T0);
not(nT0,T0);
pmos(q1[1],T0,q4[1]);
nmos(q1[1],nT0,q4[1]);
pmos(q1[1],nT0,q3[1]);
nmos(q1[1],T0,q3[1]);
not(q2[1],q1[1]);
not(q3[1],q2[1]);
pmos(q4[1],nT0,q2[1]);
nmos(q4[1],T0,q2[1]);
pmos(q4[1],T0,q5[1]);
nmos(q4[1],nT0,q3[1]);
not(T1,q4[1]);
not(q5[1],T1);
```

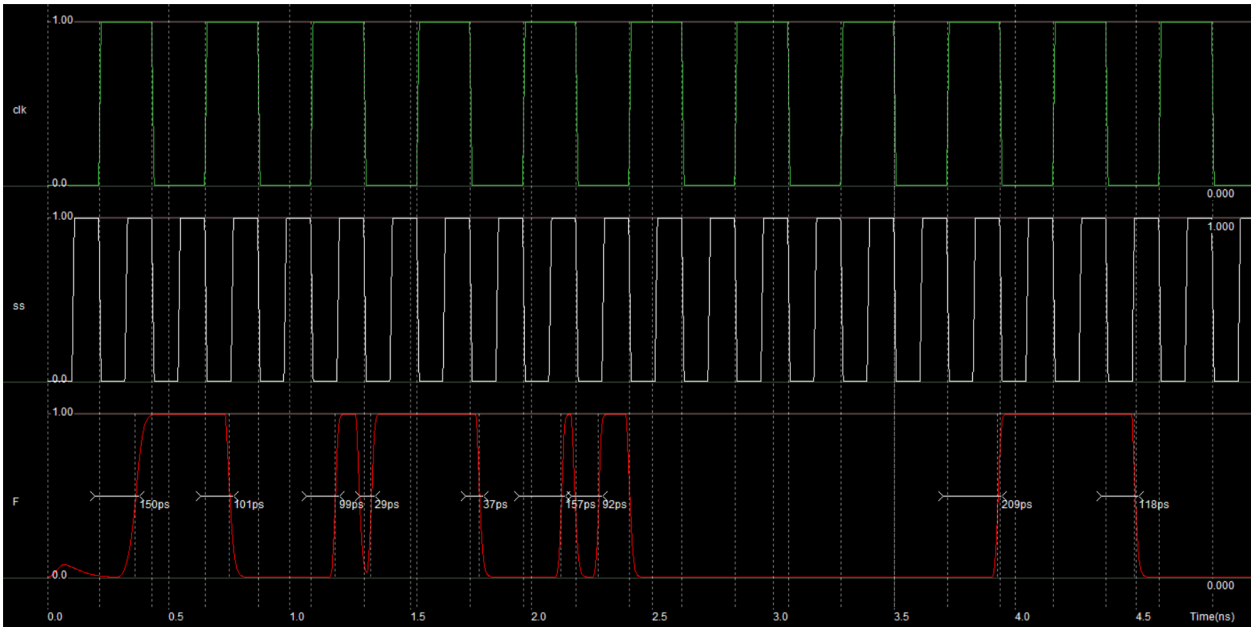
```
not(nT1,T1);
pmos(q1[2],T1,q4[2]);
nmos(q1[2],nT1,q4[2]);
pmos(q1[2],nT1,q3[2]);
nmos(q1[2],T1,q3[2]);
not(q2[2],q1[2]);
not(q3[2],q2[2]);

pmos(q4[2],nT1,q2[2]);
nmos(q4[2],T1,q2[2]);
pmos(q4[2],T1,q5[2]);
nmos(q4[2],nT1,q3[2]);
not(T2,q4[2]);
not(q5[2],T2);
```

```
not(nT2,T2);
or(or1,nT0,T1,nT2);
or(or2,T0,T2);
or(or3,nT1,T2);
and(F,or1,or2,or3);
endmodule
```

Κυκλωματική υλοποίηση Κυματομορφής:



Γραφική Παράσταση:



Τομή σε κρίσιμη περιοχή: