



Πανεπιστήμιο Ιωαννίνων

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Προπτυχιακό Μάθημα: «Ψηφιακή Σχεδίαση II»

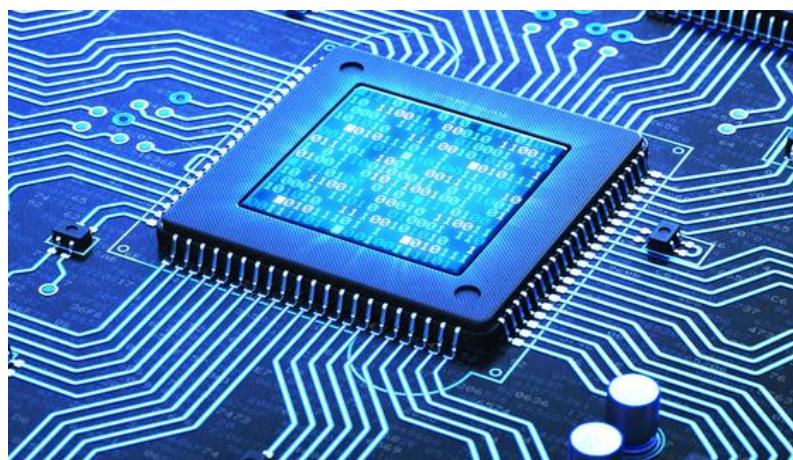
Ομάδα 15Α

Μέλη/Α.Μ. :

Αναστασόπουλος Δημήτριος / 3179

Κρομμύδας Γεώργιος / 3260

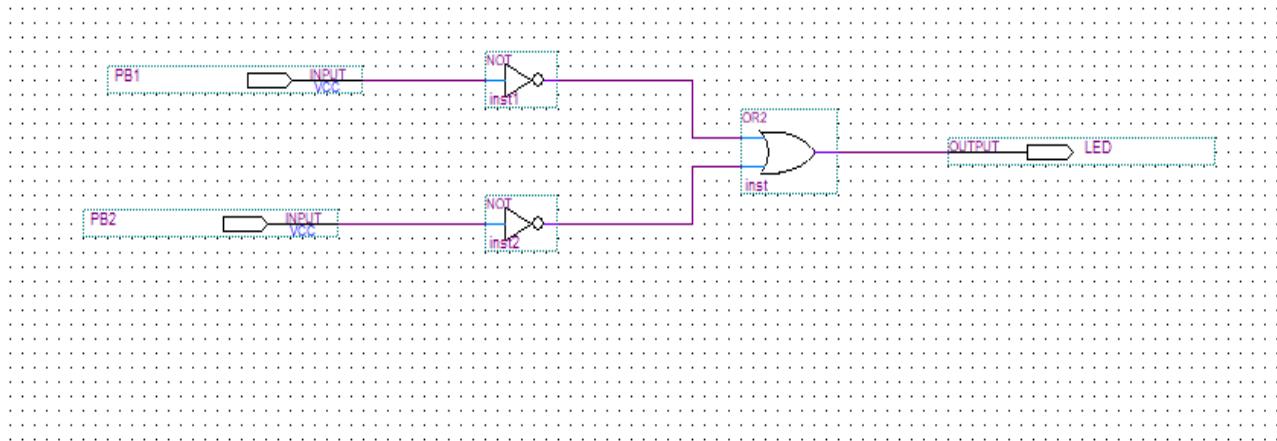
Χαντζής Γεράσιμος / 3360



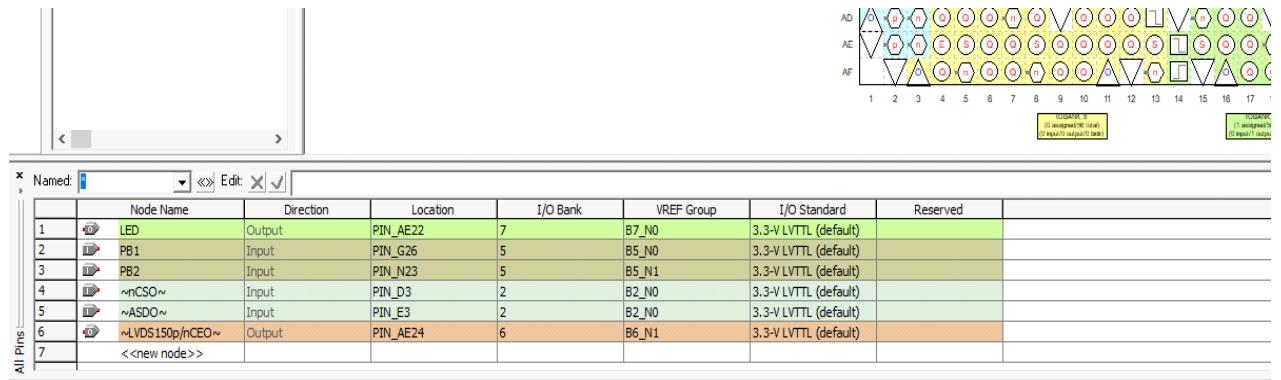
1^η Εργαστηριακή Άσκηση: Περιβάλλον Εργασίας και απλά ψηφιακά κυκλώματα

Μέρος 1^ο:

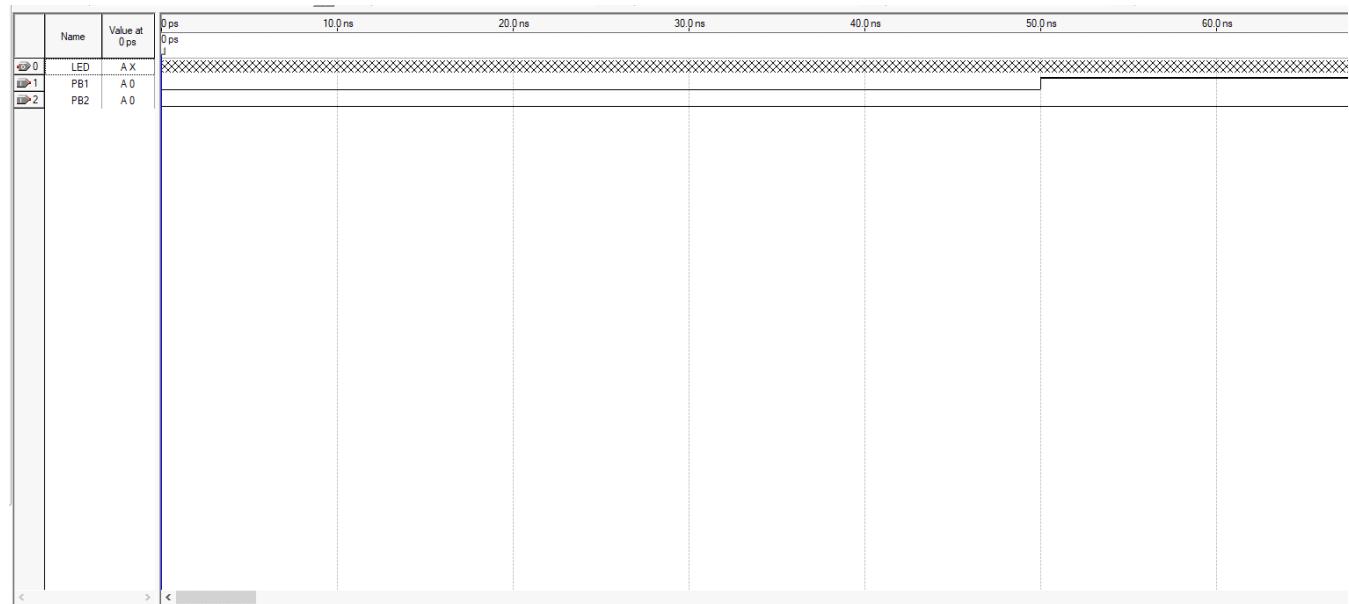
Αρχικά επιλέξαμε 'file --> New Project wizard' για να βρουμε την βιβλιοθήκη 'Cyclone II'. Εφόσον έχουμε ονοματήσει το πρόγραμμα εργαζόμαστε στο περιβάλλον σχεδιασμού του κυκλώματος. Για το πρώτο κομμάτι χρησημοποιήσαμε 2 inputs, 2 not gates, 1 or gate και 1 output.



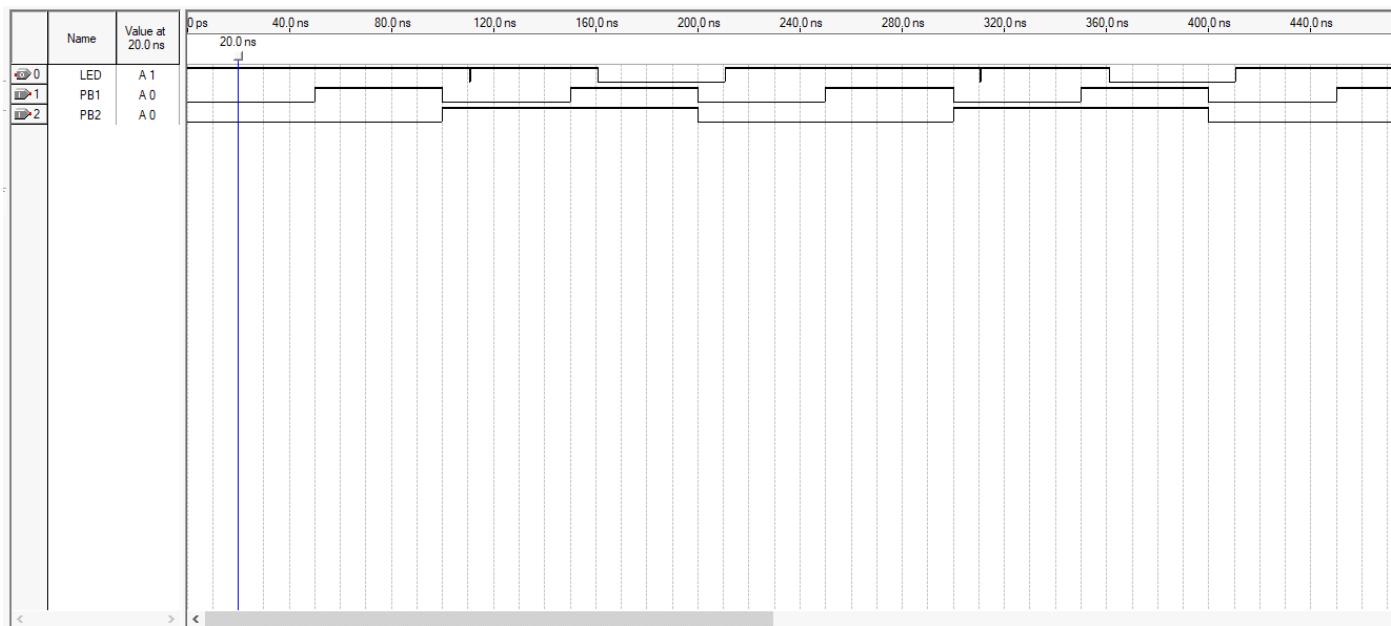
Στη συνέχεια επιλέξαμε 'Processing --> Start Compilation' για να κάνουμε compile το σχήμα μας. Εφόσον δεν έχουμε error επιλέγουμε 'Assigments--> Pin Planner' και χρησιμοποιούμε τα κατάλληλα pins.



Επαναλαμβάνουμε το compile. Μετά επιλέγουμε 'File --> New... --> Vector Waveform File' και ανοίγει το simulation. Έπειτα επιλέγουμε στις εισόδους τις κατάλληλες τιμές/συχνότητες που θέλουμε να γίνονται οι αλλαγές.



Μετά επιλέγουμε 'Processing --> Simulator tool' για να πάρουμε το waveform.

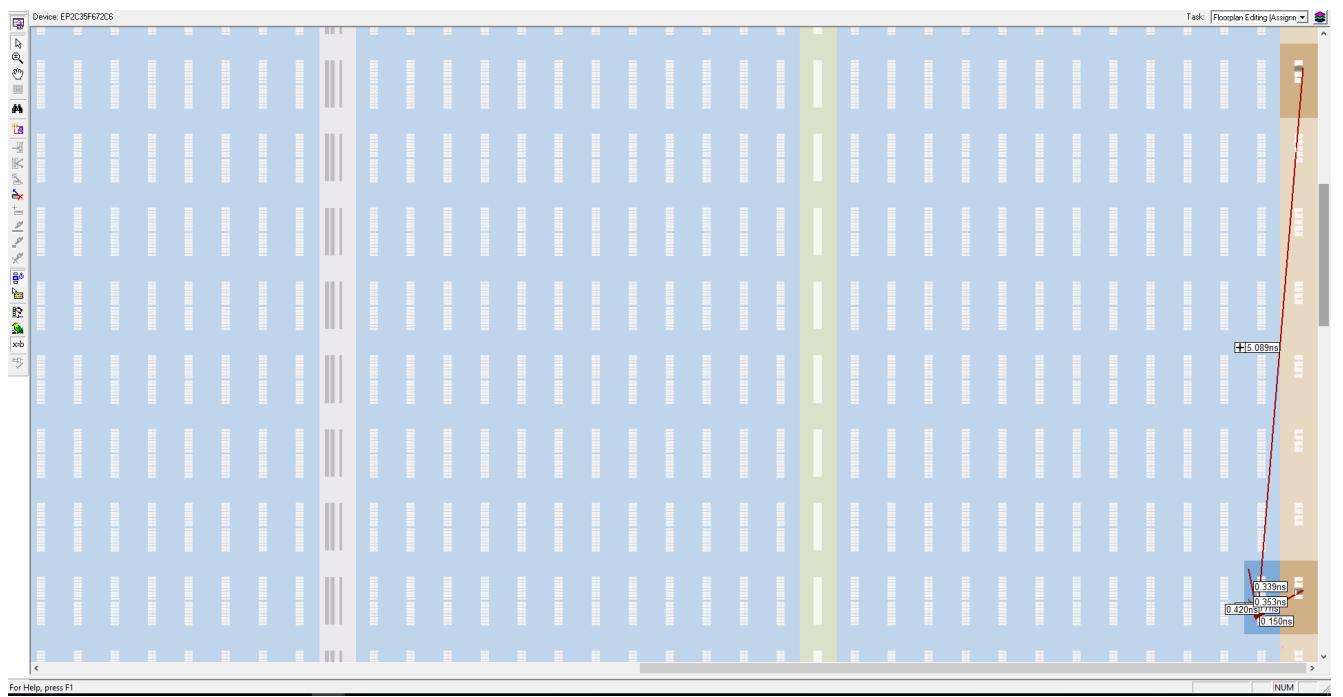


Τιμές : LED=1

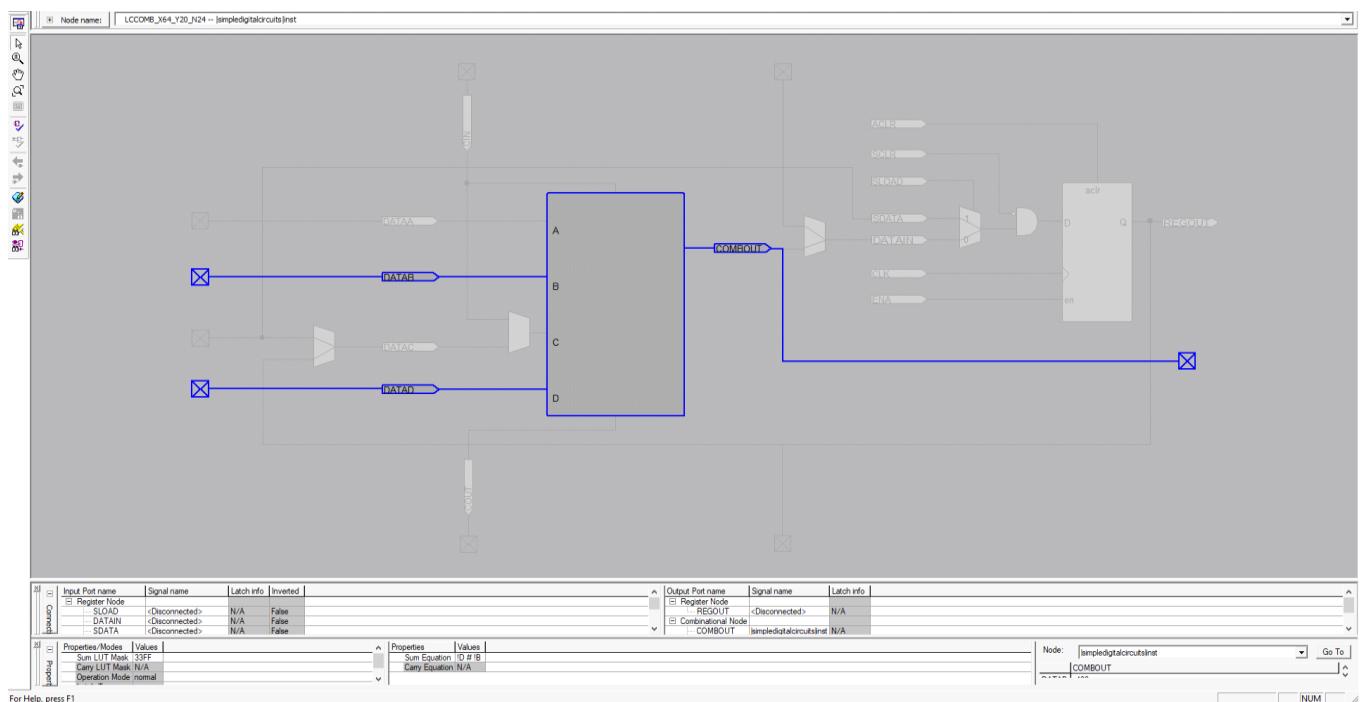
PB1= 50ns

PB2= 100ns

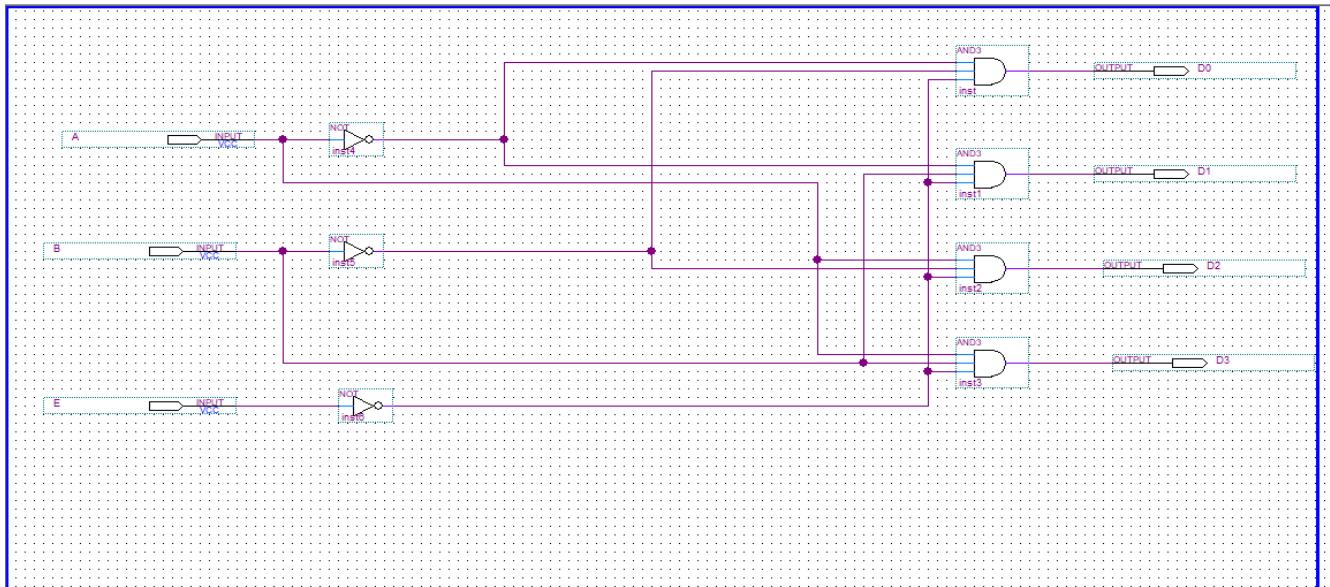
Έπειτα επιλέγουμε 'Assignments --> Back-Annotate Assignments' και στο αναδυόμενο παράθυρο πατάμε 'OK'.



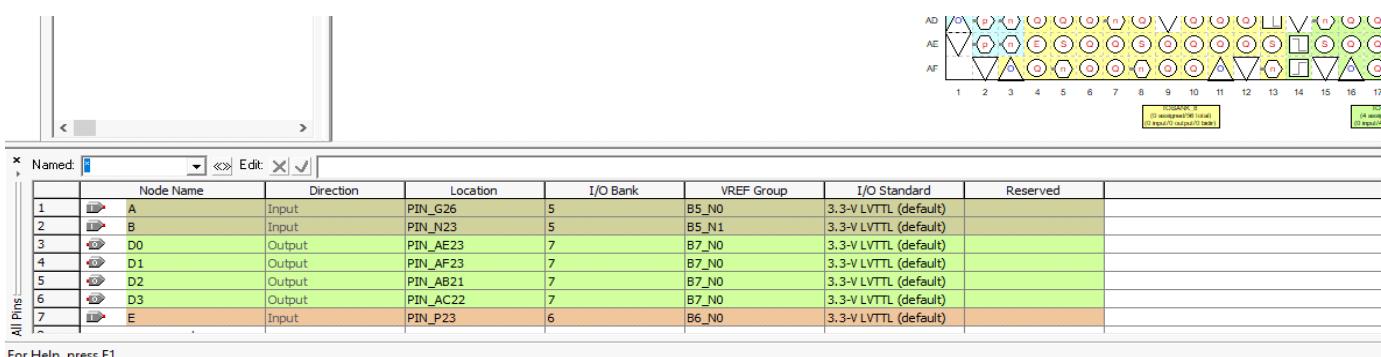
Τέλος επιλέγουμε 'Tools --> Chip Planner' για να δούμε το chip. Πατώντας σε ένα καφέ πλαίσιο του chip εμφανίζεται το εσωτερικό του chip.

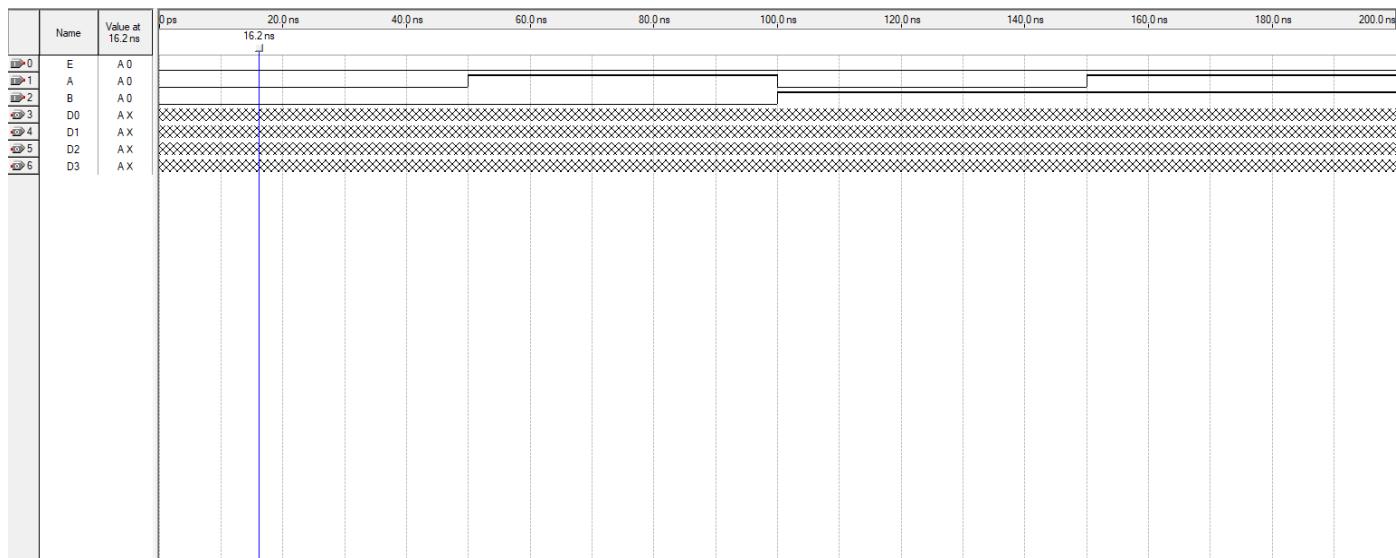


Μέρος 2^ο (a) : Decoder



Σε αυτό το μέρος χρησιμοποιήσαμε 3 inputs, 3 NOT gates, 4 AND gates και 4 outputs.



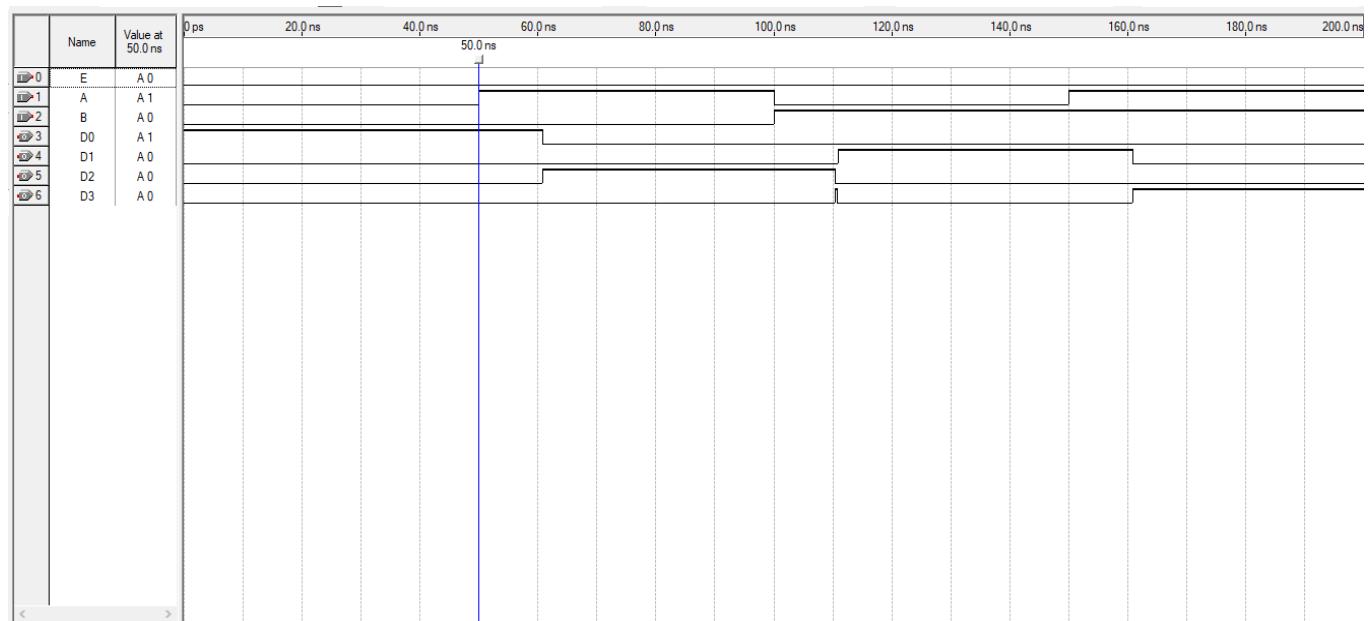


Θέτουμε τιμές στις εισόδους : $A = 50 \text{ ns}$

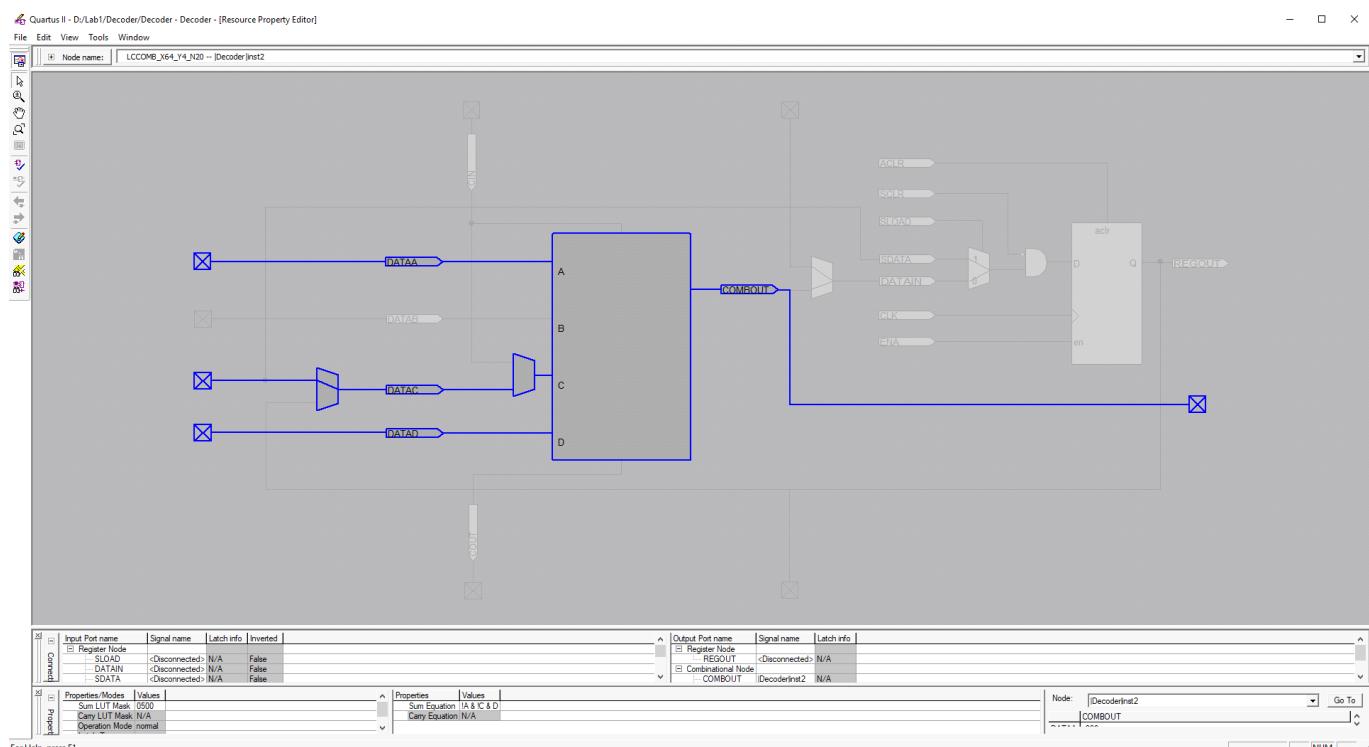
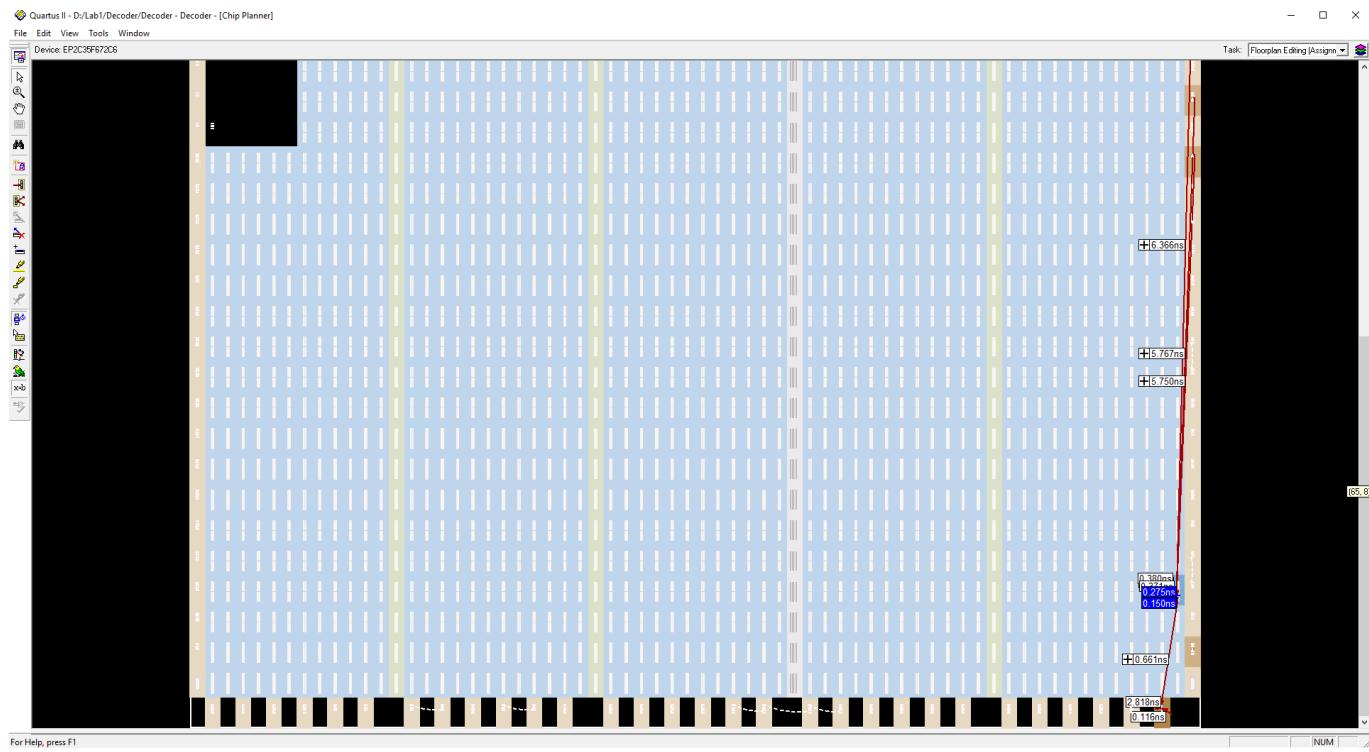
$B = 100 \text{ ns}$

$E = 0$

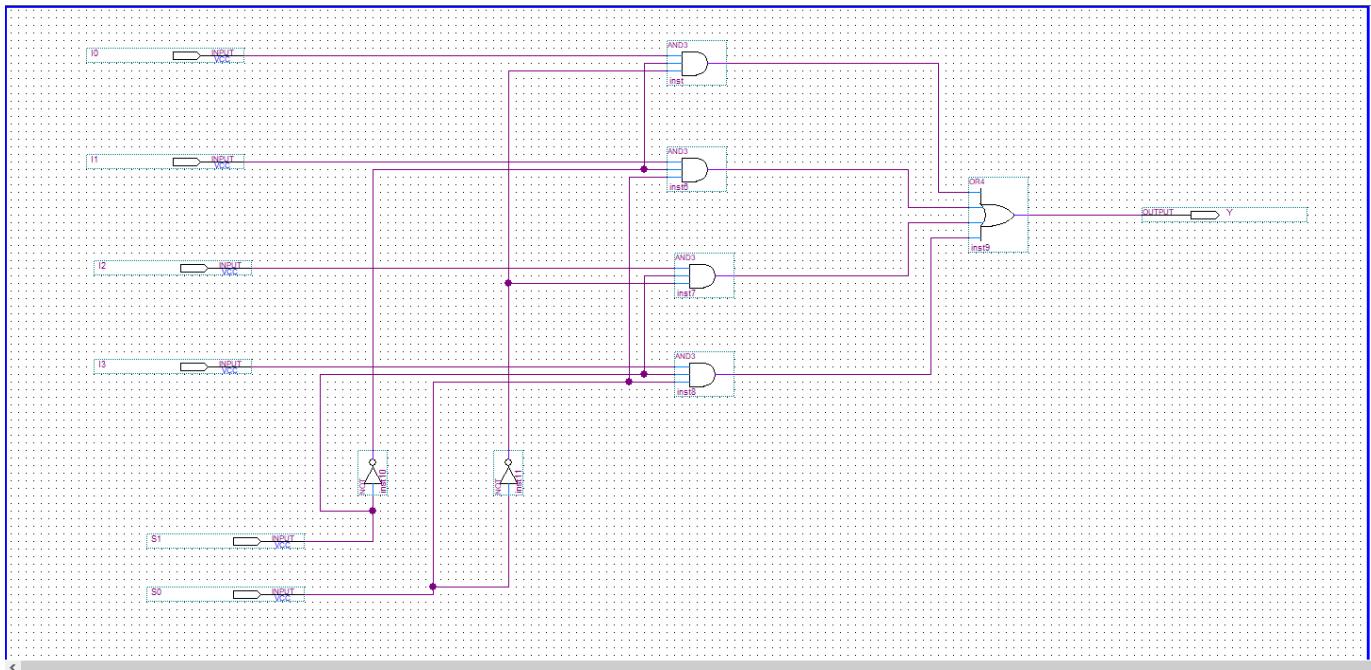
Έτσι έχουμε :



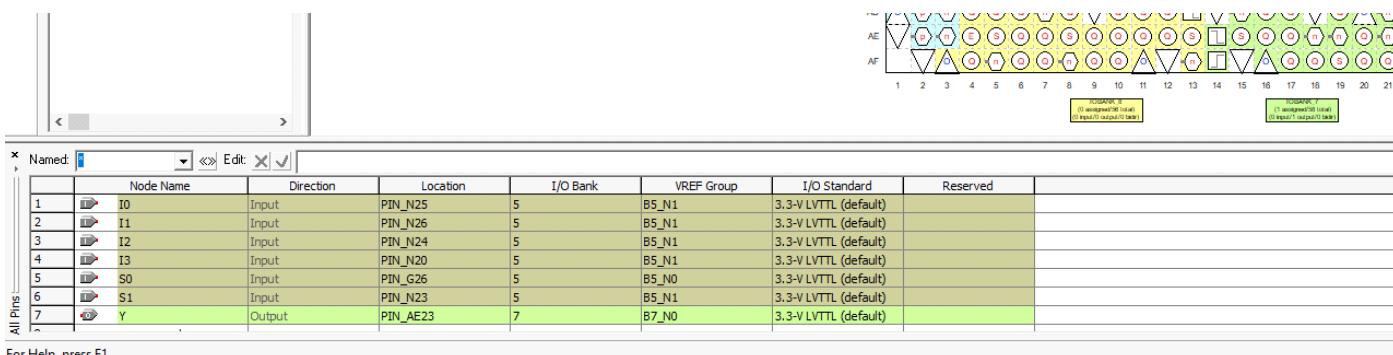
Τέλος το chip του Decoder είναι:

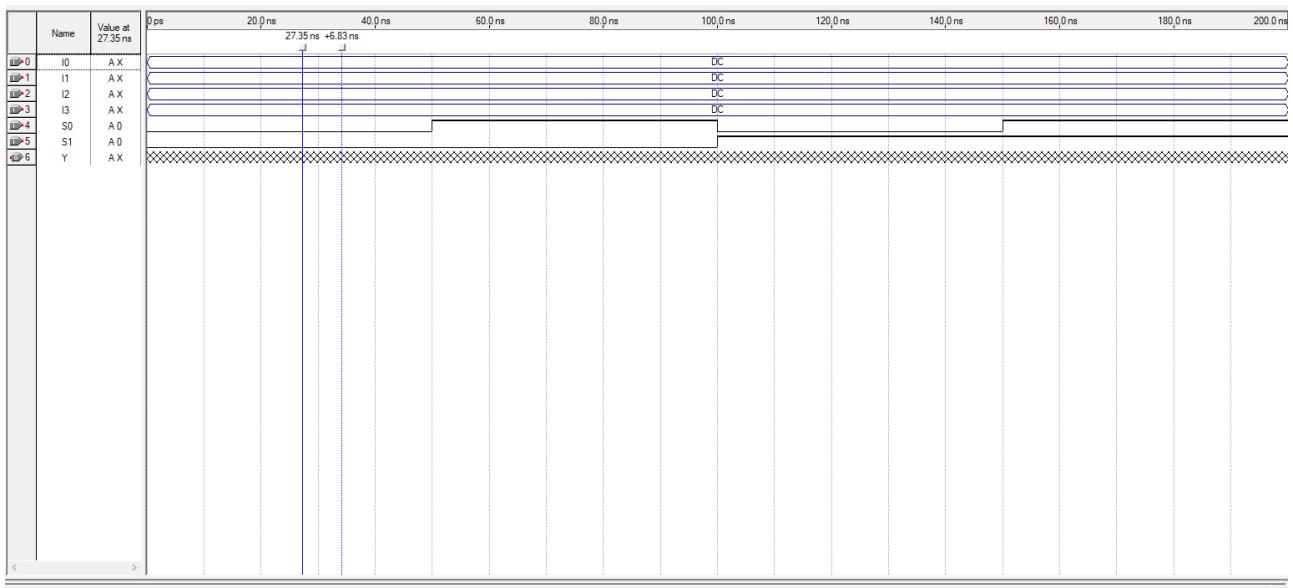


Μέρος 2^ο (β) : Multiplexer



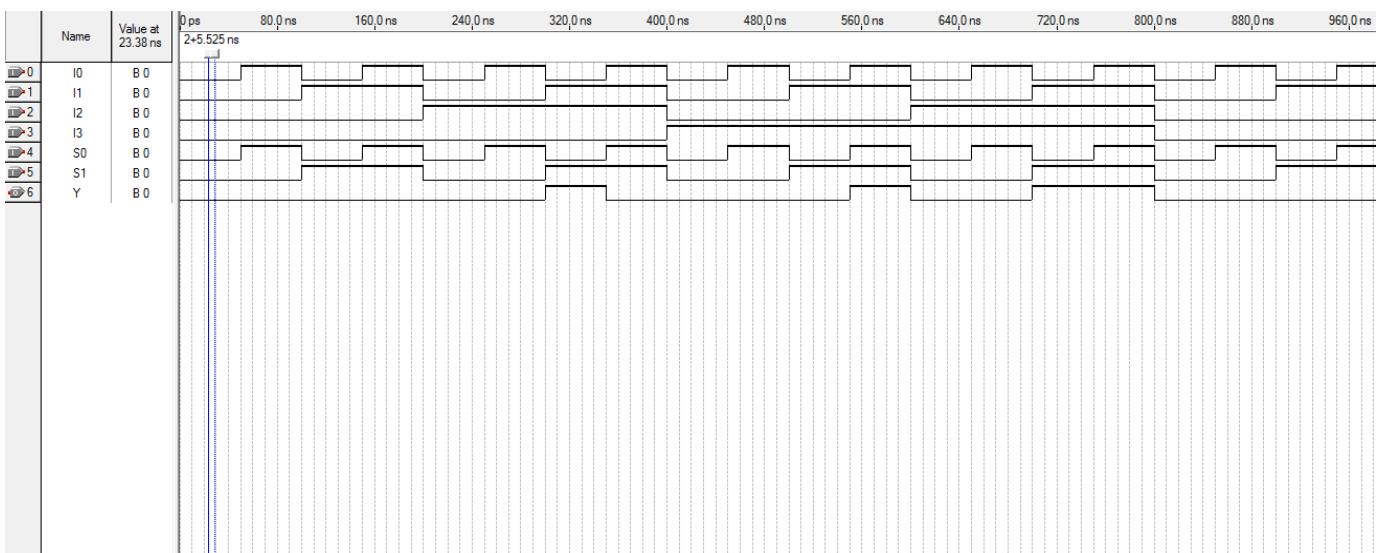
Σε αυτό το μέρος χρησιμοποιήσαμε 6 inputs, 2 NOT gates, 4 AND gates(3 inputs), 1 OR gate(4 inputs) και 1 output.



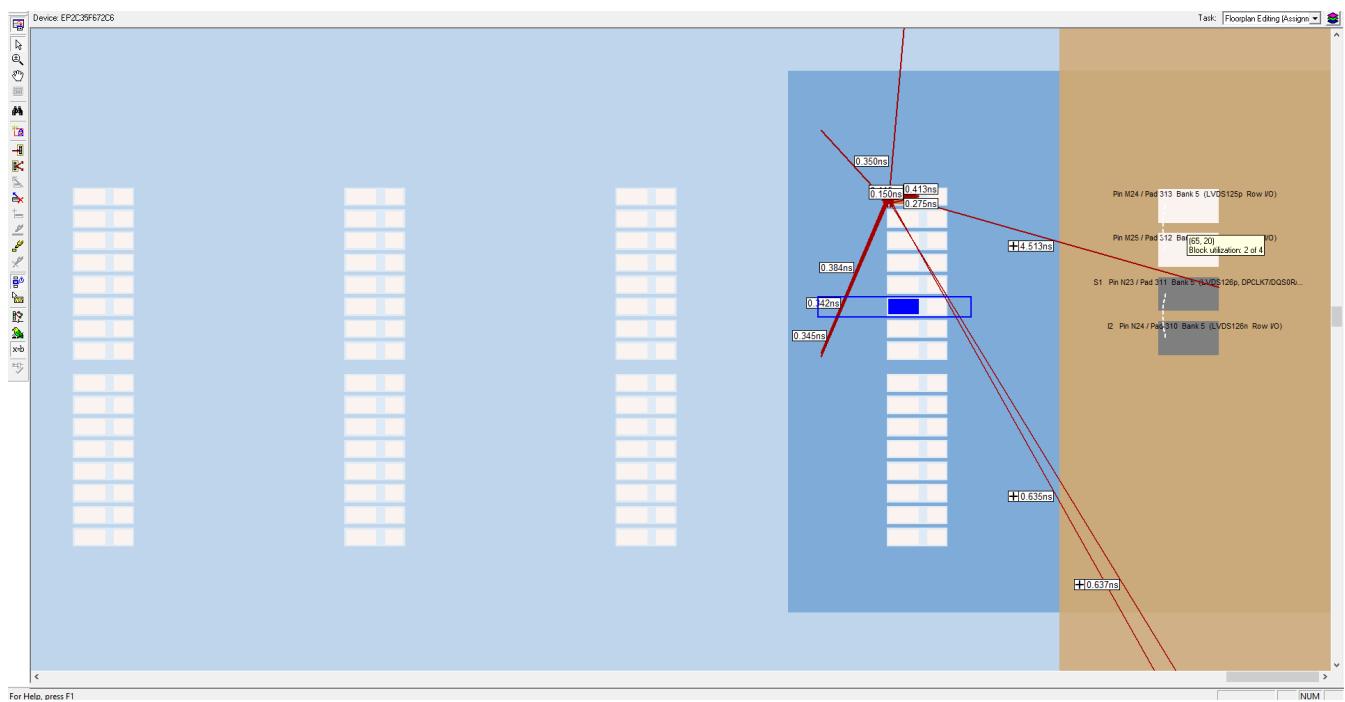


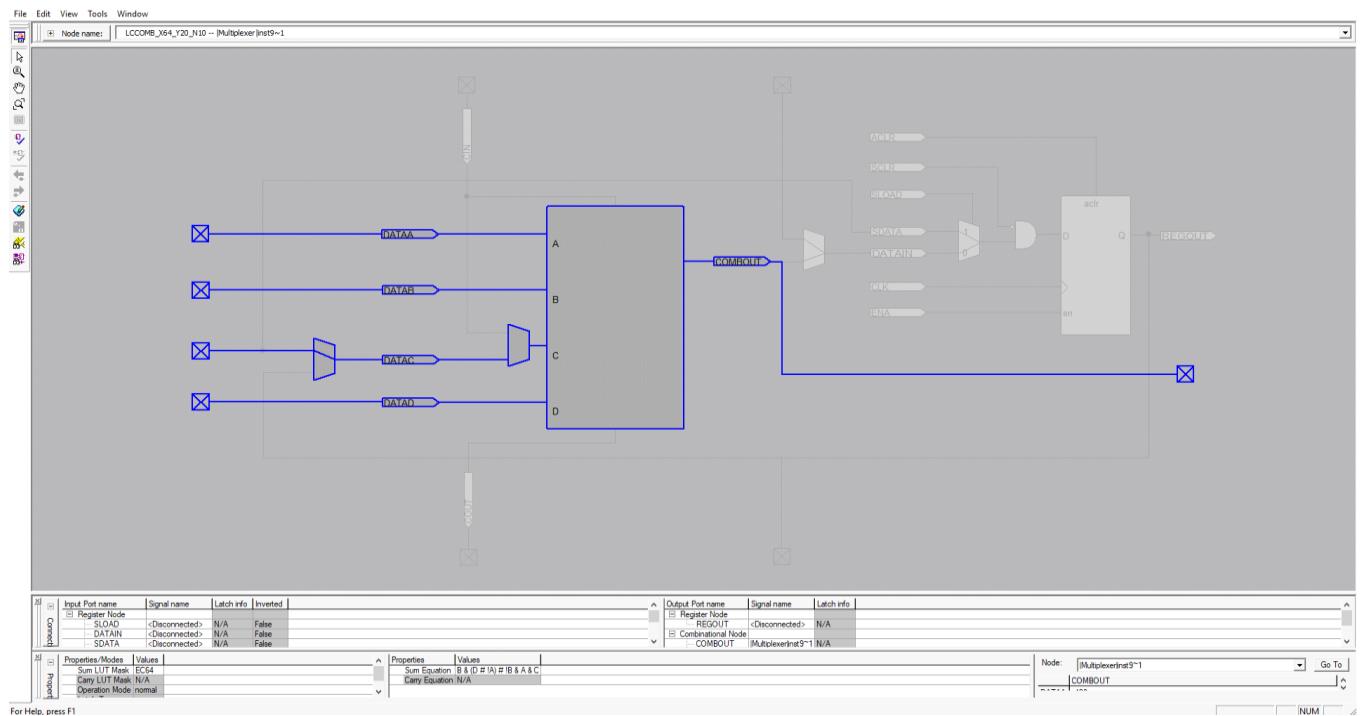
Τώρα θα θέσουμε στην είσοδο I0 την τιμή 1 ανά 50 ns, στην I1 ανά 100ns, στην I2 ανά 200ns και στην I3 ανά 400ns. Επίσης, τις εισόδους S0 και S1 θα τις θέσουμε στα 50ns και 100ns αντίστοιχα.

Έτσι έχουμε :

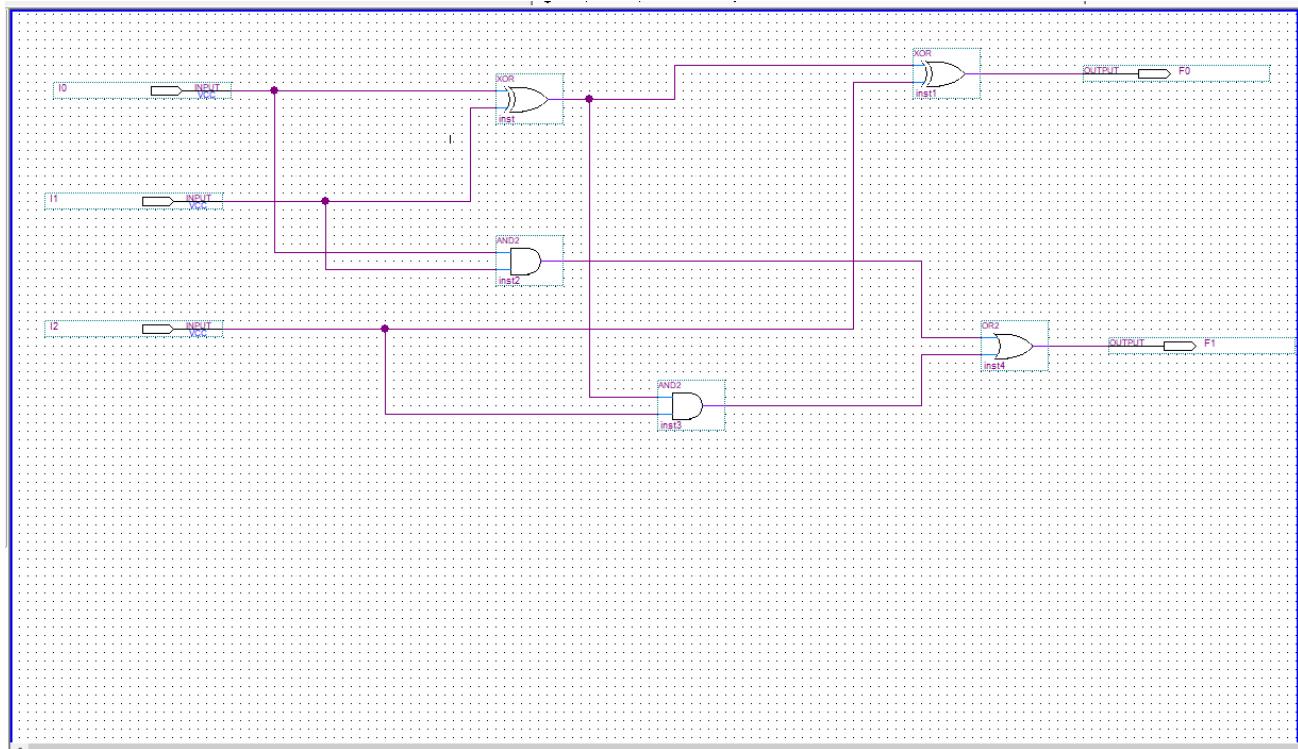


Τέλος το chip του Multiplexer είναι:





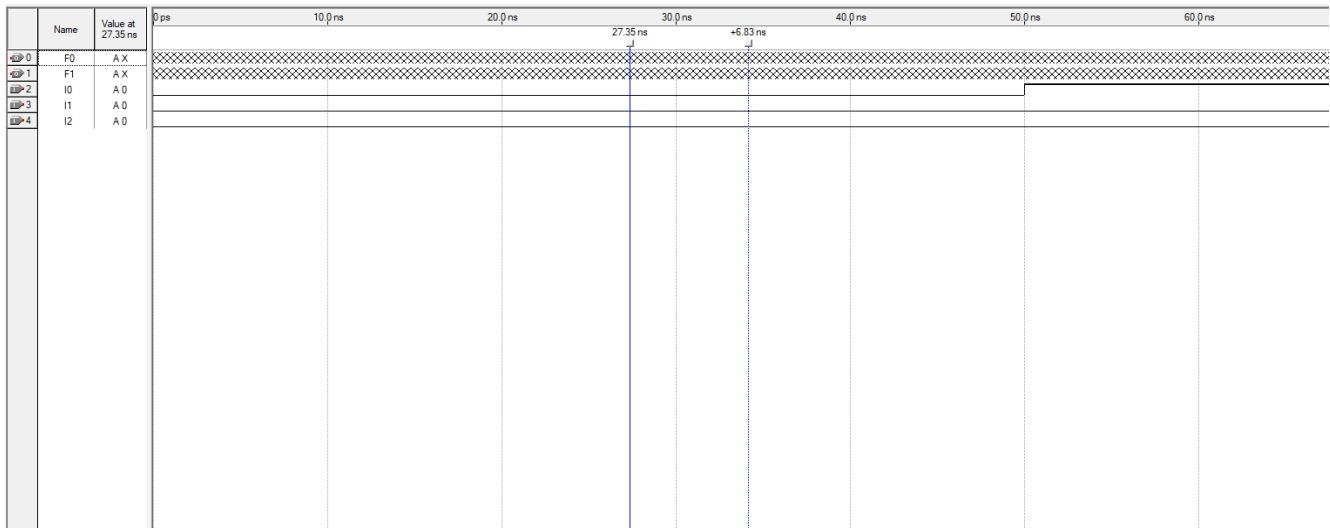
Μέρος 2^ο (γ): Full Adder



Σε αυτό το μέρος χρησιμοποιήσαμε 3 inputs, 2 XOR gates, 2 AND gates, 1 OR gate και 2 outputs.

The screenshot shows a software interface for pin assignment. The table lists the following information:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	F0	Output	PIN_AE22	7	B7_N0	3.3-V LVTTL (default)	
2	F1	Output	PIN_AF22	7	B7_N0	3.3-V LVTTL (default)	
3	IO	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
4	I1	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	

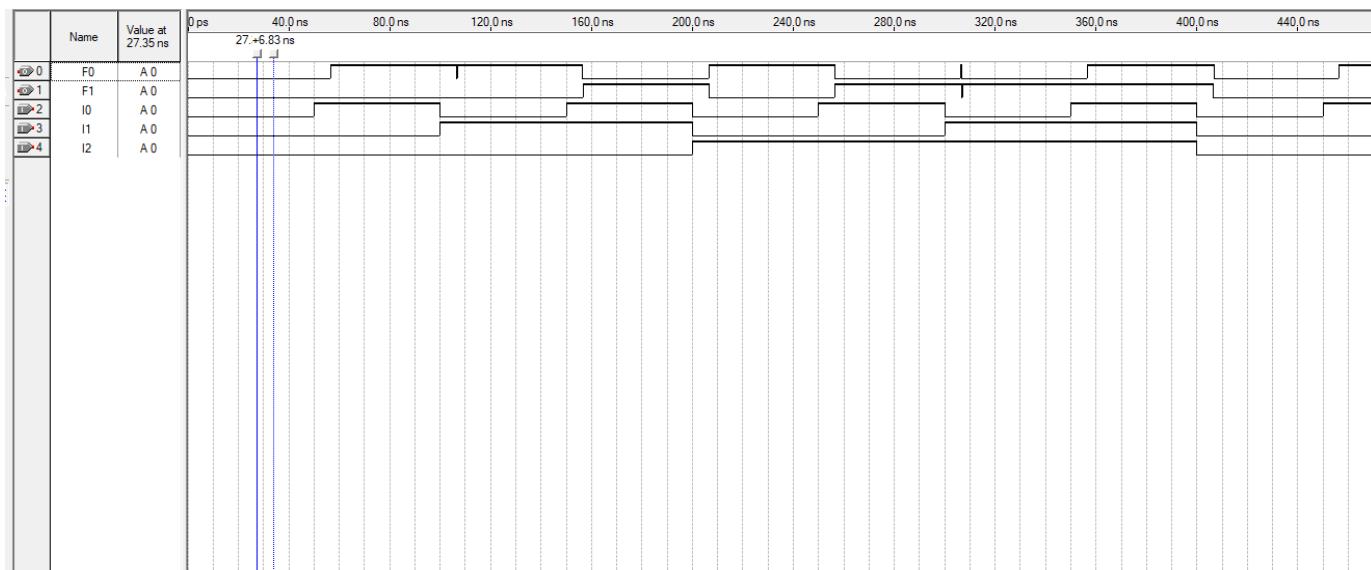


Θέτουμε τιμές στις εισόδους : $I0 = 50 \text{ ns}$

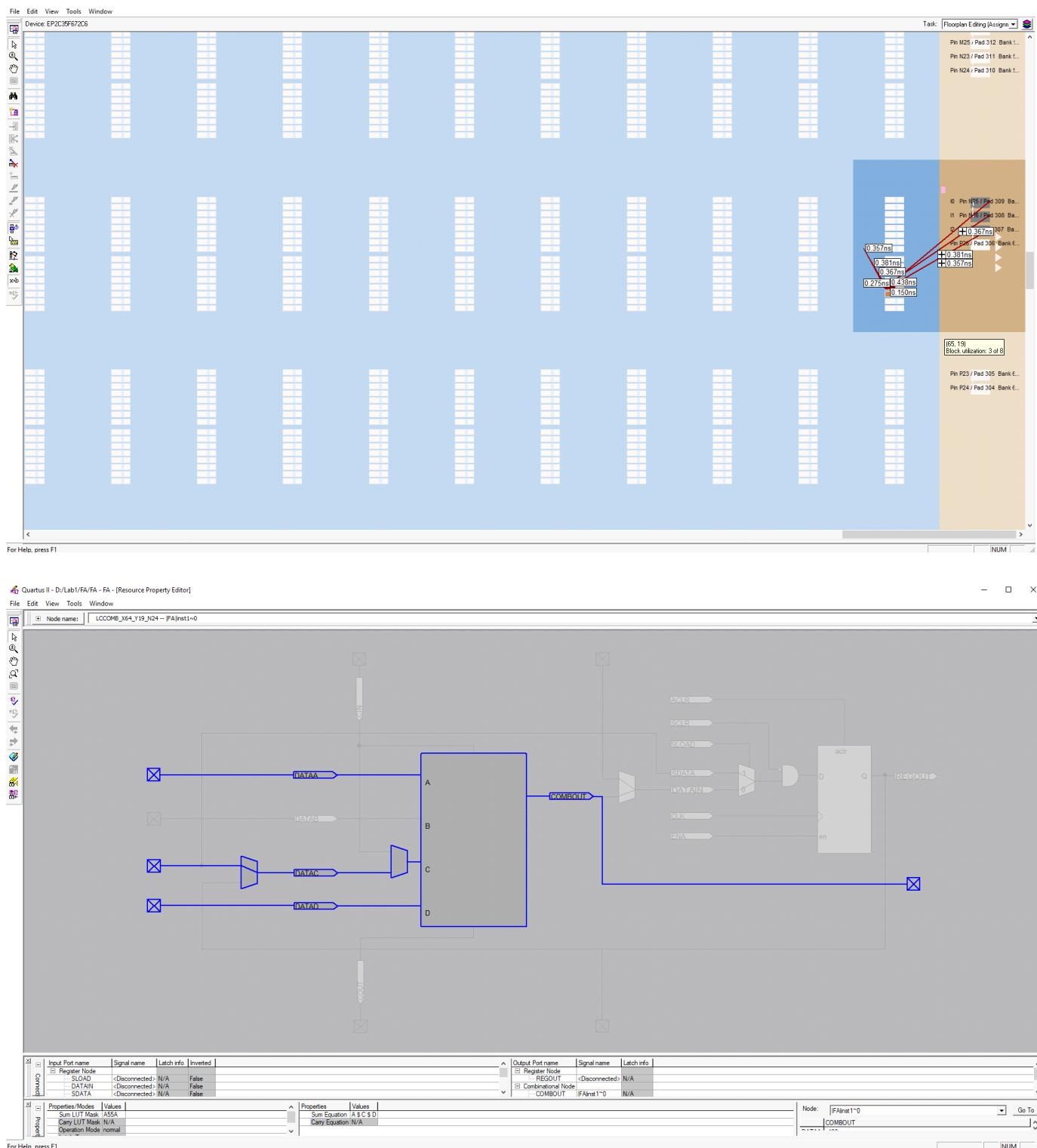
$I1 = 100 \text{ ns}$

$I2 = 200 \text{ ns}$

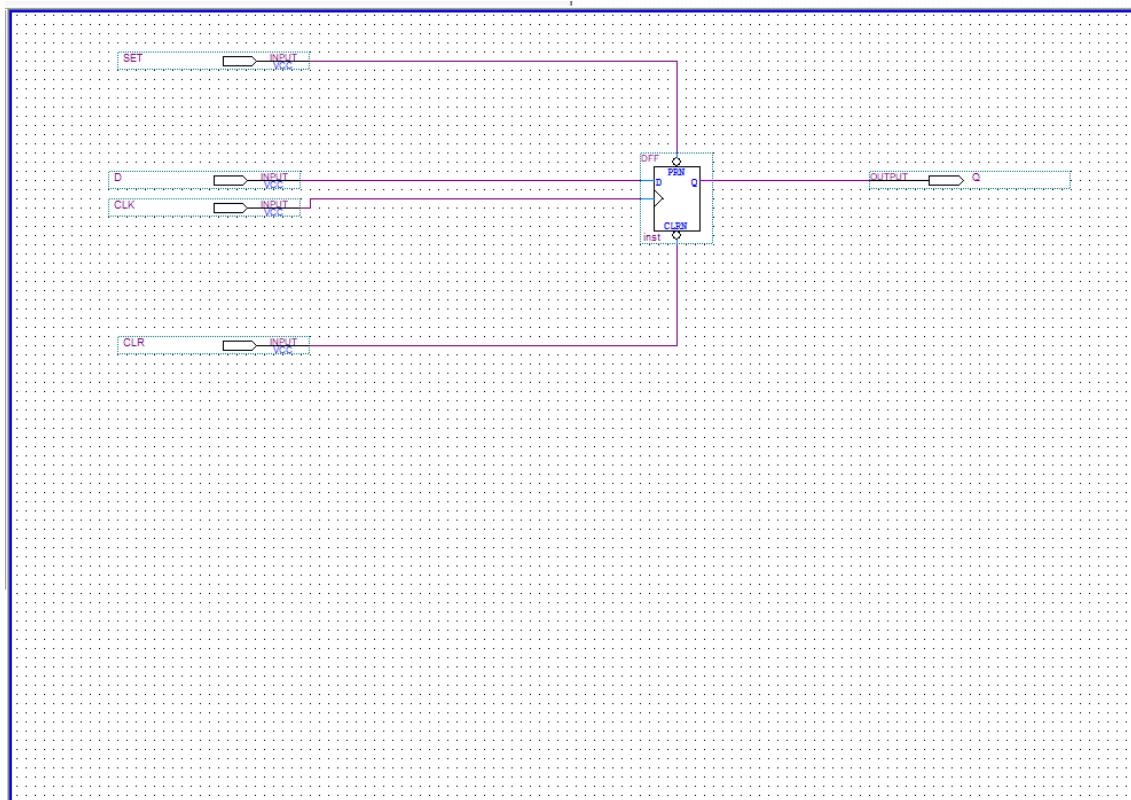
Έτσι έχουμε :



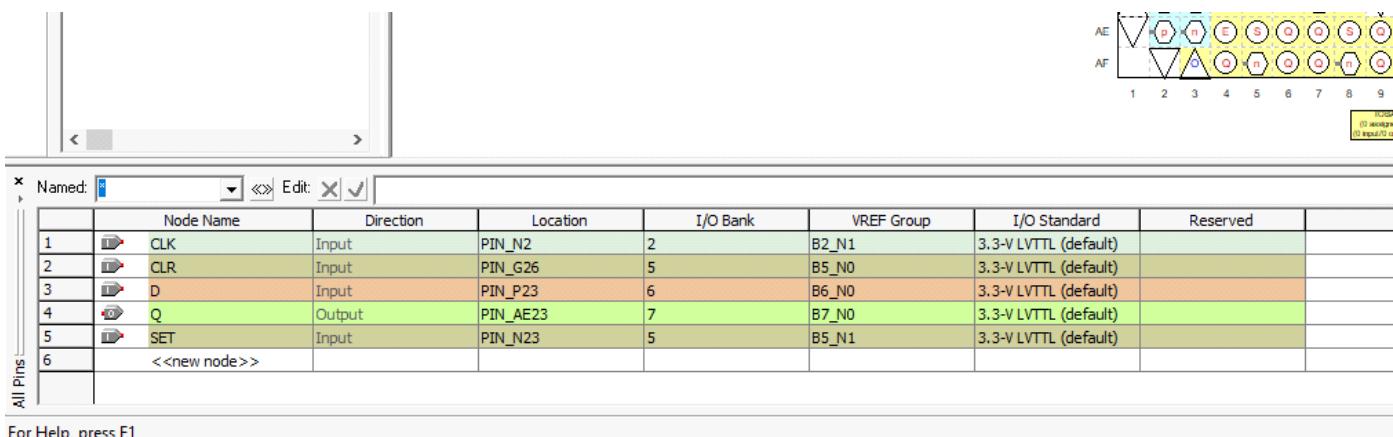
Τέλος το chip του FA είναι:

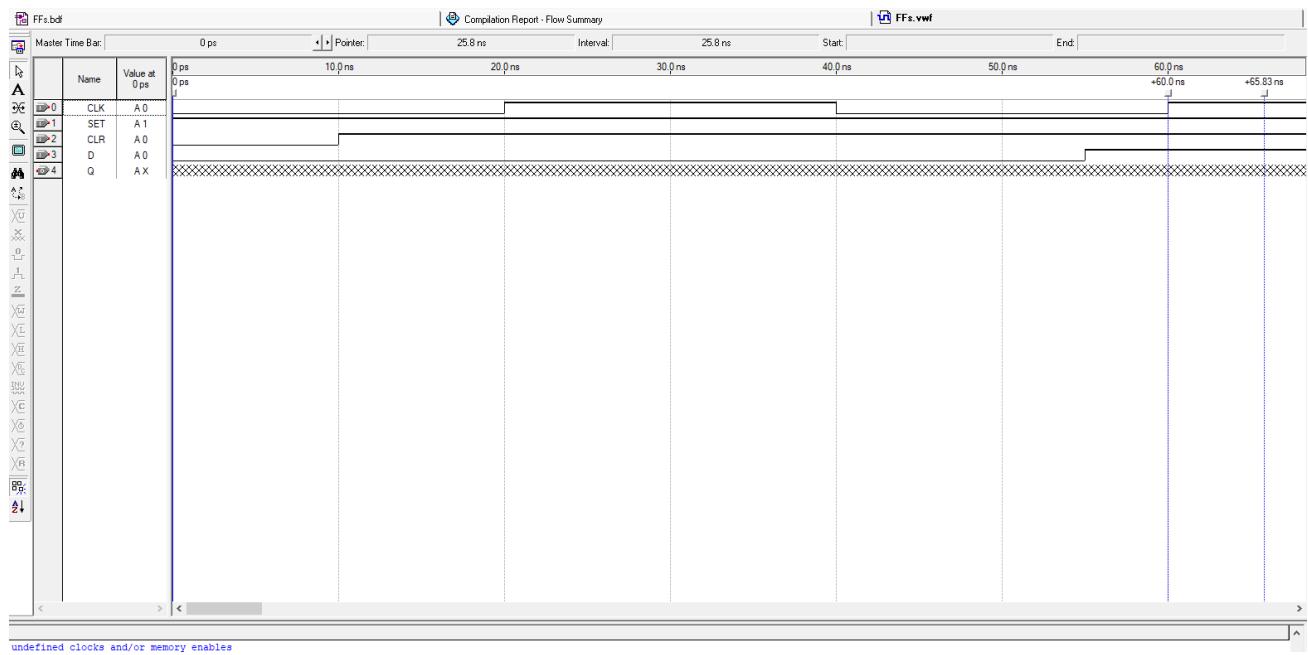


Μέρος 3^ο (α) : D-FF



Σε αυτό το μέρος χρησιμοποιήσαμε 4 inputs, 1 D-FF και 1 output.





Θέτουμε τιμές στις εισόδους : $CLK = 20 \text{ ns}$

$SET = 1$

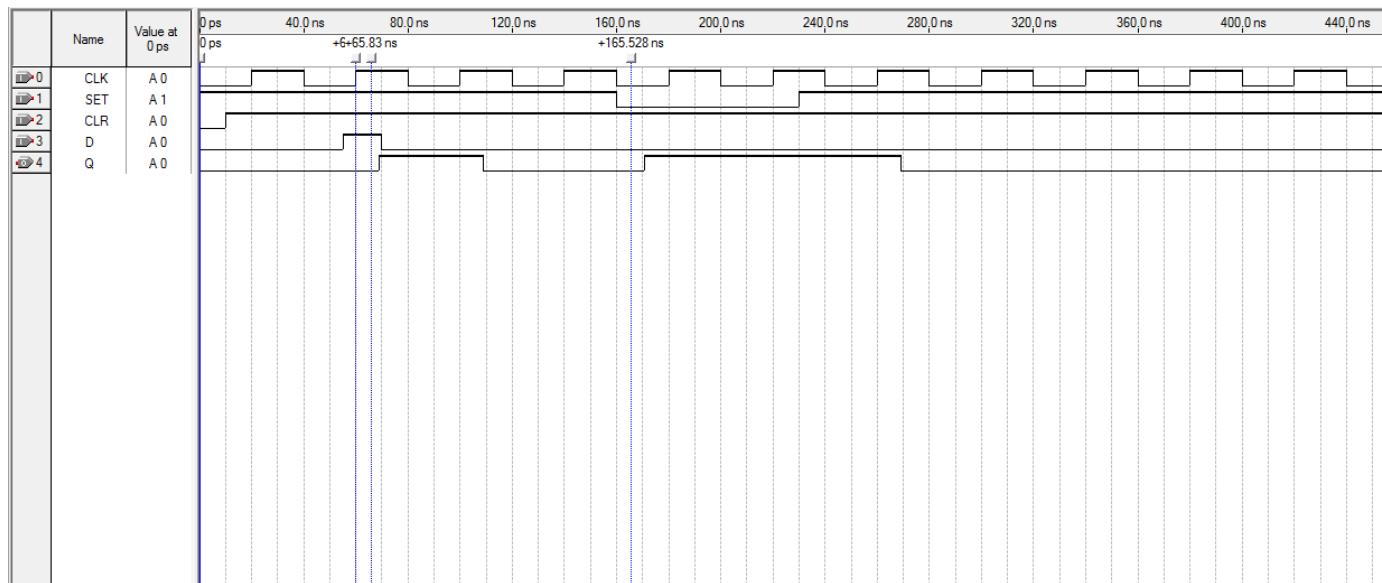
$CLR = 0\text{-}10 \text{ ns} ==> 0$

$CLR = 10\text{+} \text{ ns} ==> 1$

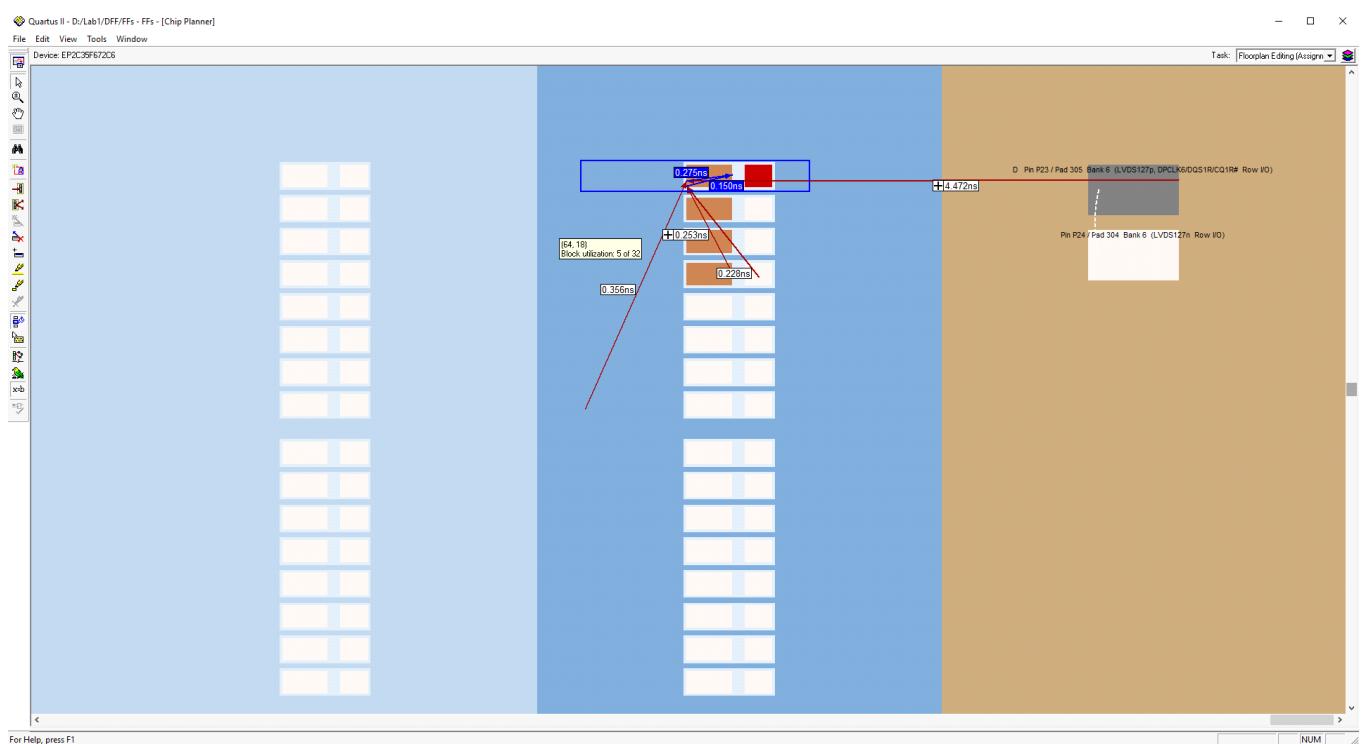
$D = 0\text{-}55 \text{ και } 70\text{+} \text{ ns} ==> 0$

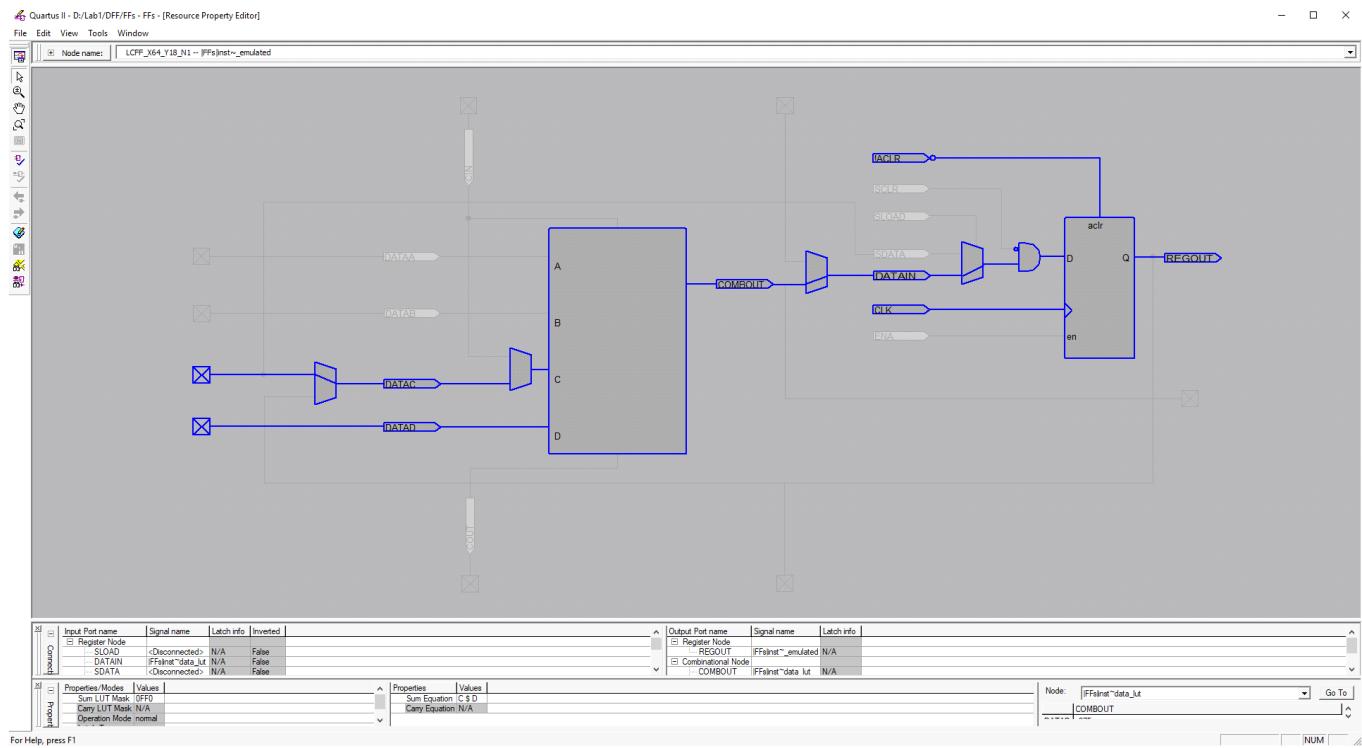
$D = 55\text{-}70 \text{ ns} ==> 1$

Έτσι έχουμε:

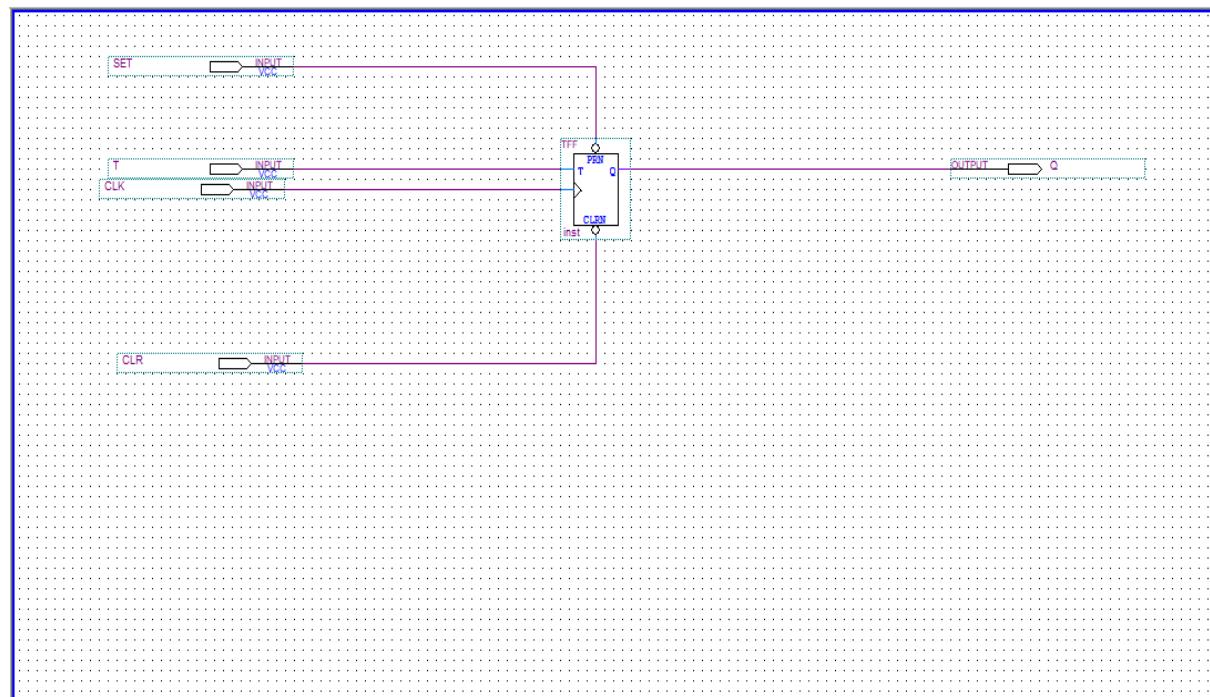


Τέλος το chip του D-FF είναι:

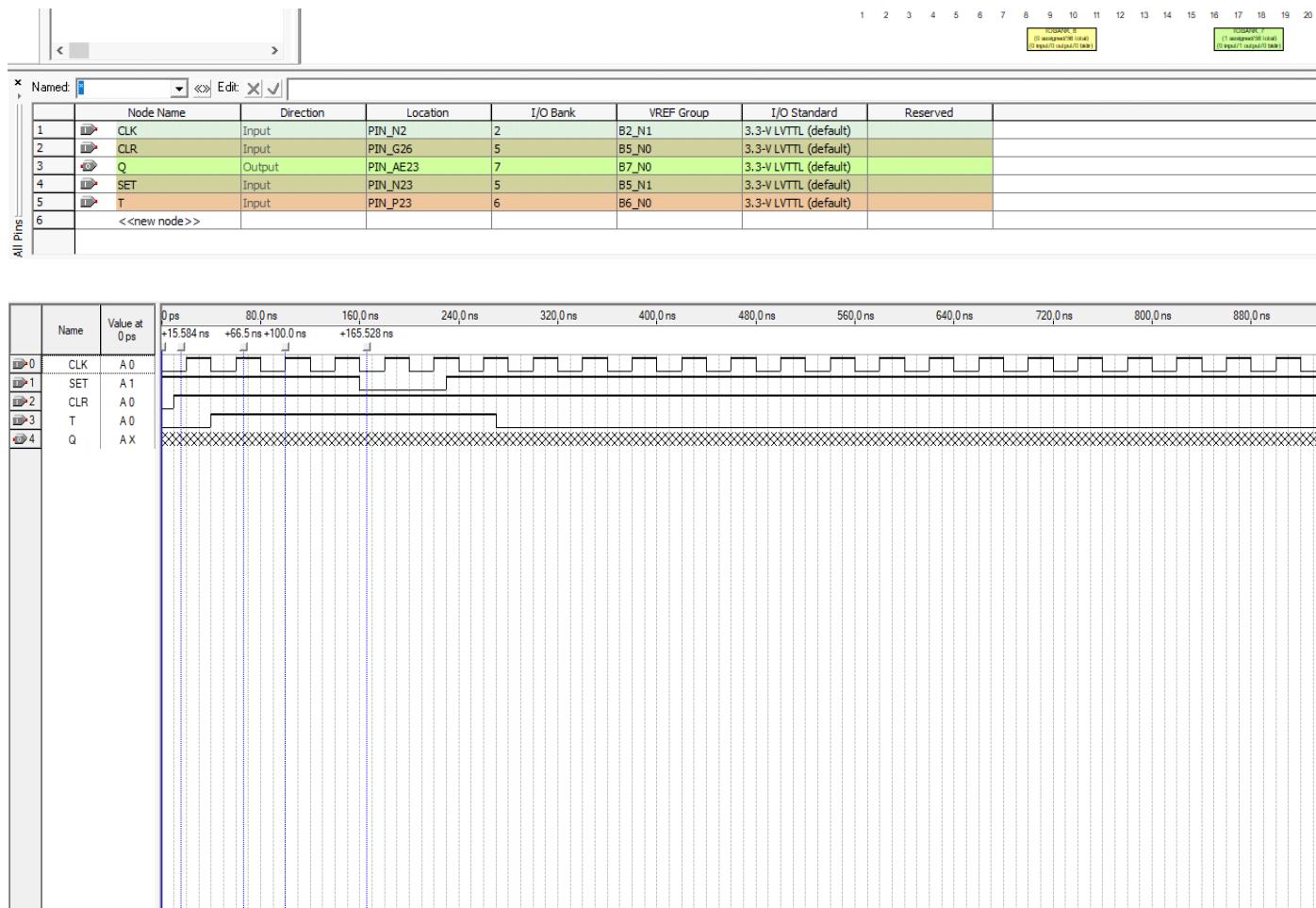




Μέρος 3^ο (β) : T-FF



Σε αυτό το μέρος χρησιμοποιήσαμε 4 inputs, 1 T-FF και 1 output.



Θέτουμε τιμές στις εισόδους : CLK = 20 ns

SET = 160-230 ns ==> 0

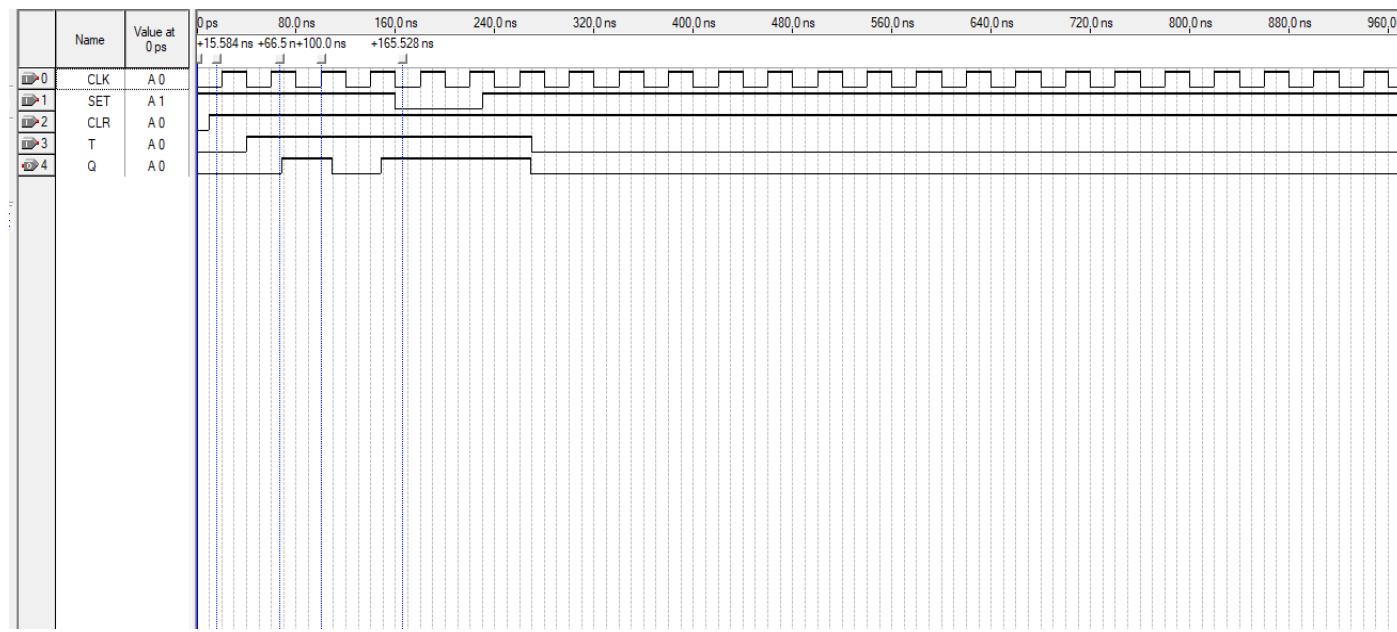
SET = 0-160 και 230+ ns ==> 1

CLR = 0-10 ns ==> 0

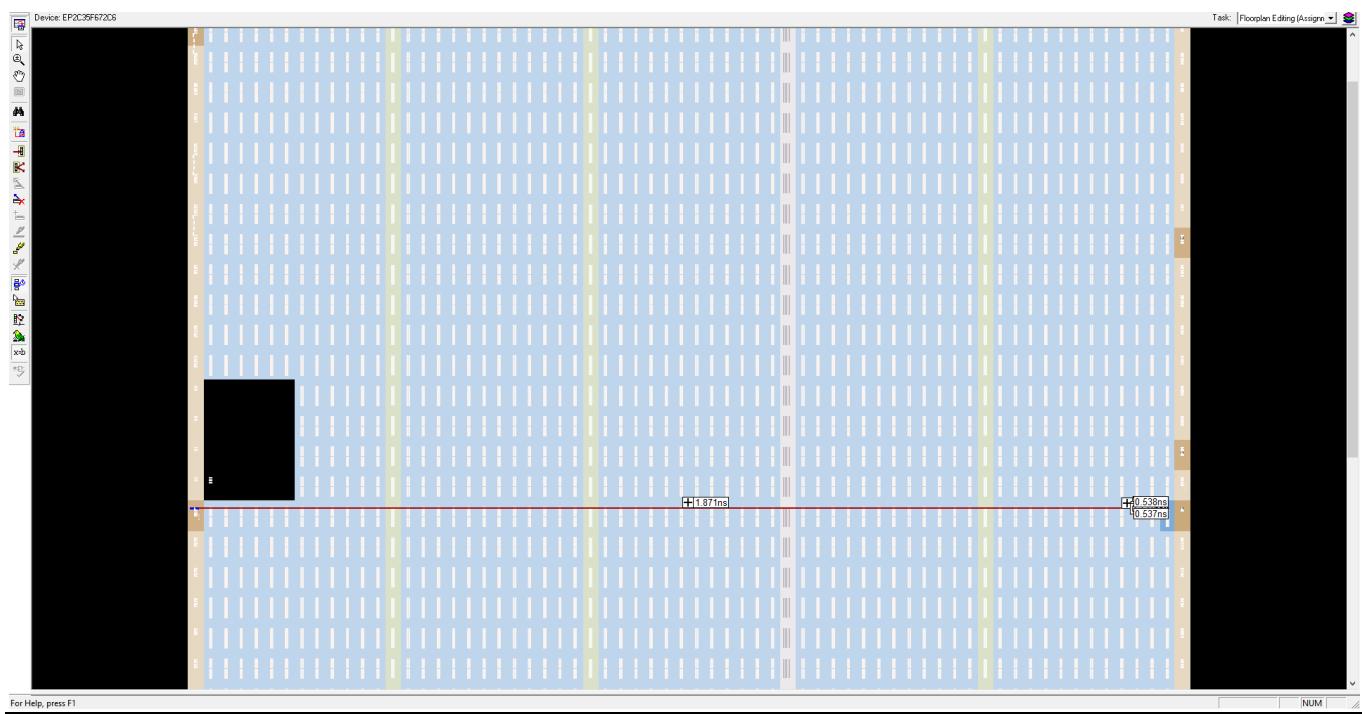
CLR = 10+ ns ==> 1

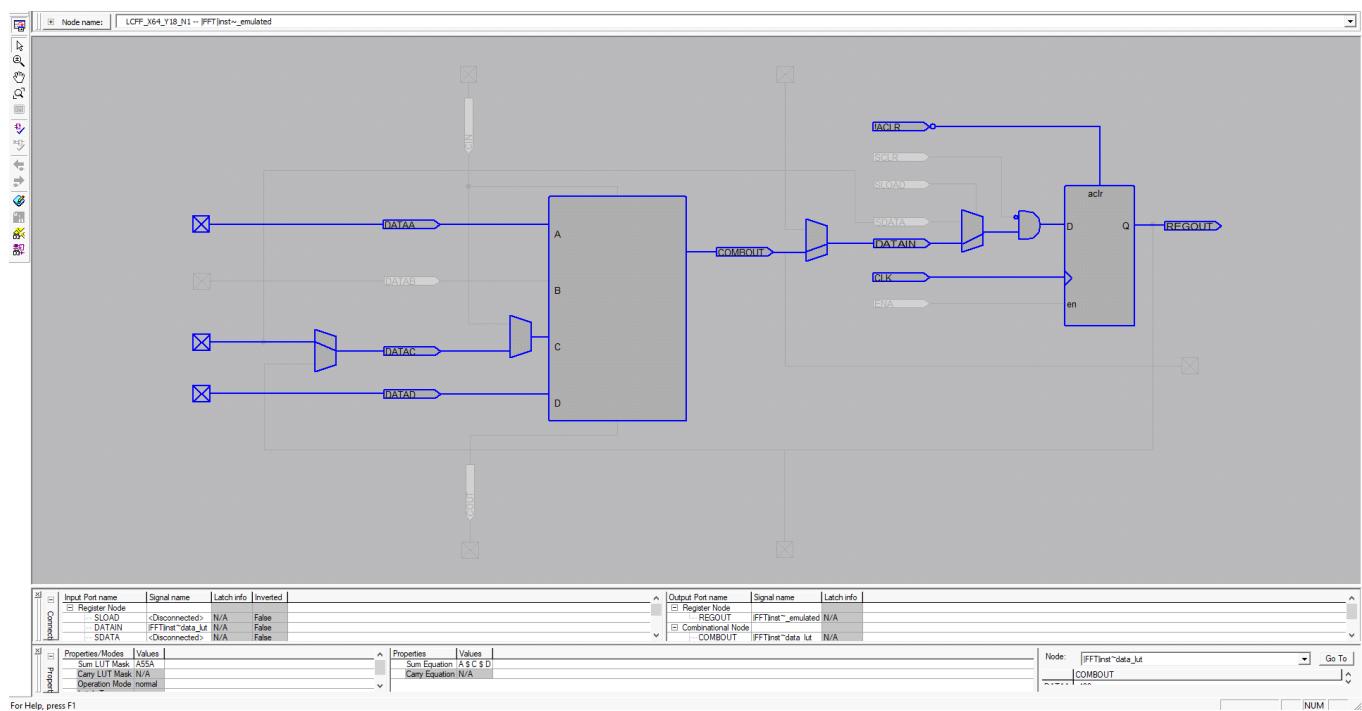
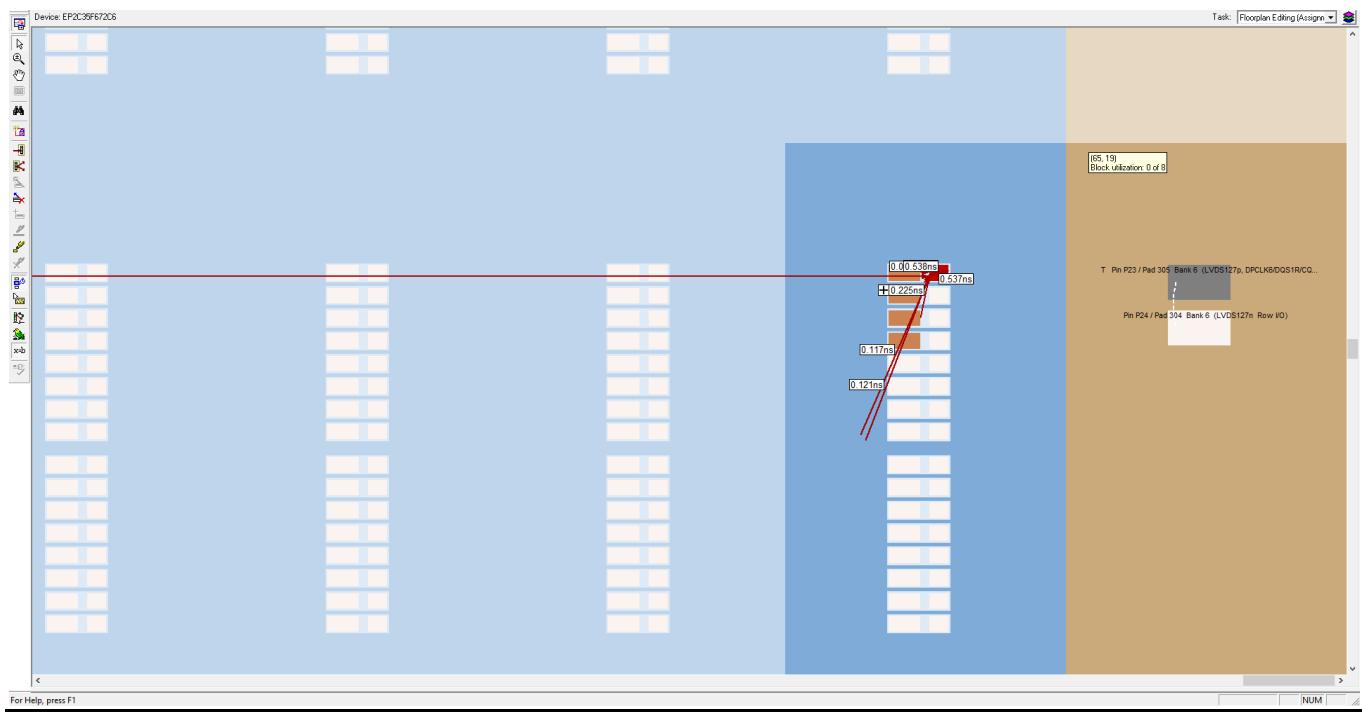
T = 0-55 και 250+ ns ==> 0

T = 55-250 ns ==>1

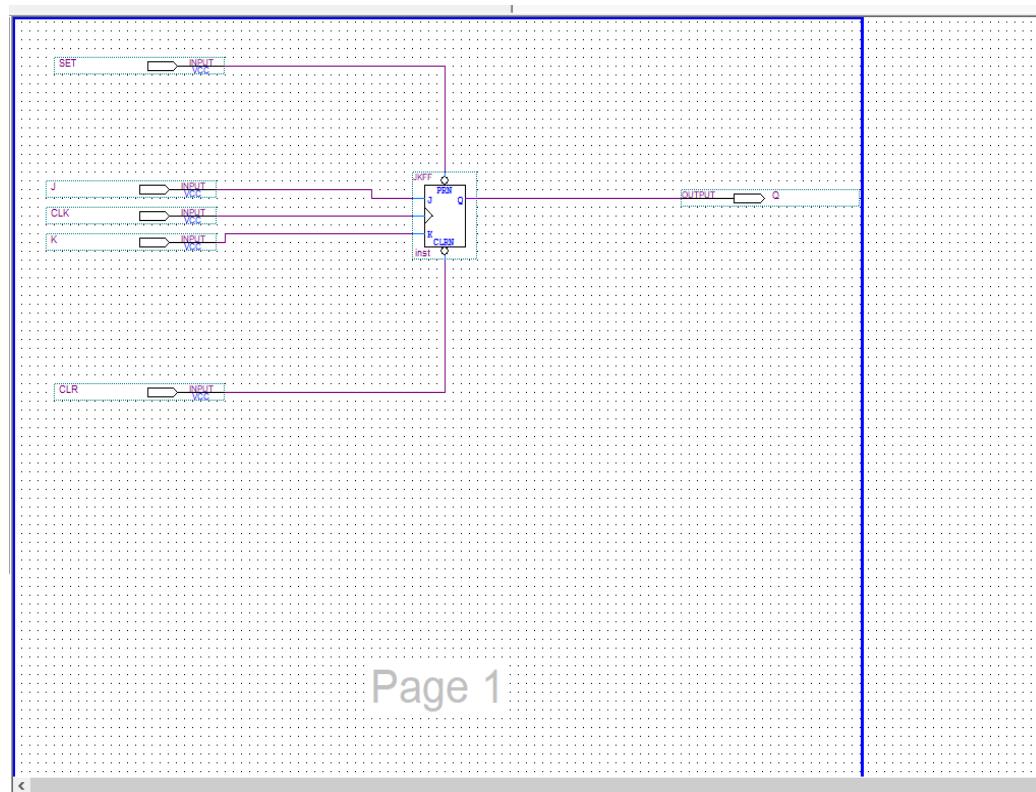


Τέλος το chip του T-FF είναι:





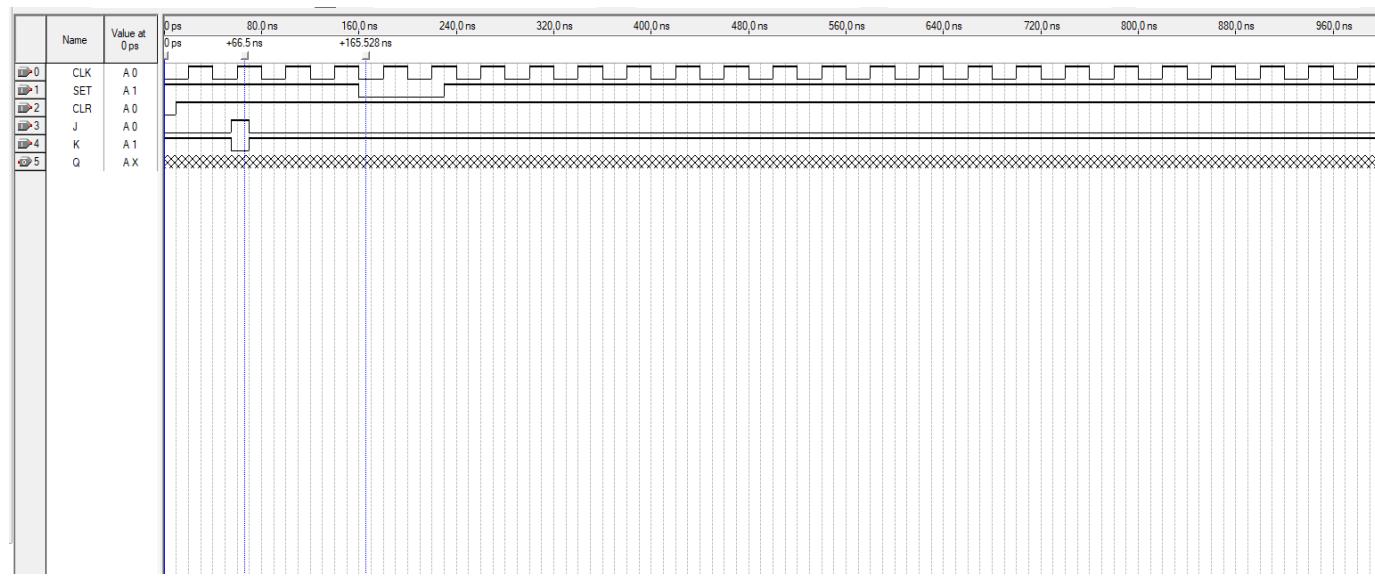
Μέρος 3^ο (γ) : JK-FF



Page 1

Σε αυτό το μέρος χρησιμοποιήσαμε 5 inputs, 1 JK-FF και 1 output.

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
1	CLK	Input	PIN_N2	2	B2_N1	3.3-V LVTTL (default)		
2	CLR	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)		
3	J	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)		
4	K	Input	PIN_P23	6	B6_N0	3.3-V LVTTL (default)		
5	Q	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
6	SET	Input	PIN_W26	6	B6_N1	3.3-V LVTTL (default)		
7	<<new node>>							



Θέτουμε τιμές στις εισόδους : $CLK = 20 \text{ ns}$

$$SET = 160-230 \text{ ns} \Rightarrow 0$$

$$SET = 0-160 \text{ και } 230+ \text{ ns} \Rightarrow 1$$

$$CLR = 0-10 \text{ ns} \Rightarrow 0$$

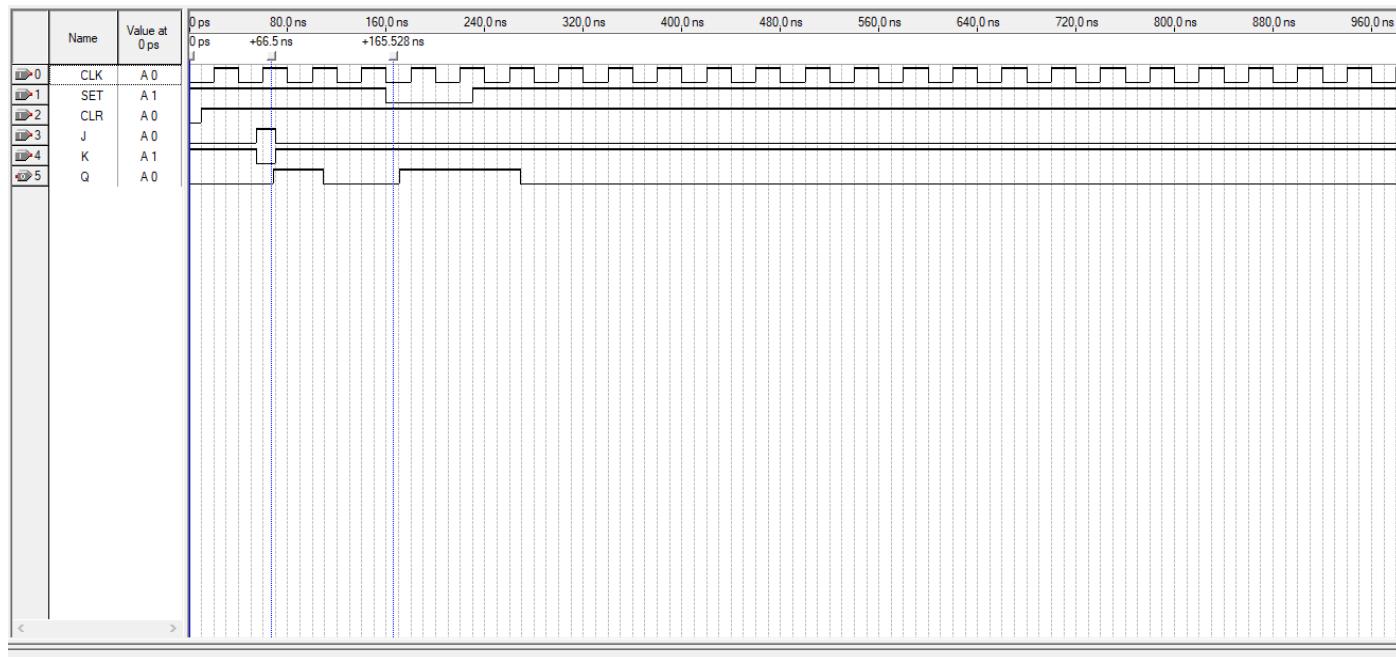
$$CLR = 10+ \text{ ns} \Rightarrow 1$$

$$J = 0-55 \text{ και } 70+ \text{ ns} \Rightarrow 0$$

$$J = 55-70 \text{ ns} \Rightarrow 1$$

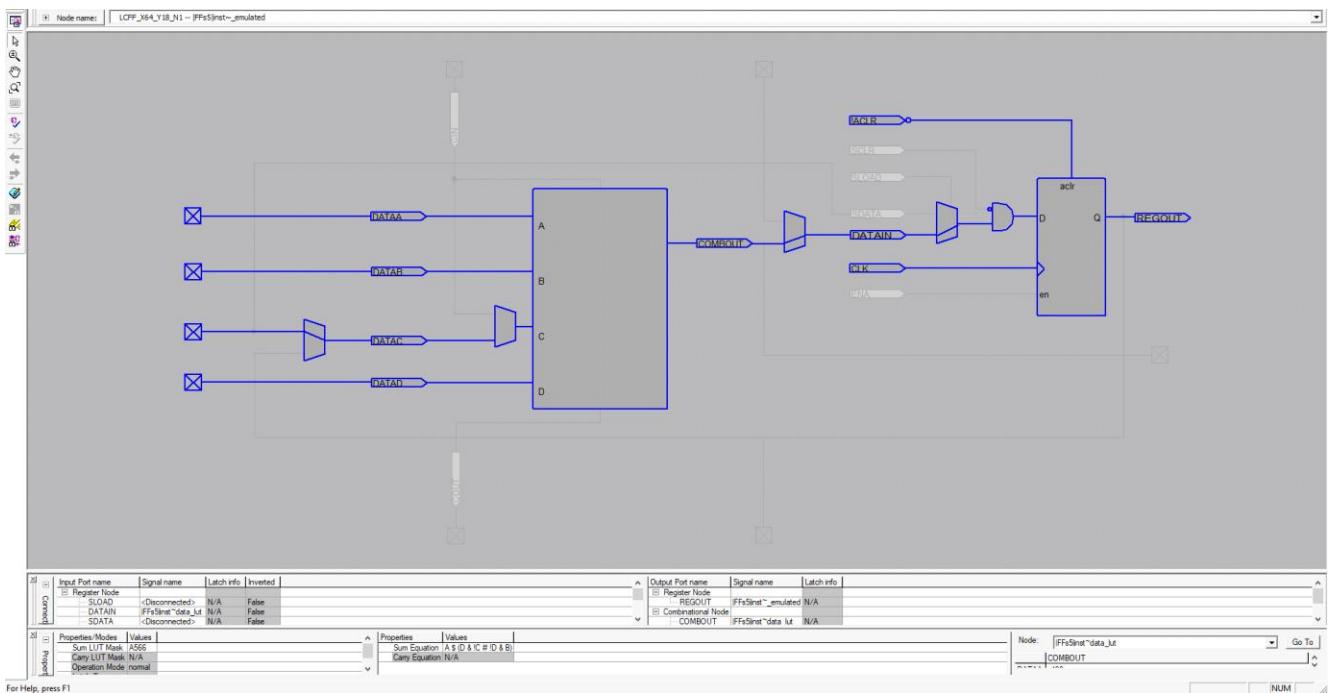
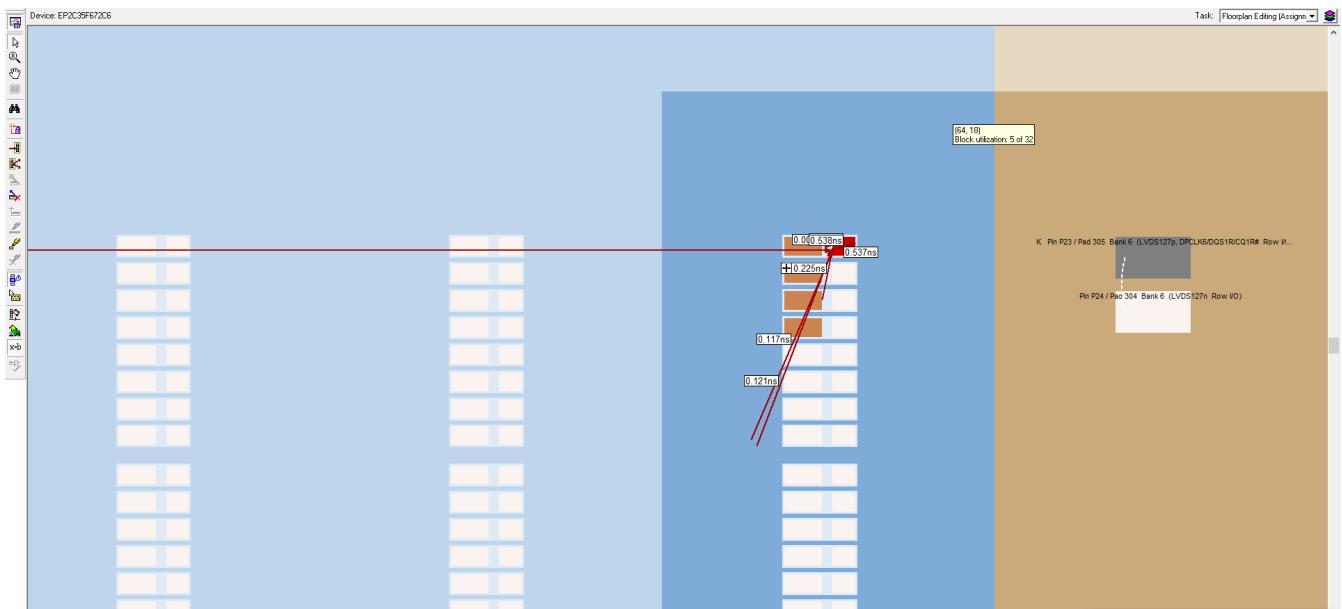
$$K = 0-55 \text{ και } 70+ \text{ ns} \Rightarrow 1$$

$$K = 55-70 \text{ ns} \Rightarrow 0$$



Τέλος το chip του JK-FF είναι:





Μέρος 2(δ):

Σε αυτό το κομμάτι είχαμε να σχεδιάσουμε τον καταχωρητή της εκφώνησης, χρησιμοποιώντας το κύκλωμα το πολυπλέκτη το οποίο σχεδιάσαμε προηγουμένως μαζί με D flip-flop. Σύμφωνα με τον πίνακα λειτουργίας έχουμε τα εξής δεδομένα:

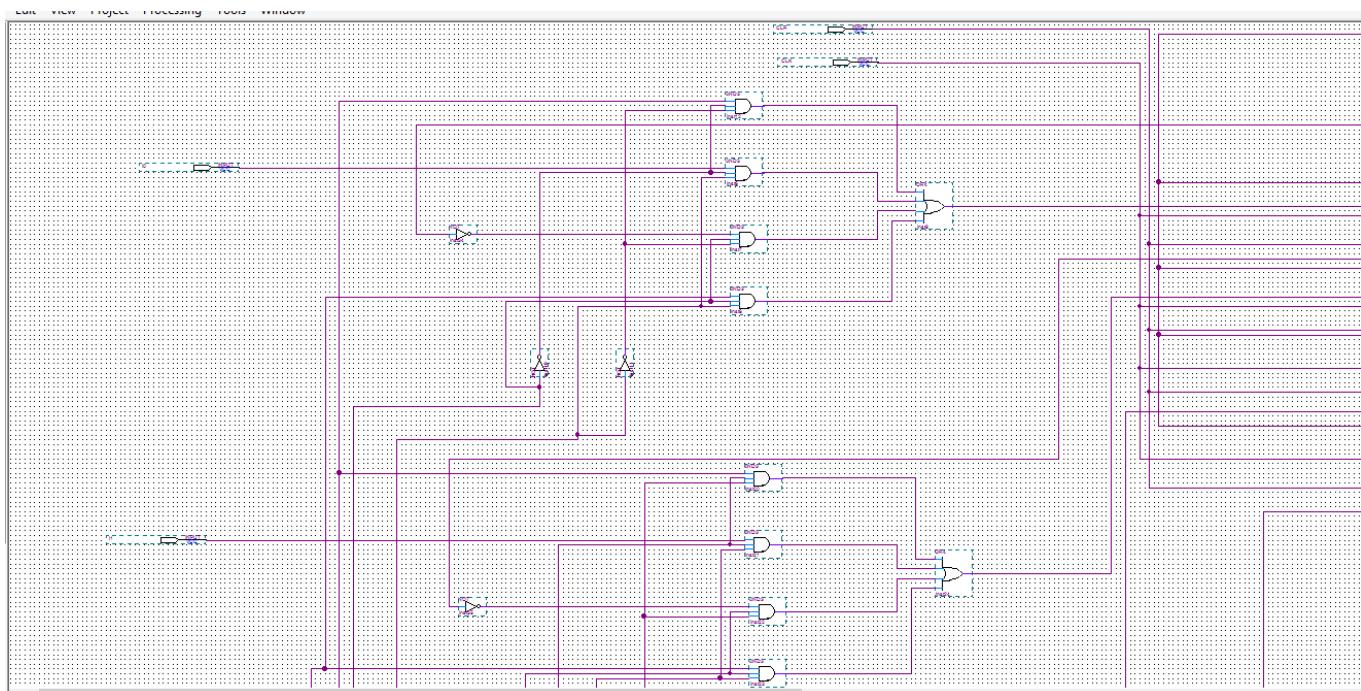
- $(S1, S0) = (0,0)$: Έχουμε πως ο καταχωρητής μηδενίζεται συγχρόνως.

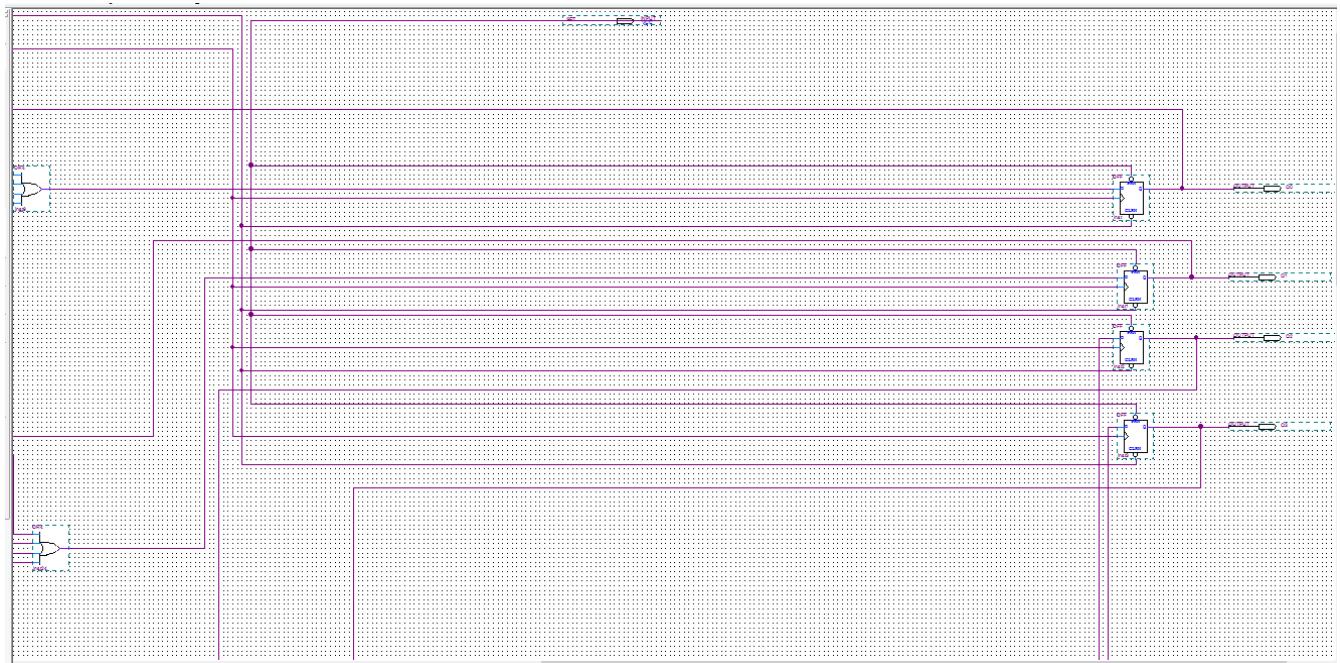
- $(S1, S0) = (0,1)$: Ο καταχωρητής σε αυτή την περίπτωση εκτελεί παράλληλη φόρτωση των δεδομένων $I[3..0]$.

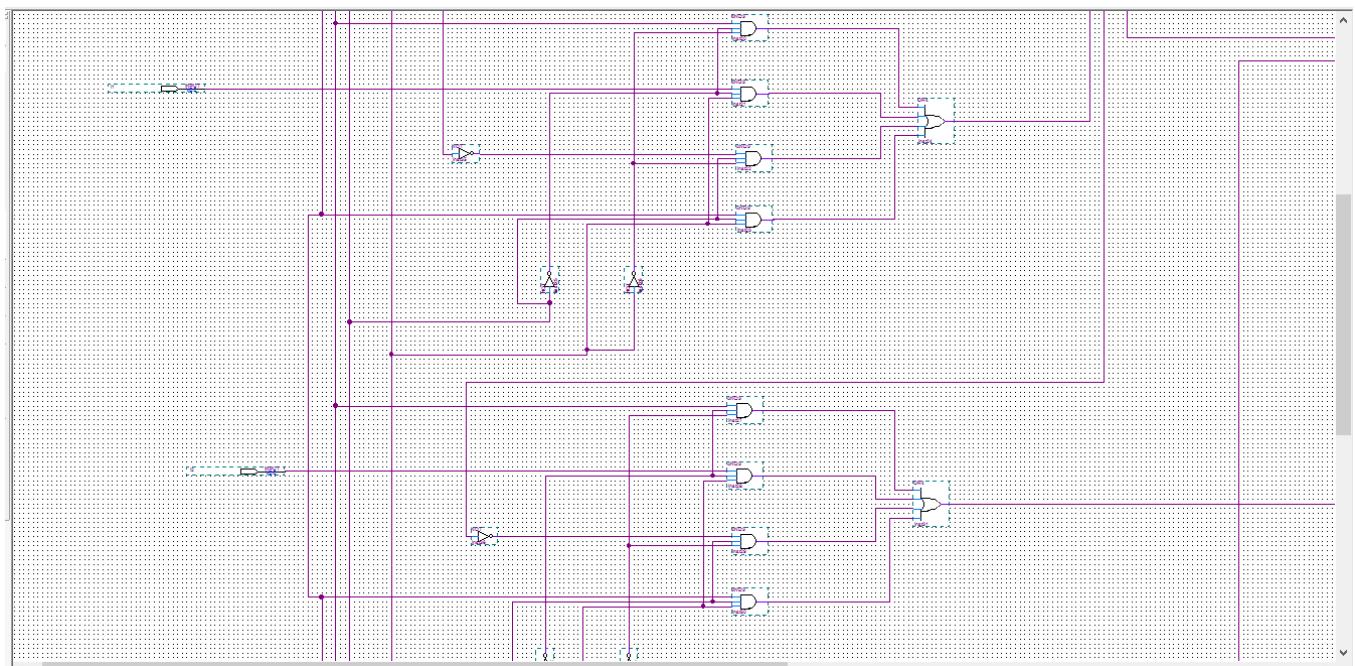
- $(S1, S0) = (1,0)$: Ο καταχωρητής σε αυτή την περίπτωση φορτώνει την συμπληρωματική έξοδο του D flip-flop στην είσοδο του πολυπλέκτη.

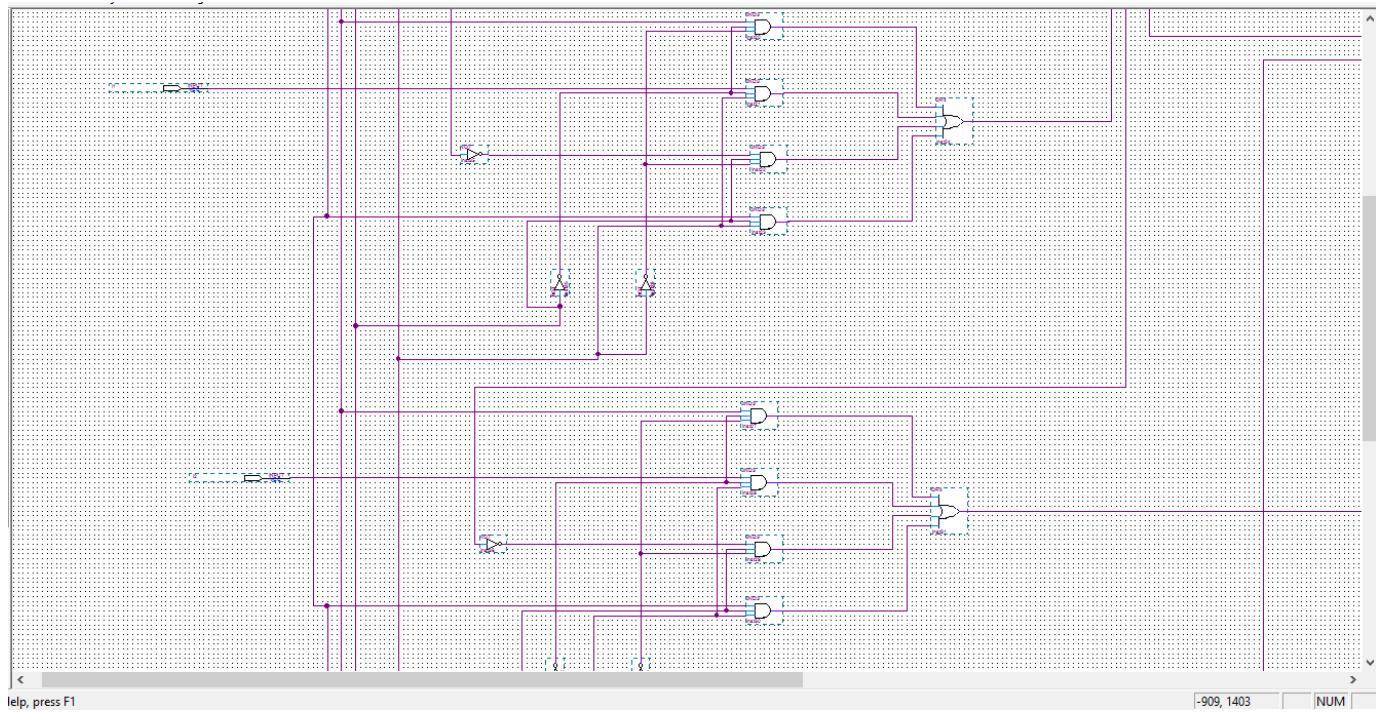
- $(S1, S0) = (1,1)$: Ο καταχωρητής σε αυτή την περίπτωση παίρνει την μονάδα συγχρόνως.

Άρα, το κύκλωμα θα είναι το εξής:

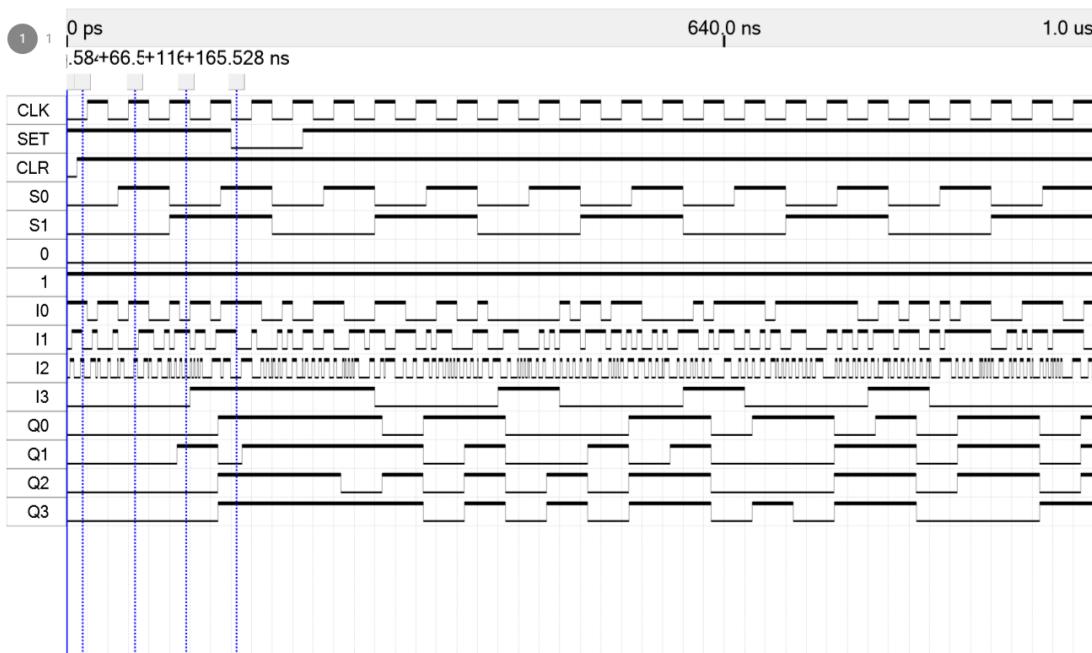








To waveform του καταχωρητή είναι το εξής:



Όπου οι είσοδοι S1 και S2 έχουν τιμή αλλαγής στα 50ns και 100ns ,αντίστοιχα. Το ρολόι έχει περίοδο 40ns και το σήμα SET μηδενίζεται μόνο στο διάστημα [160ns,320ns]. Επίσης θέσαμε τυχαίες τιμές στις εισόδους I3,I2,I1 και I0,αντίστοιχα και το σήμα CLR είναι 0 μόνο για τα πρώτα 10ns.

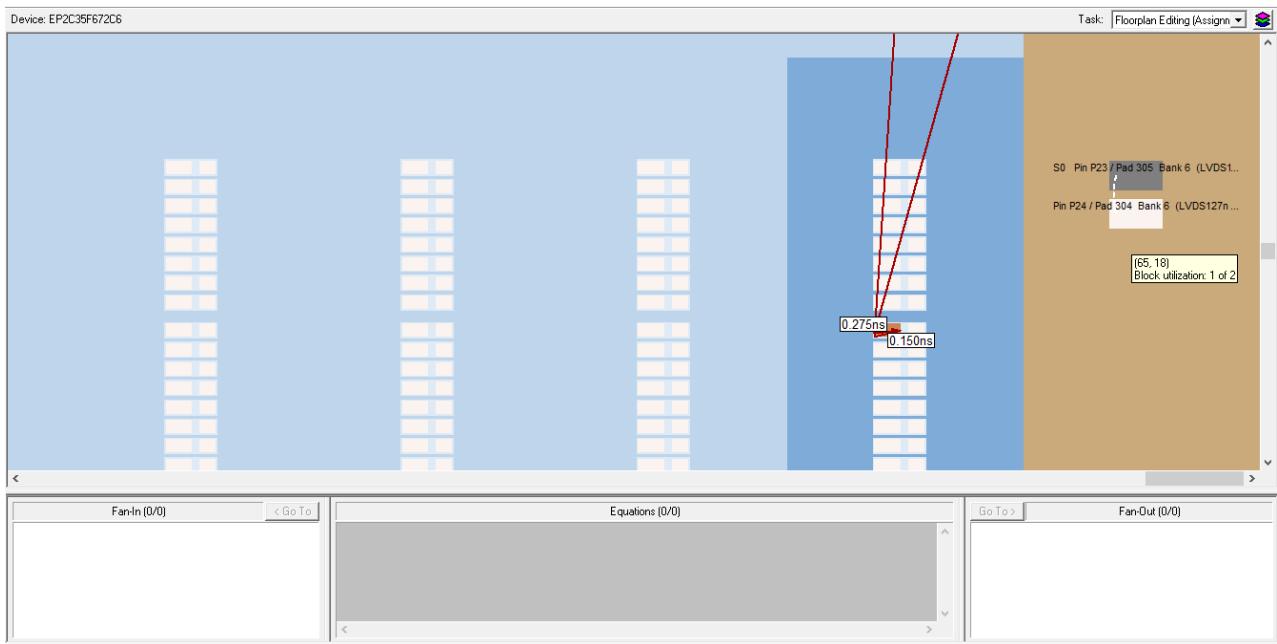
Ta pins τα οποία χρησιμοποιήσαμε είναι τα εξής:

Named:

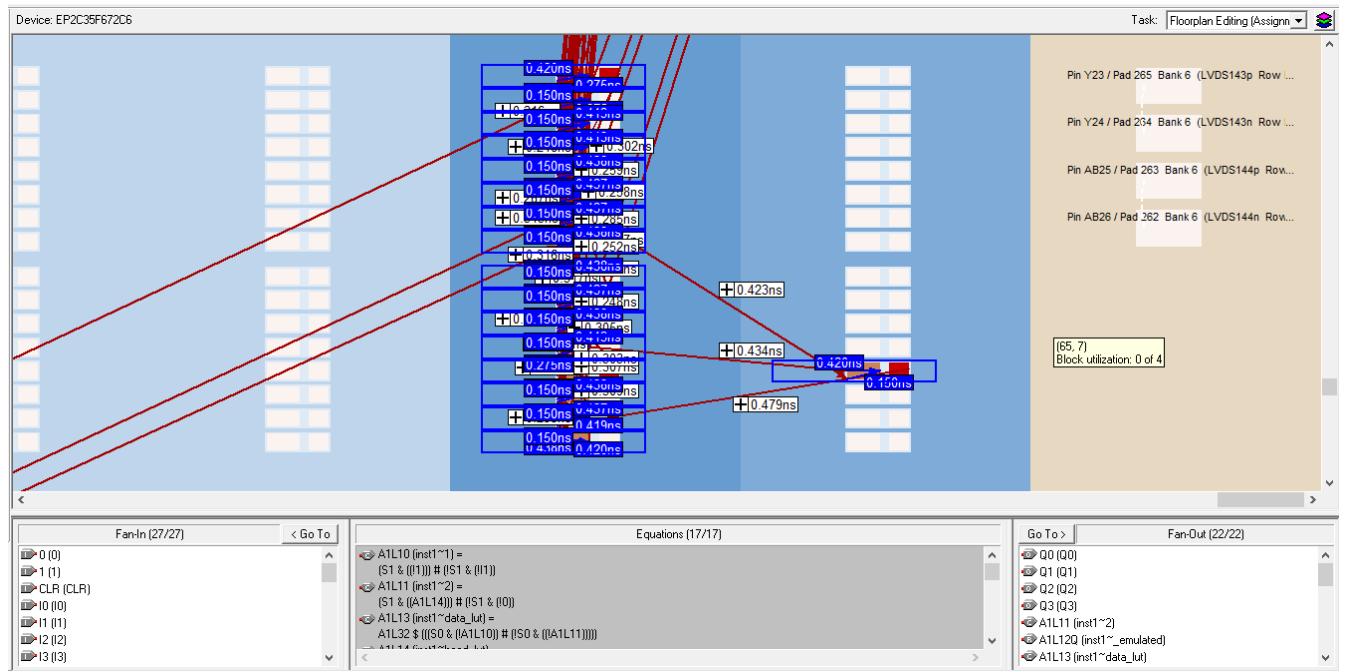
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	0	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
2	1	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
3	CLK	Input	PIN_N2	2	B2_N1	3.3-V LVTTL (default)	
4	CLR	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)	
5	I0	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
6	I1	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
7	I2	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)	
8	I3	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)	
9	Q0	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
10	Q1	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
11	Q2	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)	
12	Q3	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)	
13	S0	Input	PIN_P23	6	B6_N0	3.3-V LVTTL (default)	
14	S1	Input	PIN_W26	6	B6_N1	3.3-V LVTTL (default)	
15	SET	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)	
16	<<new node>>						

All Pins

Και παρακάτω φαίνεται το chip plan:

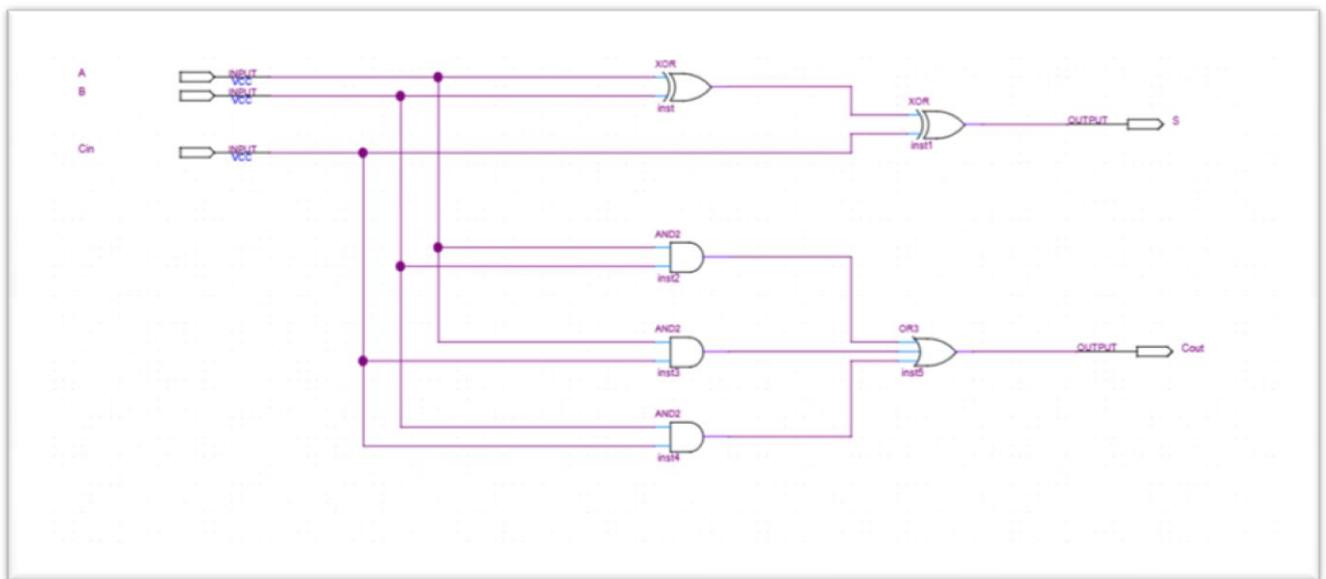


Και

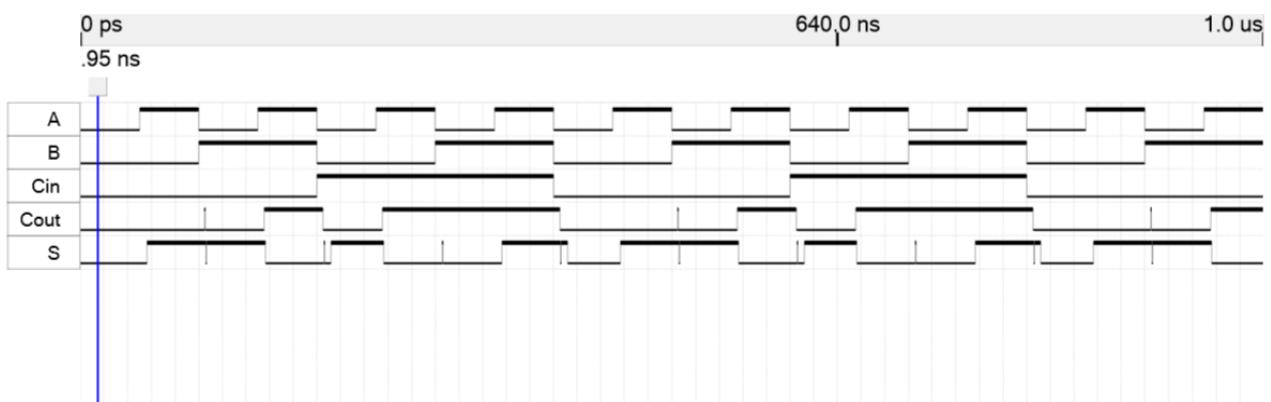


Εργαστηριακή Ασκηση 2: Ιεραρχική Σχεδίαση και προσγεδιασμένοι πυρήνες

Το ζητούμενο κύκλωμα είναι ο συσσωρευτής, ωστόσο για να τον σχεδιάσουμε, πρέπει πρώτα να τον αναλύσουμε στα επιμέρους στοιχεία. Αρχικά, σχεδιάσαμε τον πλήρη αθροιστή, ο οποίος θα αθροίσει τα ψηφία με την χαμηλότερη σημαντικότητα. Το σχηματικό του πλήρη αθροιστή είναι το εξής:

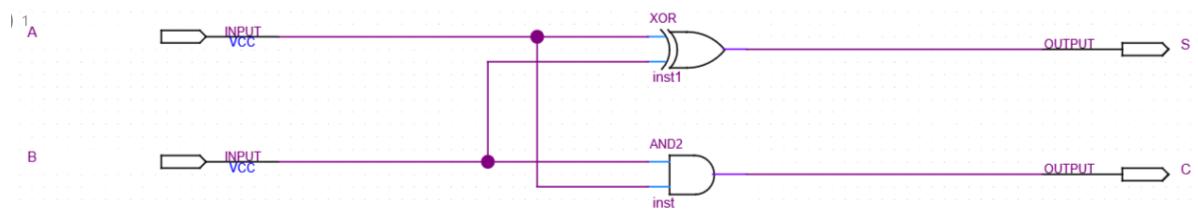


επίσης παρακάτω έχουμε και διάγραμμα κυματομορφών:

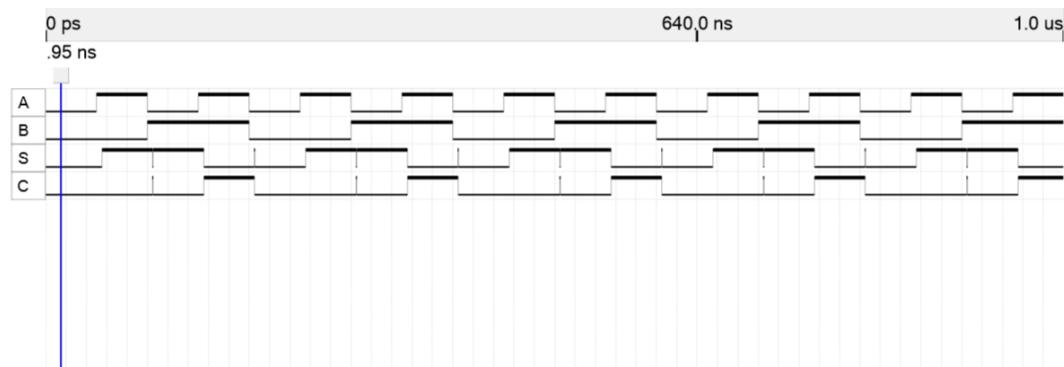


Για τις εισόδους του αθροιστή κατά την εξομοίωση, τοποθετήσαμε στην είσοδο A την τιμή 1, όπου την μεταβάλλαμε με περίοδο 50ns. Αντίστοιχα την είσοδο B την μεταβάλλαμε κατά 100ns και την είσοδο Cin κατά 200ns.

Στην συνέχεια, τα ψηφία με την υψηλότερη σημαντικότητα θα μπουν ως είσοδο στους ημιαθροιστές και θα αθροίζονται με το κρατούμενο εξόδου της προηγούμενης βαθμίδας. Το κύκλωμα του ημιαθροιστή είναι το εξής:

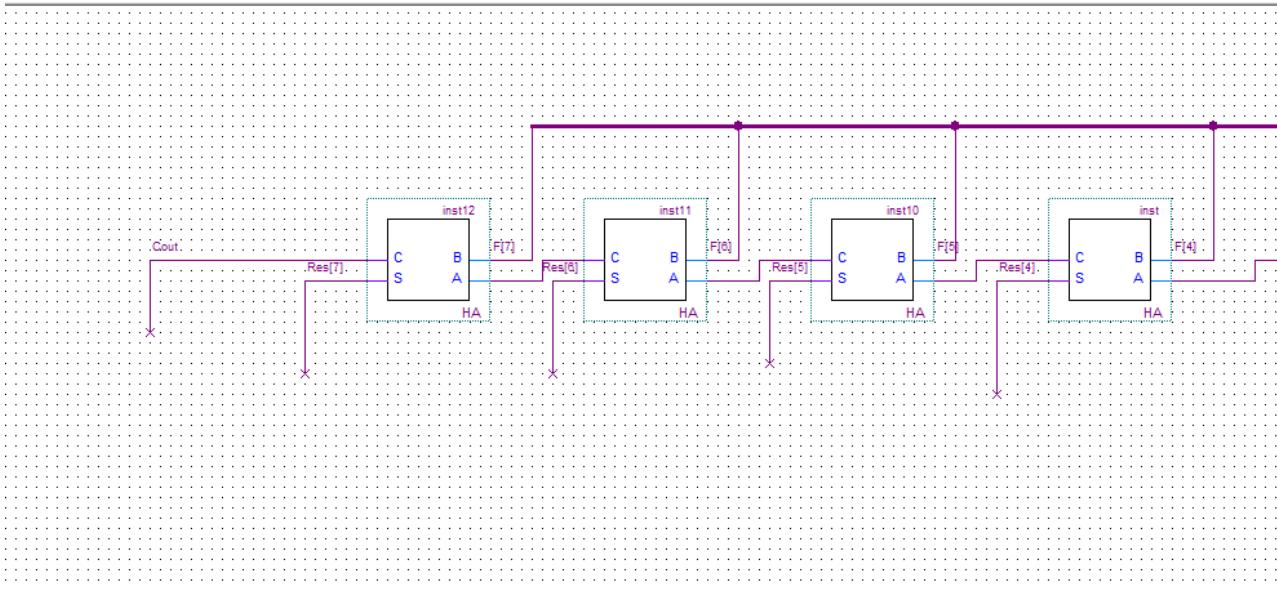
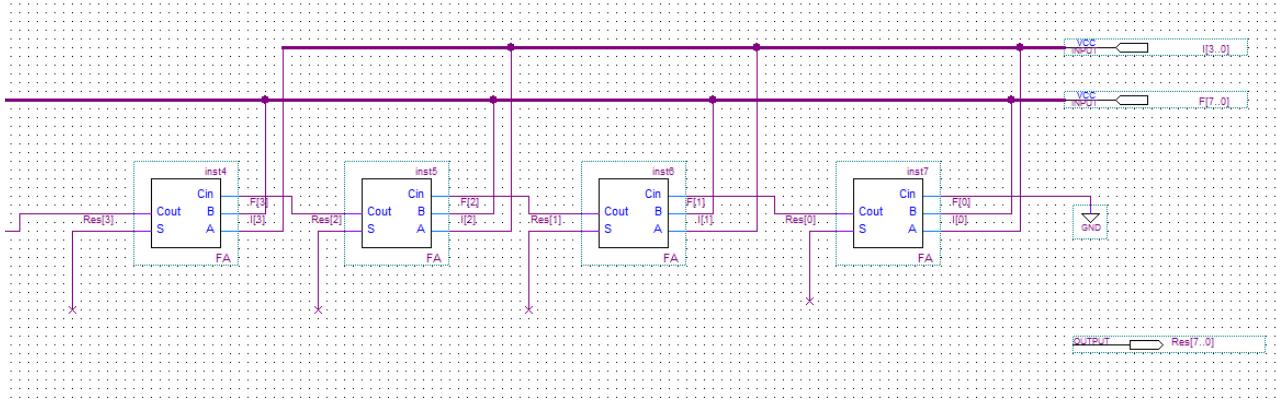


Και το διάγραμμα των κυματομορφών είναι το εξής:

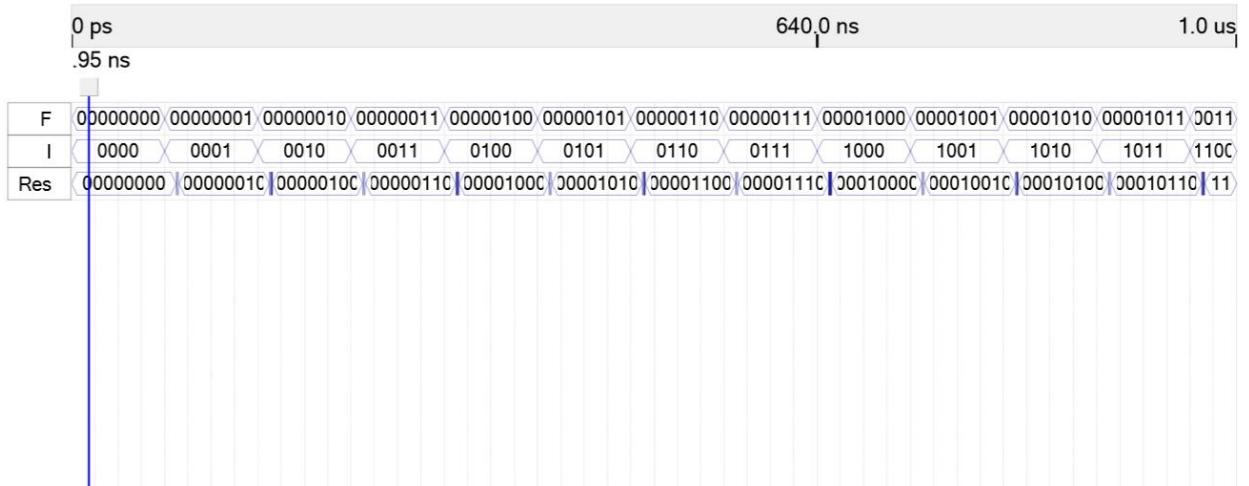


Για τις εισόδους του ήμιαθροιστή κατά την εξομοίωση, τοποθετήσαμε στην είσοδο A την τιμή 1, όπου την μεταβάλλαμε με περίοδο 50ns και την είσοδο B κατά 100ns.

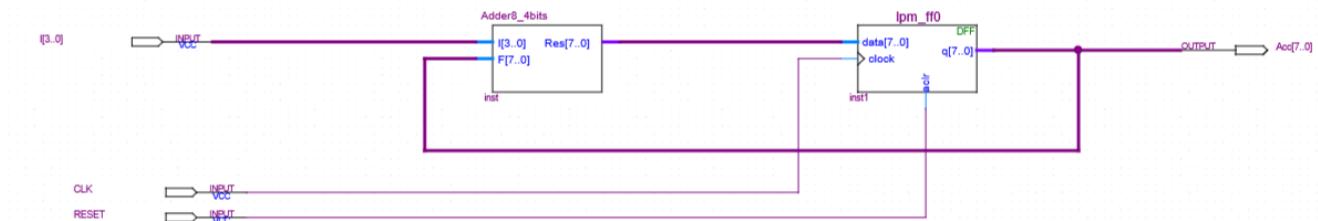
Τώρα που έχουμε στην κατοχή μας τα δύο blocks, μπορούμε να κατασκευάσουμε τον αθροιστή 8 δυαδικών ψηφίων. Ο αθροιστής των 8 bit παίρνει ως είσοδο 4 bits $I[3..0]$, τα οποία προστίθονται με τα 4 χαμηλής σημαντικότητας bits της εισόδου $F[7..0]$. Στην συνέχεια τα υπόλοιπα 4 bits της $F[7..0]$ θα εισαχθούν στους ημιαθροιστές μαζί με τα κρατούμενα εξόδους των προηγούμενων βαθμίδων. Το σχηματικό είναι το εξής:



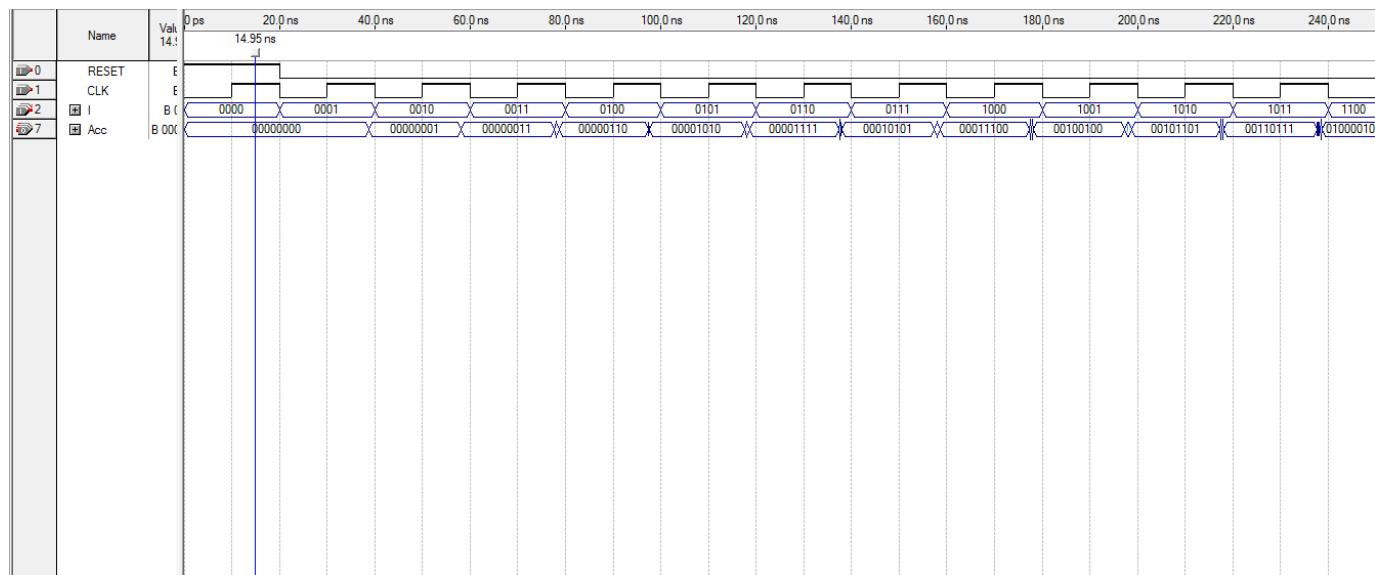
Επίσης, στις εισόδους θέσαμε συχνότητα περιόδου 80ns εμφανίζοντας το ακόλουθο διάγραμμα χρονισμού:



Τέλος, ο συσσωρευτής είναι το αριθμητικό κύκλωμα το οποίο αποθηκεύει την έξοδο του αθροιστή στον καταχωρήτη lpm_FF0. Επίσης, οι είσοδοι στον αθροιστή είναι τα bit εξόδου του παραπάνω καταχωρητή με 4 bit εισόδου. Οπότε το σχηματικό είναι το εξής:



Επιπλέον, στο διάγραμμα χρονισμού θέσαμε στο Reset την τιμή 1 μόνο για 20 ns, ενώ στο ρολόι θέσαμε περίοδο 20ns και αυτό φαίνεται στο παρακάτω waveform :

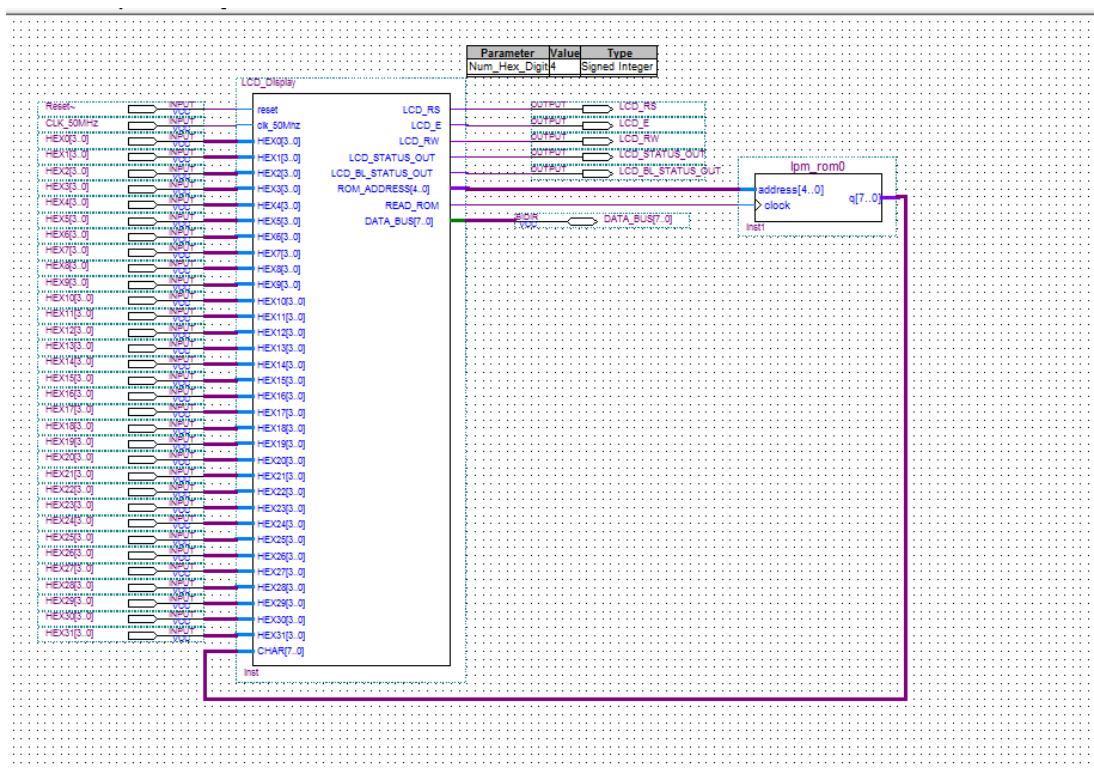


Τα δεδομένα εξόδου του συσσωρευτή μπορούμε να τα προβάλλουμε σε hexidecimal display μέσω του LCD. Για την λειτουργία του LCD αρχικοποιήσαμε το Reset στο 0 και συνδέσαμε στο PIN_N2 του board το ρολόι, το οποίο περιέχει συχνότητα 50MHz. Επίσης, συνδέσαμε τις κατάλληλες εισόδους HEX, οι οποίες αναπαριστούν θέσεις 32 αριθμών των 4 δυαδικών bit ο καθένα. Τέλος, βγάλαμε τις κατάλληλες εξόδους οι οποίες καθορίζουν τα σήματα χρονισμού του πρωτόκολλου επικοινωνίας που είναι το Display. Στη συνέχεια δημιουργήσαμε ένα αρχείο ROM.hex, το οποίο περιέχει χαρακτήρες ASCII σε

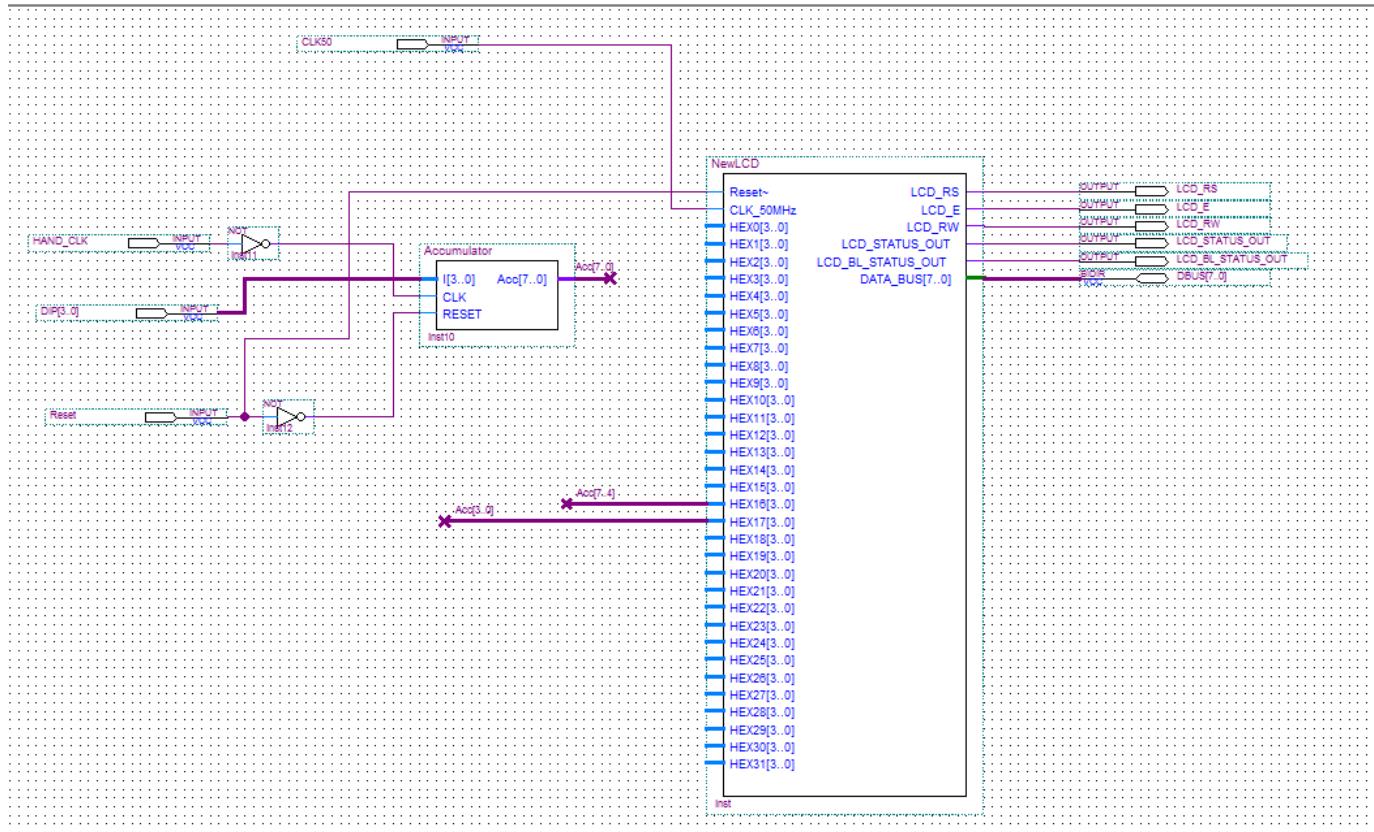
δεκαεξαδική μορφή. To αρχείο είναι το εξής:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	41	43	43	55	4D	55	4C	41
08	54	4F	52	20	20	20	20	20
10	00	00	20	20	20	20	20	20
18	20	20	20	20	20	20	20	20

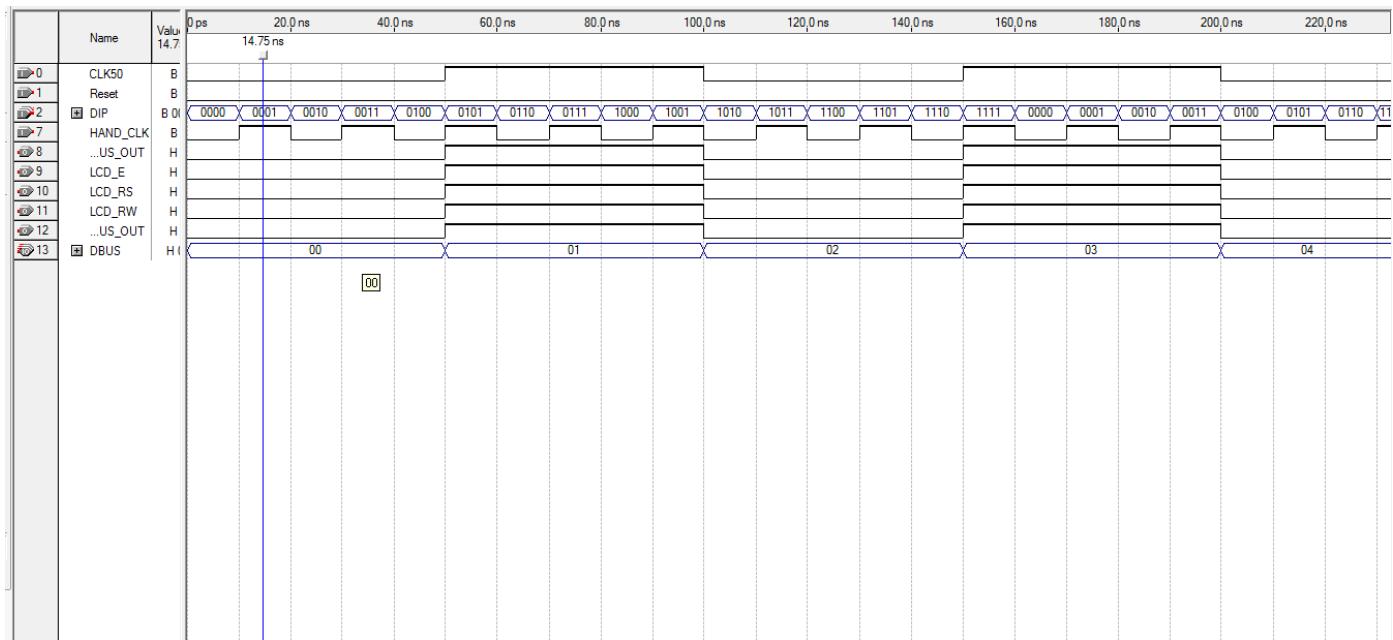
Οι παραπάνω χαρακτήρες απεικονίζουν την λέξη “ACCUMULATOR”. Το εσωτερικό του πυρήνα NewLCD φαίνεται παρακάτω:



όπου το αρχείο ROM.hex, το τοποθετήσαμε μέσα στο LPM_ROM, για να προβληθούν τα στοιχεία του παραπάνω αρχείου. Το τελικό μας κύκλωμα είναι το εξής:



Στο διάγραμμα χρονισμού τοποθετήσαμε στις εισόδους DIP μεταβολή της τιμής σε περίοδο 10ns και το HAND_CLK σε περίοδο 10ns, όπως φαίνεται στην παρακάτω εικόνα:



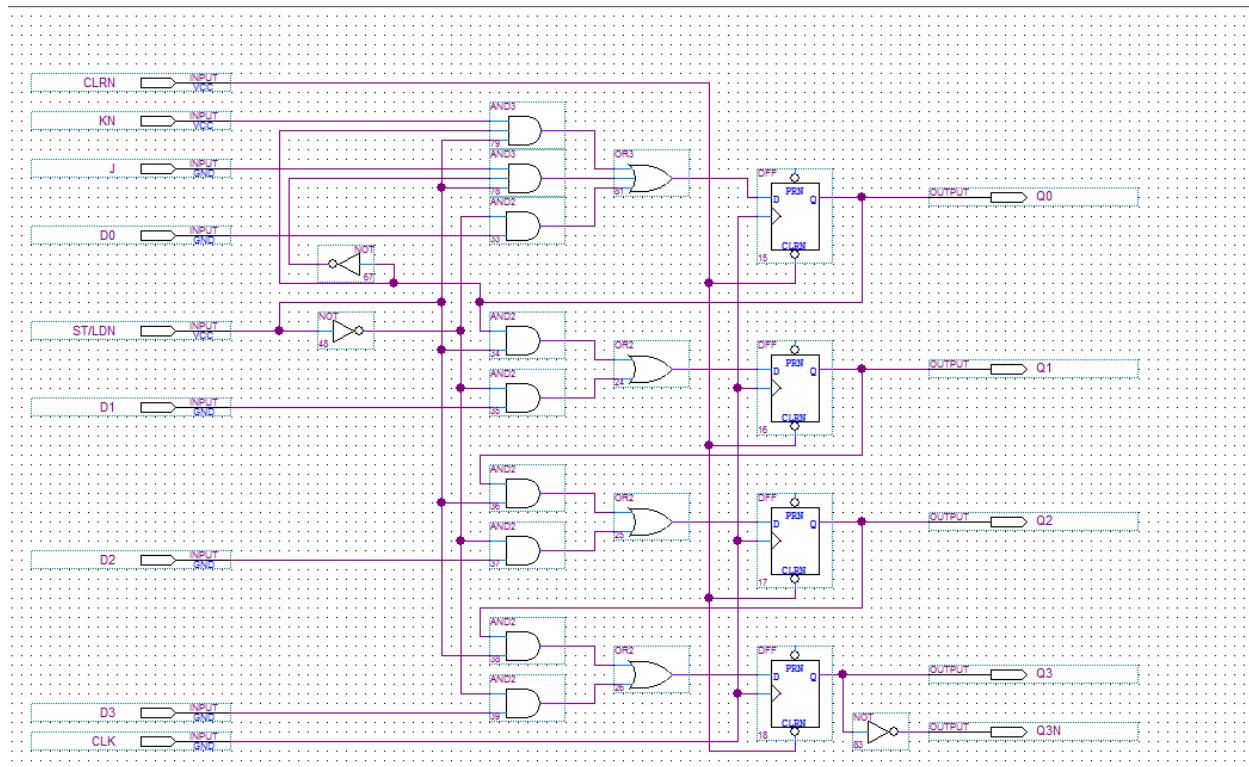
Τέλος, στην παρακάτω εικόνα φαίνονται όλα τα pins εισόδου και εξόδου:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserv
1	CLK50	Input	PIN_N2	2	B2_N1	3.3-V LVTTL (default)	
2	DBUS[7]	Bidir	PIN_H3	2	B2_N1	3.3-V LVTTL (default)	
3	DBUS[6]	Bidir	PIN_H4	2	B2_N1	3.3-V LVTTL (default)	
4	DBUS[5]	Bidir	PIN_J3	2	B2_N1	3.3-V LVTTL (default)	
5	DBUS[4]	Bidir	PIN_J4	2	B2_N1	3.3-V LVTTL (default)	
6	DBUS[3]	Bidir	PIN_H2	2	B2_N1	3.3-V LVTTL (default)	
7	DBUS[2]	Bidir	PIN_H1	2	B2_N1	3.3-V LVTTL (default)	
8	DBUS[1]	Bidir	PIN_J2	2	B2_N1	3.3-V LVTTL (default)	
9	DBUS[0]	Bidir	PIN_J1	2	B2_N1	3.3-V LVTTL (default)	
10	DIP[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
11	DIP[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
12	DIP[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
13	DIP[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
14	HAND_CLK	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)	
15	LCD_BL_STATUS_OUT	Output	PIN_K2	2	B2_N1	3.3-V LVTTL (default)	
16	LCD_E	Output	PIN_K3	2	B2_N1	3.3-V LVTTL (default)	
17	LCD_RS	Output	PIN_K1	2	B2_N1	3.3-V LVTTL (default)	
18	LCD_RW	Output	PIN_K4	2	B2_N1	3.3-V LVTTL (default)	
19	LCD_STATUS_OUT	Output	PIN_L4	2	B2_N1	3.3-V LVTTL (default)	
20	Reset	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)	
21	<<new node>>						

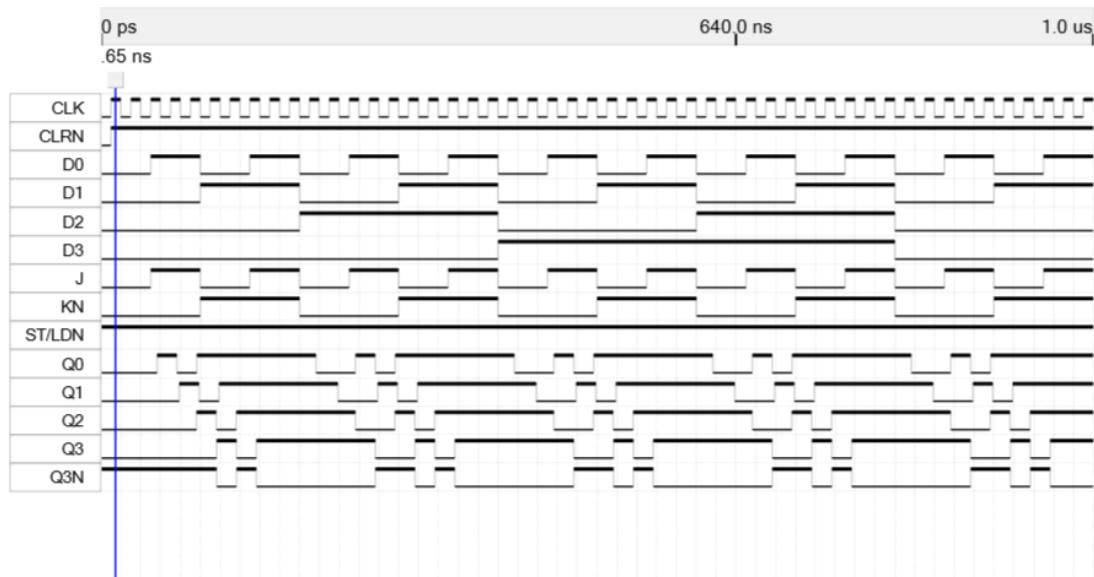
Εργαστηριακή Άσκηση 3 : Σχεδίαση Αλγορίθμικής Μηχανής Κατάστασης

Όπως είδαμε στην 3^η εργαστηριακή άσκηση, η σχεδίαση με χρήση του Gated-Clock μπορεί να προκαλέσει προβλήματα χρονισμού. Ωστόσο, αυτά τα προβλήματα μπορούμε να τα παραλείψουμε, χρησιμοποιώντας καλύτερο τρόπο σχεδίασης.

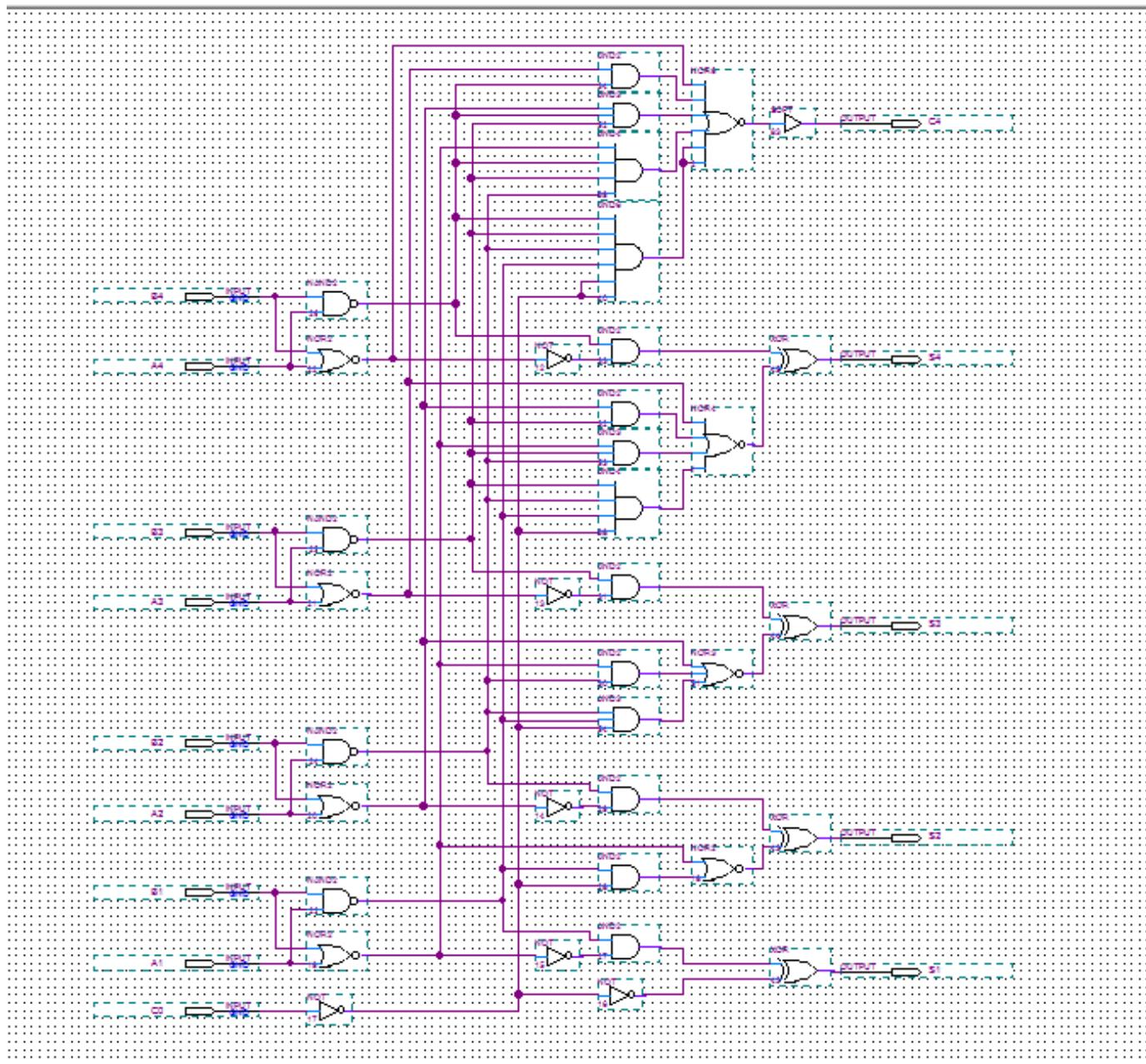
Αρχικά, αναλύσαμε όλες τις μονάδες του Data Path. Πρώτα από όλα, το δομικό στοιχείο του Data Path είναι ο καταχωρητής:



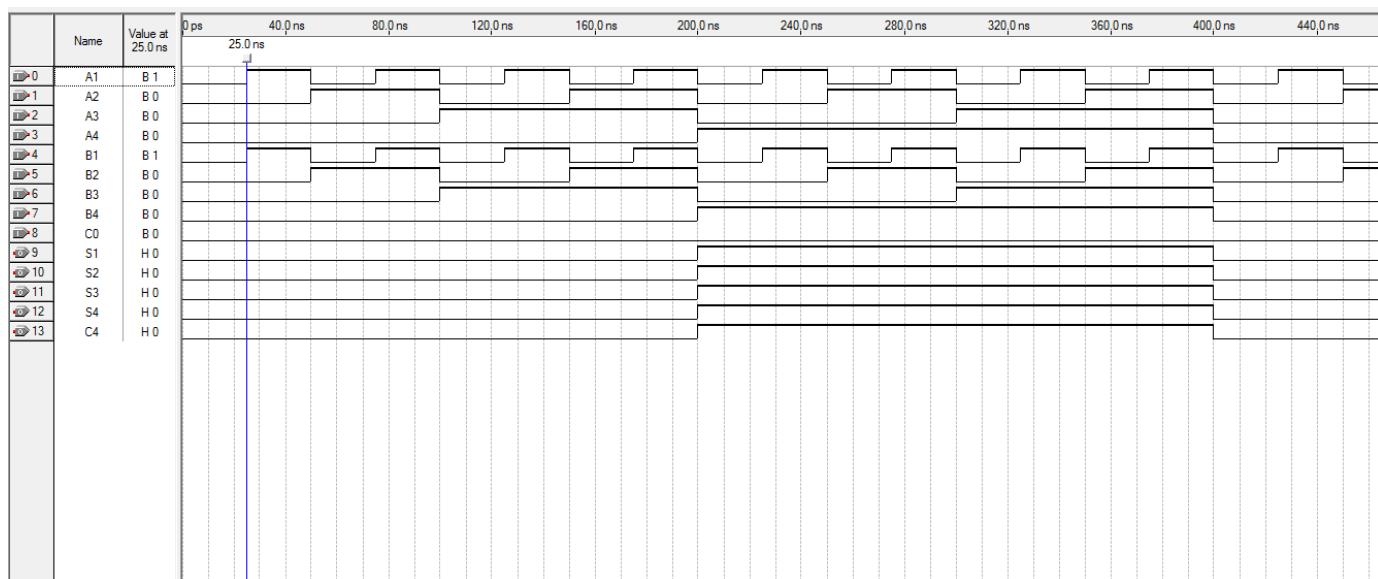
Ο συγκεκριμένος είναι το chip SN74LS195A, το οποίο χρησιμοποιήσαμε στο εργαστήριο. Για το διάγραμμα χρονισμού, θέσαμε το ρολόι σε περίοδο 20ns και στο CLRN θέσαμε την τιμή 0 στο διάστημα [0ps,10ns] και στην συνέχεια πήρε την τιμή 1. Επιπλέον, στην είσοδο D0 θέσαμε περίοδο αλλαγής 50ns, στην D1 100ns, στην D2 200ns και στην D3 400ns. Το SH\LDN ΤΟ έχουμε στην 1 και τα J,KN στα 50ns και 100ns αντίστοιχα. Το διάγραμμα φαίνεται παρακάτω:



Στη συνέχεια έχουμε τον αθροιστή του κυκλώματος, ο οποίος είναι το chip SN74LS83. Το κύκλωμα είναι το εξής:

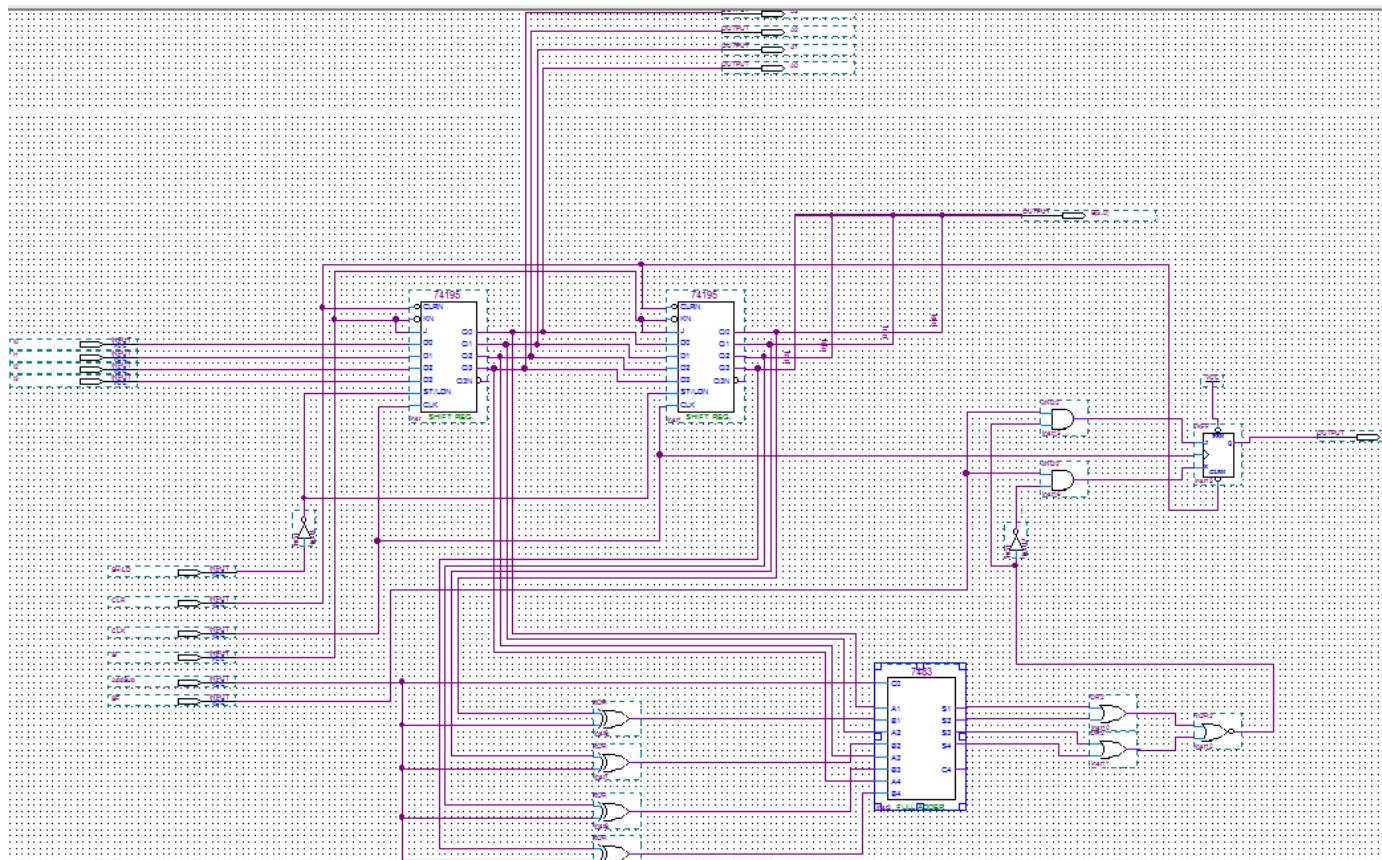


Το διάγραμμα χρονισμού είναι το εξής:



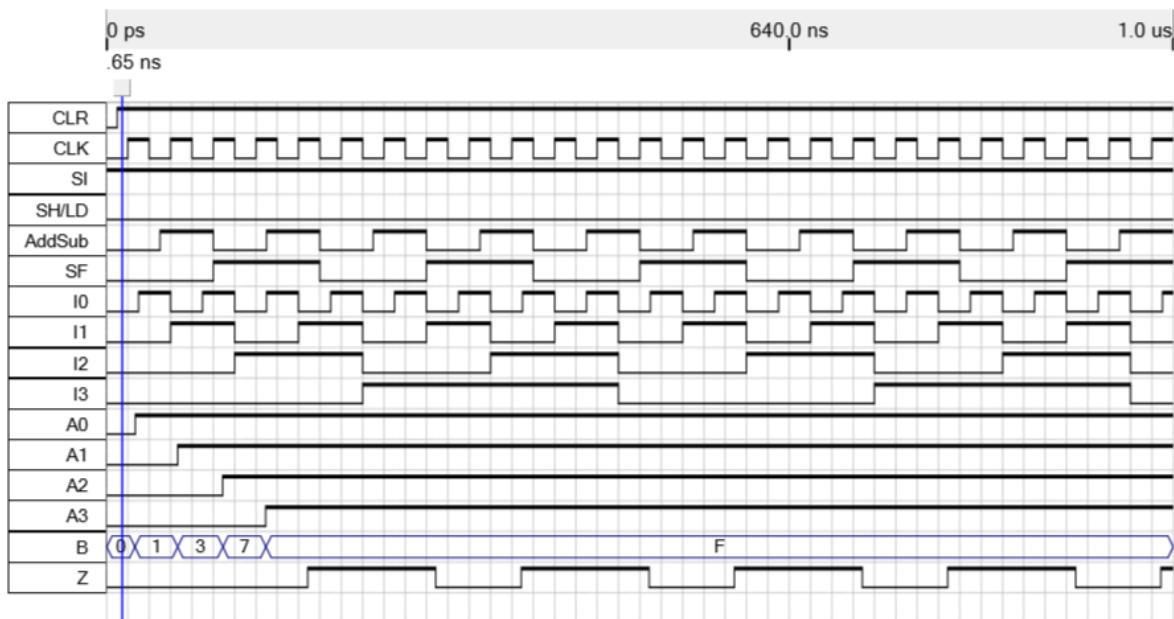
Θέσαμε τις τιμές των A1,B1 με περίοδο αλλαγής 50 ns, των A2,B2 στα 100ns, των A3,B3 στα 200 και των A4,B4 στα 400 ns, ενώ το κρατούμενο εισόδου C0 το είχαμε μόνιμα στο 0.

Μετά από αυτό σχεδιάσαμε το Data Path, εφαρμόζοντας τις αλλαγές που μας έδωσε η εκφώνηση. Το κύκλωμα φαίνεται παρακάτω:

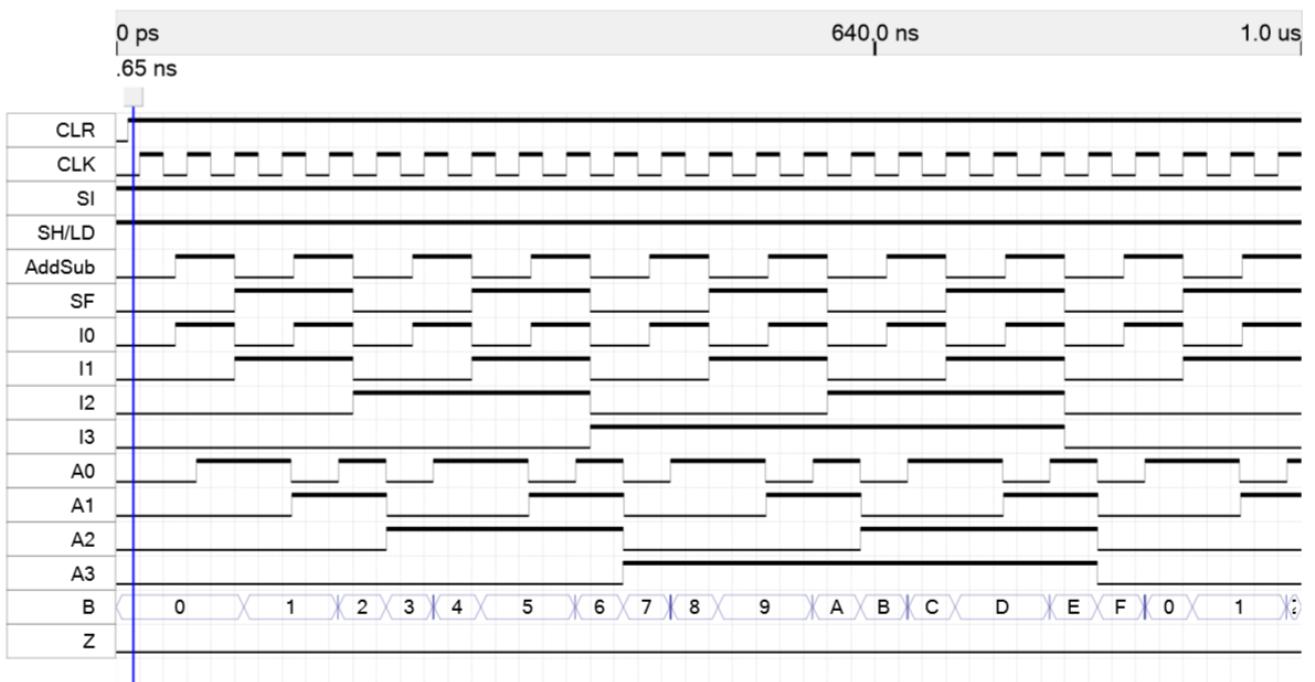


Το διάγραμμα χρονισμού είναι το εξής:

-Όταν το SH\LD=0 τότε:

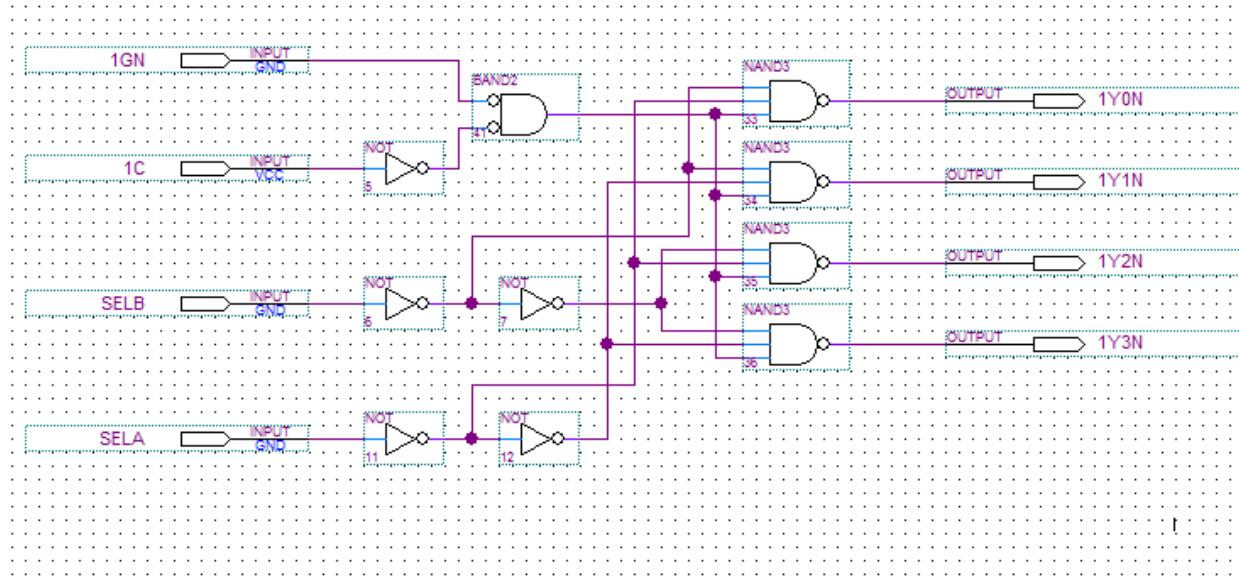


-Όταν $SH \setminus LD = 1$ τότε:

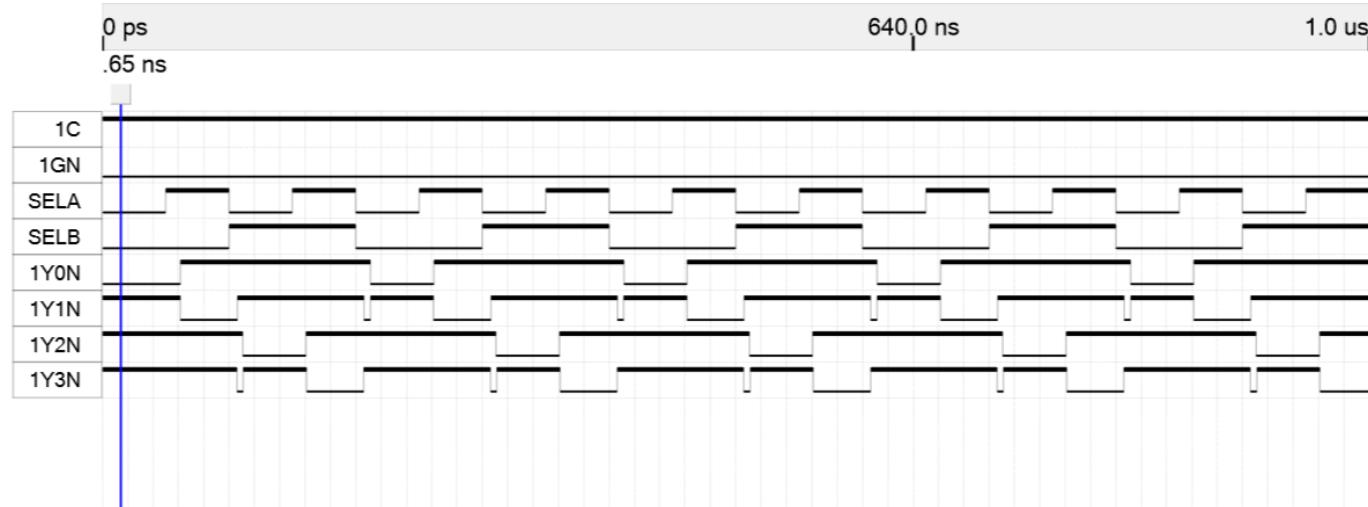


Οι τιμές των υπόλοιπων εισόδων φαίνεται στο παραπάνω waveform.

Το επόμενο κομμάτι που χρειάζεται το Data Path για να λειτουργεί σωστά, είναι η μονάδα ελέγχου. Πρώτα απ' όλα, η μονάδα ελέγχου περιέχει έναν αποκωδικοποιητή 2-σε-4, ο οποίος φαίνεται παρακάτω:

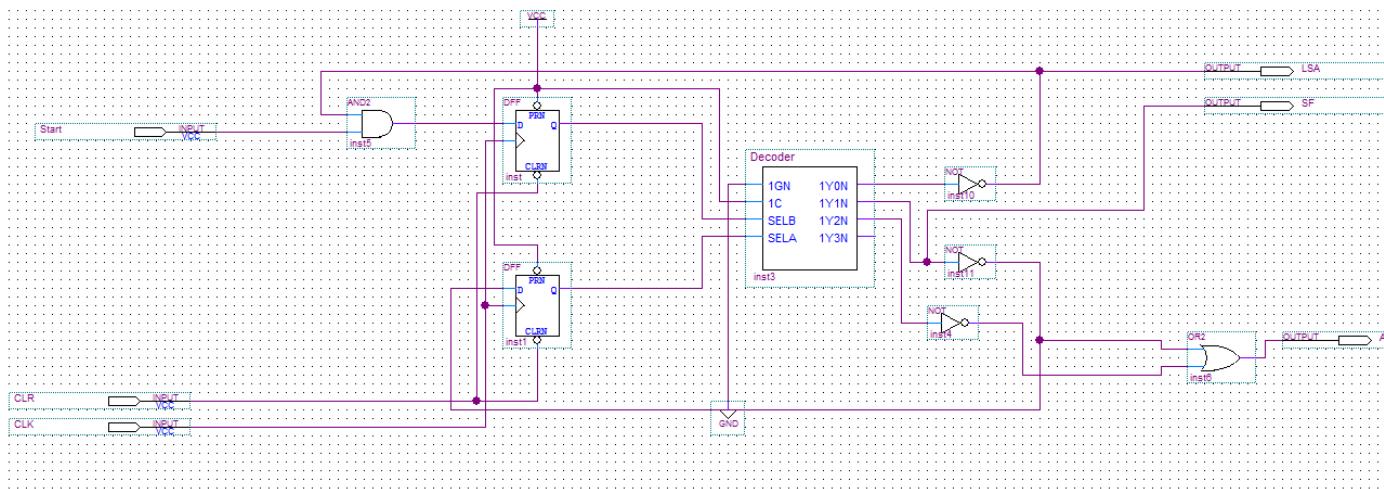


Το chip που χρησιμοποιήσαμε είναι το SN74LS155. Η κυματομορφή του φαίνεται παρακάτω:



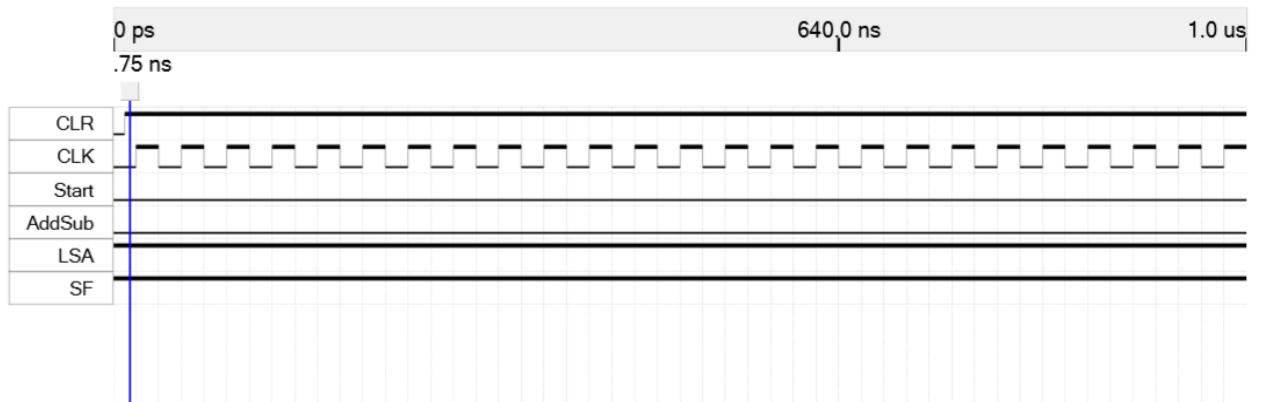
Όπου το SELB είναι η είσοδος με το σημαντικότερο bit(2^1) και το SELA είναι η είσοδος με το λιγότερο σημαντικό bit(2^0).

Το κύκλωμα της μονάδας ελέγχου είναι το παρακάτω:

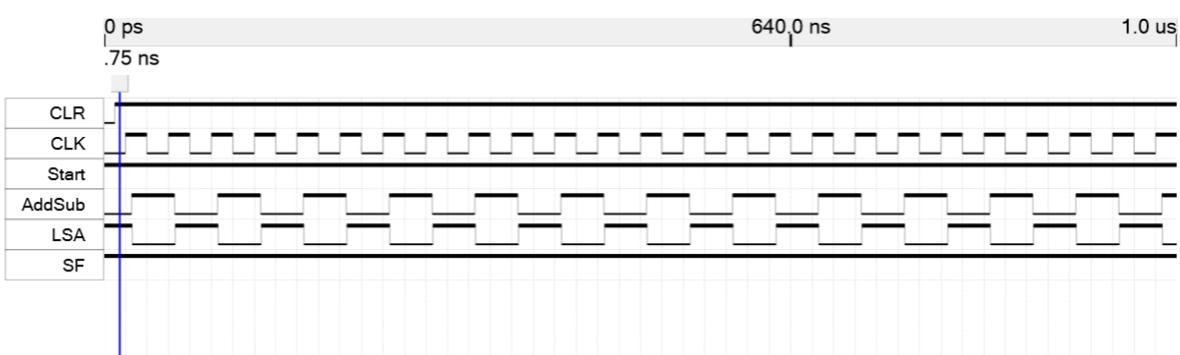


Το διάγραμμα χρονισμού είναι το εξής:

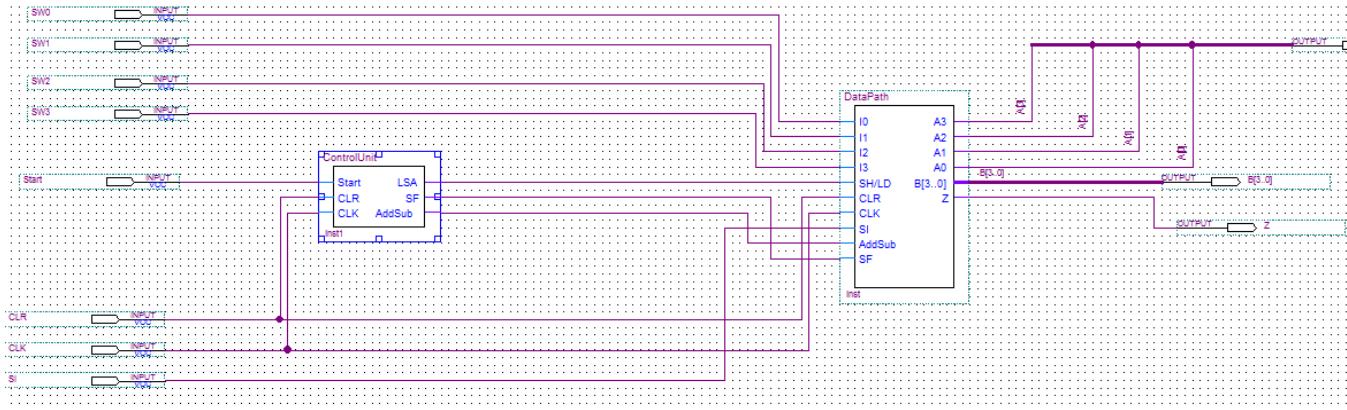
-Όταν το Start=0 τότε:



-Όταν το Start=1 τότε:

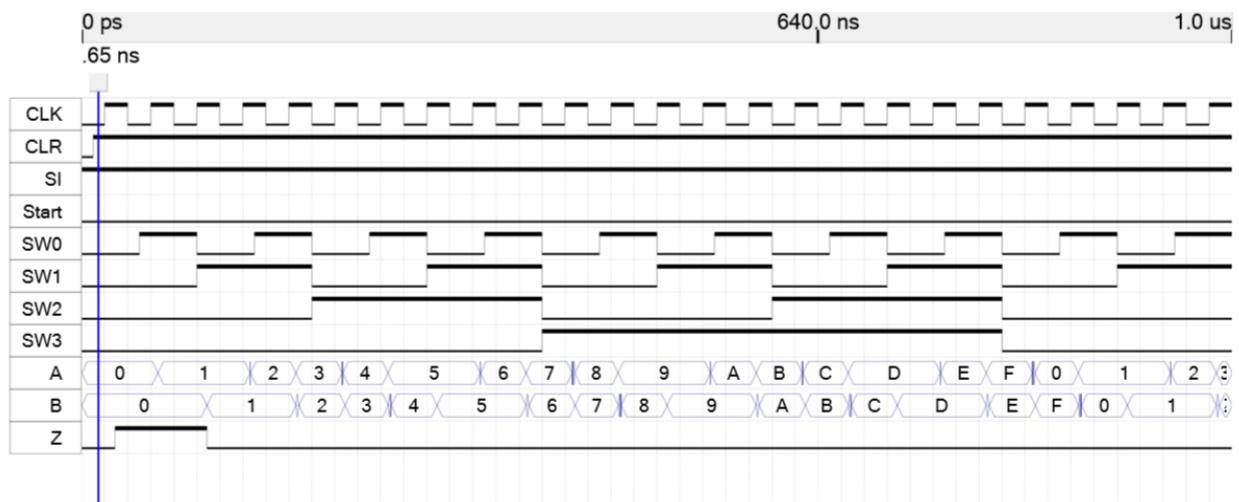


Οπότε, ολόκληρη η μηχανή φαίνεται στο παρακάτω σχήμα:

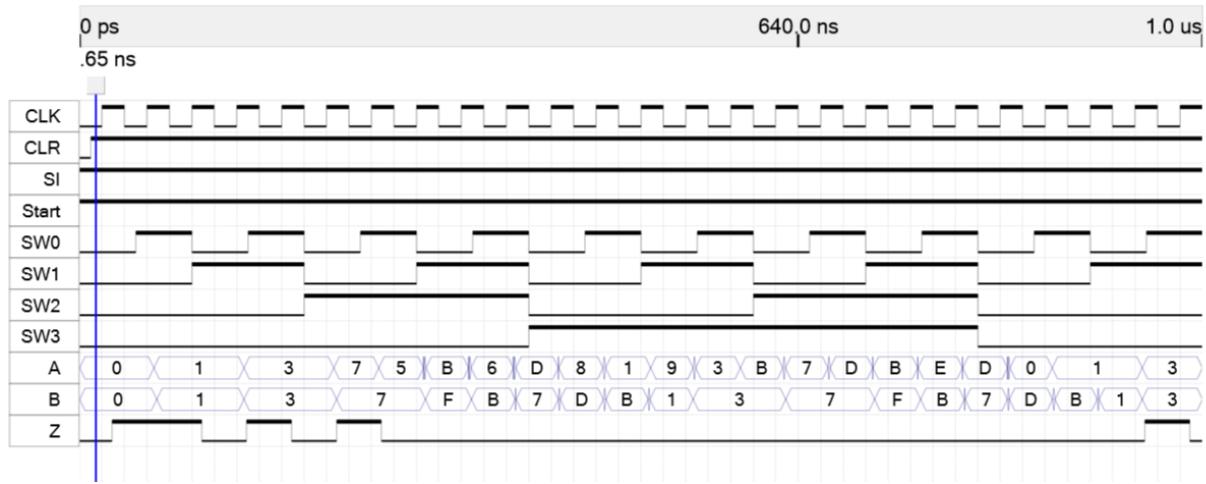


Και το διάγραμμα χρονισμού είναι το εξής:

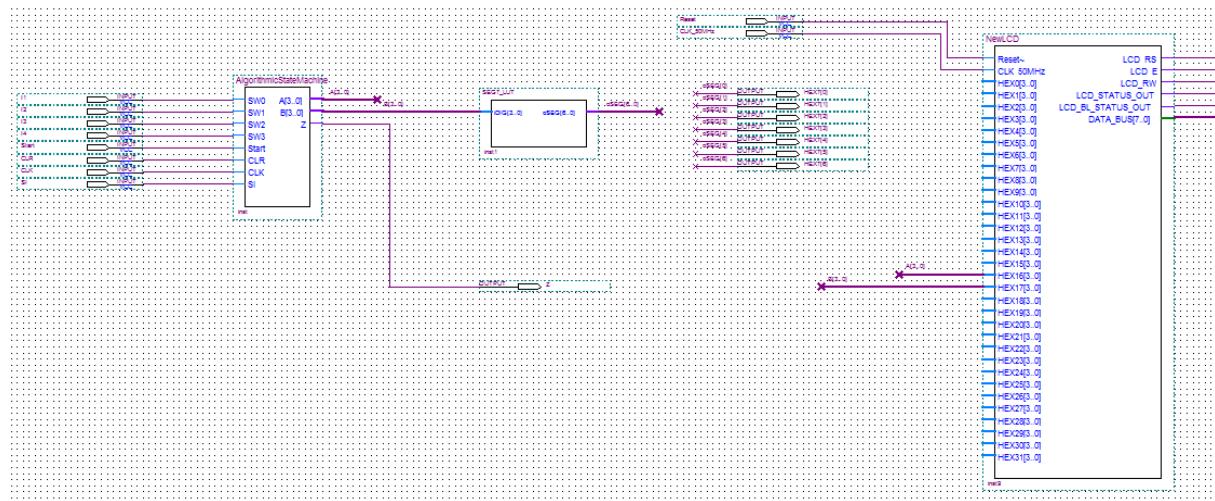
-Όταν το Start=0 τότε:



-Όταν το Start=1 τότε:



Τέλος, τα αποτελέσματα του καταχωρητή Α και Β θα πρέπει να προβληθούν στο LCD και τα αποτελέσματα του Β στο 7-segment display. Αυτό γίνεται ακολούθως στο παρακάτω κύκλωμα:



Επίσης, για το LCD δημιουργήσαμε ένα ROM1.hex αρχείο το οποίο τυπώνει την λέξη “ MACHINE”,όπου φαίνεται παρακάτω:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	4D	41	43	48	69	4E	45	20
8	20	20	20	20	20	20	20	20
16	00	00	20	20	20	20	20	20
24	20	20	20	20	20	20	20	20

Επιπλέον, παρακάτω φαίνονται και τα pins, τα οποία χρησιμοποιήσαμε:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	CLK	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)	
2	CLK_50MHz	Input	PIN_N2	2	B2_N1	3.3-V LVTTL (default)	
3	CLR	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)	
4	DBUS[7]	Bidir	PIN_H3	2	B2_N1	3.3-V LVTTL (default)	
5	DBUS[6]	Bidir	PIN_H4	2	B2_N1	3.3-V LVTTL (default)	
6	DBUS[5]	Bidir	PIN_J3	2	B2_N1	3.3-V LVTTL (default)	
7	DBUS[4]	Bidir	PIN_J4	2	B2_N1	3.3-V LVTTL (default)	
8	DBUS[3]	Bidir	PIN_H2	2	B2_N1	3.3-V LVTTL (default)	
9	DBUS[2]	Bidir	PIN_H1	2	B2_N1	3.3-V LVTTL (default)	
10	DBUS[1]	Bidir	PIN_J2	2	B2_N1	3.3-V LVTTL (default)	
11	DBUS[0]	Bidir	PIN_J1	2	B2_N1	3.3-V LVTTL (default)	
12	HEX7[6]	Output	PIN_N9	2	B2_N1	3.3-V LVTTL (default)	
13	HEX7[5]	Output	PIN_P9	2	B2_N1	3.3-V LVTTL (default)	
14	HEX7[4]	Output	PIN_L7	2	B2_N1	3.3-V LVTTL (default)	
15	HEX7[3]	Output	PIN_L6	2	B2_N1	3.3-V LVTTL (default)	
16	HEX7[2]	Output	PIN_L9	2	B2_N1	3.3-V LVTTL (default)	
17	HEX7[1]	Output	PIN_L2	2	B2_N1	3.3-V LVTTL (default)	
18	HEX7[0]	Output	PIN_L3	2	B2_N1	3.3-V LVTTL (default)	
19	I1	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
20	I2	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
21	I3	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
22	I4	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	

Groups

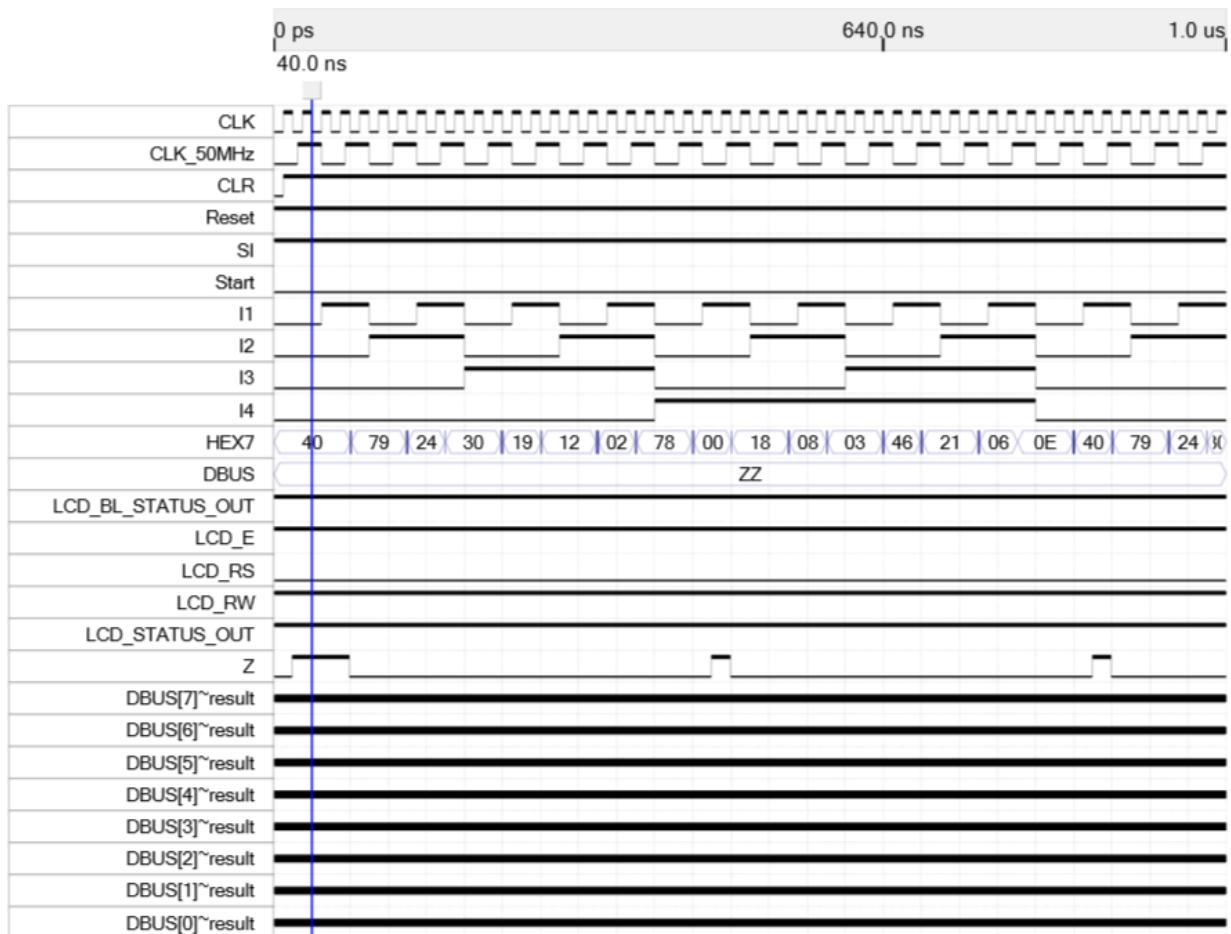
Top View - Wire Bond

Και

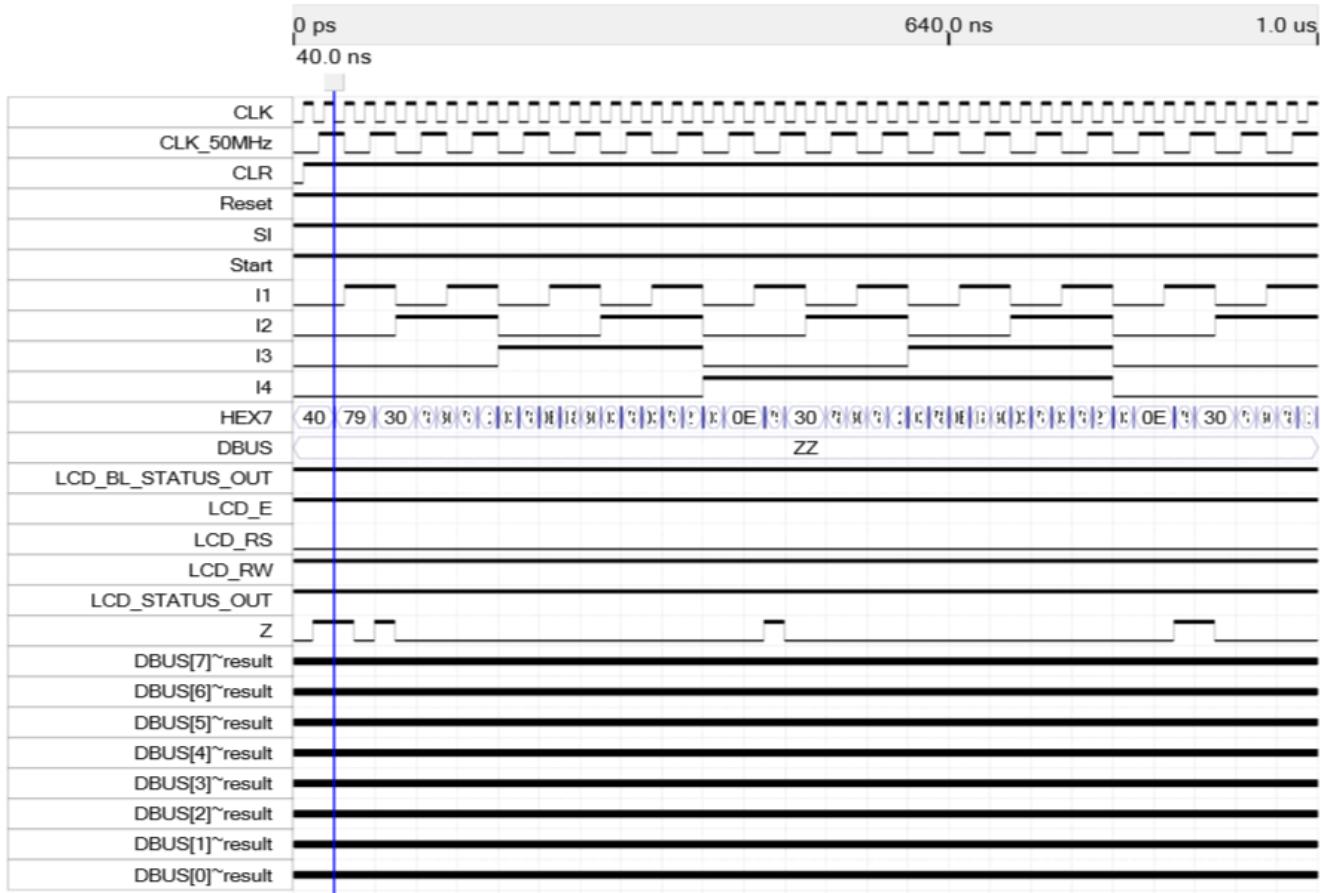
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
11	DBUS[0]	Bidir	PIN_J1	2	B2_N1	3.3-V LVTTL (default)		
12	HEX7[6]	Output	PIN_N9	2	B2_N1	3.3-V LVTTL (default)		
13	HEX7[5]	Output	PIN_P9	2	B2_N1	3.3-V LVTTL (default)		
14	HEX7[4]	Output	PIN_L7	2	B2_N1	3.3-V LVTTL (default)		
15	HEX7[3]	Output	PIN_L6	2	B2_N1	3.3-V LVTTL (default)		
16	HEX7[2]	Output	PIN_L9	2	B2_N1	3.3-V LVTTL (default)		
17	HEX7[1]	Output	PIN_L2	2	B2_N1	3.3-V LVTTL (default)		
18	HEX7[0]	Output	PIN_L3	2	B2_N1	3.3-V LVTTL (default)		
19	I1	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)		
20	I2	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
21	I3	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
22	I4	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)		
23	LCD_BL_STATUS_OUT	Output	PIN_K2	2	B2_N1	3.3-V LVTTL (default)		
24	LCD_E	Output	PIN_K3	2	B2_N1	3.3-V LVTTL (default)		
25	LCD_RS	Output	PIN_K1	2	B2_N1	3.3-V LVTTL (default)		
26	LCD_RW	Output	PIN_K4	2	B2_N1	3.3-V LVTTL (default)		
27	LCD_STATUS_OUT	Output	PIN_L4	2	B2_N1	3.3-V LVTTL (default)		
28	Reset	Input	PIN_P23	6	B6_N0	3.3-V LVTTL (default)		
29	SI	Input	PIN_W26	6	B6_N1	3.3-V LVTTL (default)		
30	Start	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)		
31	Z	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
32	<<new node>>							

Το διάγραμμα χρονισμού της μηχανής θα είναι το εξής:

-Όταν το Start=0 τότε:



-Όταν το Start=1 τότε:



Εργαστηριακή Ασκηση 4: Σχεδίαση με VHDL

Στην συγκεκριμένη εργαστηριακή μάθαμε πώς να σχεδιάσουμε απλά λογικά κυκλώματα βάσει της γλώσσας περιγραφής υλικού VHDL.

Μέρος 1^ο. Συνδυαστικά κυκλώματα

Ερώτημα 1^ο:

Σε αυτό το ερώτημα σχεδιάσαμε έναν πολυπλέκτη MUX 4x1. Πρώτα δημιουργήσαμε καινούργιο project με το ίδιο όνομα με το σχήμα. Πρώτα απ' όλα, θα δημιουργήσουμε ένα νέο αρχείο VHDL(*File>New>VHDL File*). Στη συνέχεια, αποθηκεύσαμε το κενό αρχείο με το όνομα MUX4_1.vhdl. Παρακάτω έχουμε τον κώδικα της παραπάνω οντότητας(δηλ. του πολυπλέκτη):

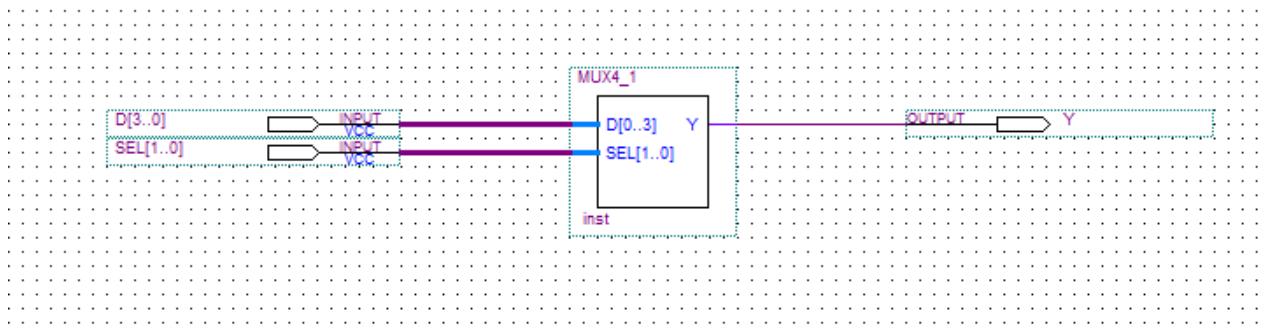
```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity MUX4_1 is
5    port(
6      D: in std_logic_vector (0 to 3);
7      SEL: in std_logic_vector (1 downto 0);
8      Y: out std_logic);
9  end MUX4_1;
10
11 architecture RTL of MUX4_1 is
12 begin
13   Y <= D(0) when SEL="00" else D(1) when SEL="01" else D(2) when SEL="10" else D(3);
14
15

```

Τέλος, αφού δημιουργήσαμε την παραπάνω οντότητα, τότε δημιουργήσαμε το σύμβολο του πολυπλέκτη. Επίσης, δημιουργήσαμε και το component του πολυπλέκτη μέσο του *File>Create/Update>Create VHDL Component Declaration Files for Current File*.

Τώρα, το παρακάτω σχήμα περιέχει το block του πολυπλέκτη, το οποίο δημιουργήσαμε προηγουμένως.



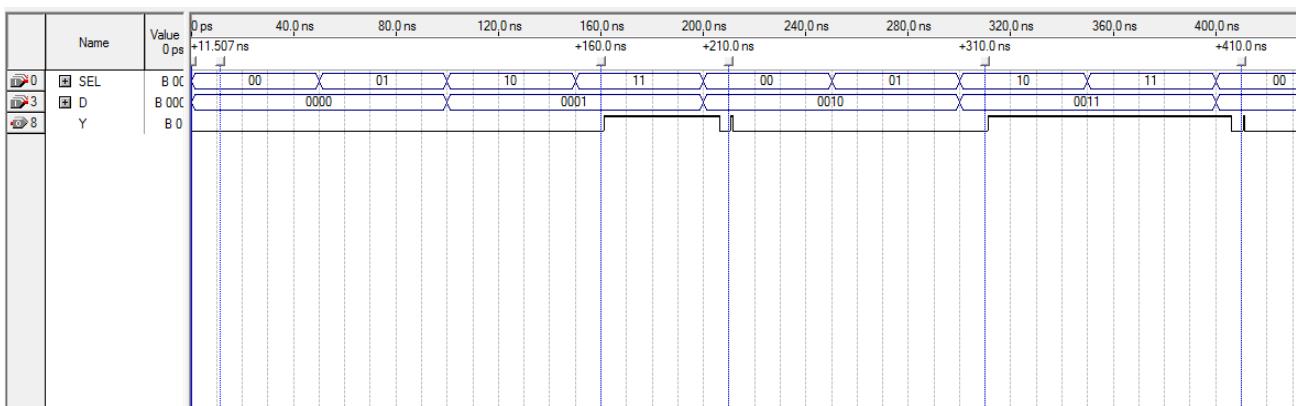
Επιπλέον, το παραπάνω σχηματικό περιέχει δύο εισόδους και μία έξοδο, αντίστοιχα και παρακάτω φαίνονται τα επιλεγμένα pins.

	Named:	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1		D[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
2		D[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
3		D[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
4		D[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
5		SEL[1]	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)	
6		SEL[0]	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)	
7		Y	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
8		<<new node>>						

Πριν κάνουμε την εξομοίωση, πρώτα θα βρούμε την καθυστέρηση του κυκλώματος, κάνοντας στατική χρονική ανάλυση όπως φαίνεται παρακάτω.

	Registered	Performance	tpd	tsu	tco	th	Custom Delays
	Slack	Required P2P Time	Actual P2P Time	From	To		
1	N/A	None	11.507 ns	SEL[0]	Y		
2	N/A	None	11.381 ns	SEL[1]	Y		
3	N/A	None	8.721 ns	D[3]	Y		
4	N/A	None	7.201 ns	D[1]	Y		
5	N/A	None	6.522 ns	D[2]	Y		
6	N/A	None	6.349 ns	D[0]	Y		

Τώρα, για την εξομοίωση του παραπάνω σχήματος θέσαμε στην είσοδο SEL την τιμή 1 με περίοδο αλλαγής 50ns και στην είσοδο D θέσαμε την τιμή 1 με περίοδο αλλαγής 100ns. Παρακάτω έχουμε την κυματομορφή του πολυπλέκτη μετά την εξομοίωση.



Ερώτημα 2^ο:

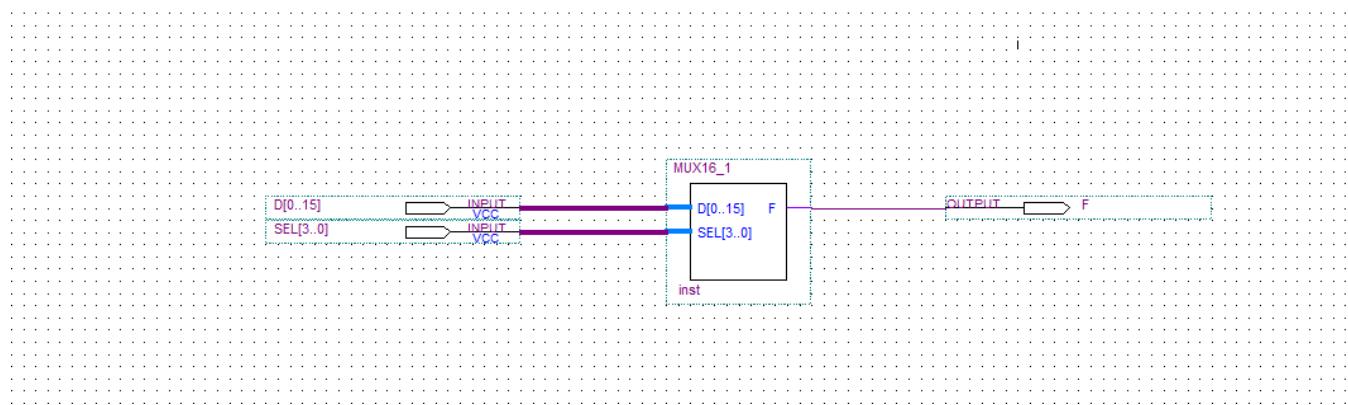
Στο δεύτερο ερώτημα υλοποιήσαμε έναν πολυπλέκτη 16x1 σε γλώσσα VHDL, χρησιμοποιώντας τον πολυπλέκτη που δημιουργήσαμε στο προηγούμενο ερώτημα. Παρακάτω φαίνεται ο κώδικας του νέου πολυπλέκτη:

```

1  library IEEE;
2  use ieee.std_logic_1164.all;
3
4  ENTITY MUX16_1 IS port(
5      D: in std_logic_vector (0 to 15);
6      SEL: in std_logic_vector (3 downto 0);
7      F: out std_logic);
8  END MUX16_1;
9
10 =ARCHITECTURE RTL OF MUX16_1 IS
11 =COMPONENT MUX4_1
12 =    port(
13         D: in std_logic_vector (0 to 3);
14         SEL: in std_logic_vector (1 downto 0);
15         Y: out std_logic);
16 END COMPONENT;
17 SIGNAL Y : STD_LOGIC_VECTOR(0 to 3);
18
19 BEGIN
20
21 u0: MUX4_1 port map(D=>D(0 to 3), SEL=>SEL(1 downto 0), Y=>Y(0));
22 u1: MUX4_1 port map(D=>D(4 to 7), SEL=>SEL(1 downto 0), Y=>Y(1));
23 u2: MUX4_1 port map(D=>D(8 to 11), SEL=>SEL(1 downto 0), Y=>Y(2));
24 u3: MUX4_1 port map(D=>D(12 to 15), SEL=>SEL(1 downto 0), Y=>Y(3));
25 u4: MUX4_1 port map(D=>Y, SEL=>SEL(3 downto 2), Y=>F);
26
27 END RTL;

```

Στη συνέχεια δημιουργήσαμε το block του παραπάνω πολυπλέκτη, χρησιμοποιώντας ως component τον πολυπλέκτη 4x1. Οπότε, παρακάτω φαίνεται το κύκλωμα:



Κατόπιν, ορίσαμε τα pins για τις εισόδους και εξόδους:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	D[15]	Input	PIN_V2	1	B1_N0	3.3-V LVTTL (default)	
2	D[14]	Input	PIN_V1	1	B1_N0	3.3-V LVTTL (default)	
3	D[13]	Input	PIN_U4	1	B1_N0	3.3-V LVTTL (default)	
4	D[12]	Input	PIN_U3	1	B1_N0	3.3-V LVTTL (default)	
5	D[11]	Input	PIN_T7	1	B1_N0	3.3-V LVTTL (default)	
6	D[10]	Input	PIN_P2	1	B1_N0	3.3-V LVTTL (default)	
7	D[9]	Input	PIN_P1	1	B1_N0	3.3-V LVTTL (default)	
8	D[8]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)	
9	D[7]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)	
10	D[6]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)	
11	D[5]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)	
12	D[4]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)	
13	D[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
14	D[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
15	D[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
16	D[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
17	F	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
18	SEL[3]	Input	PIN_W26	6	B6_N1	3.3-V LVTTL (default)	
19	SEL[2]	Input	PIN_P23	6	B6_N0	3.3-V LVTTL (default)	
20	SEL[1]	Input	PIN_N23	5	B5_N1	3.3-V LVTTL (default)	
21	SEL[0]	Input	PIN_G26	5	B5_N0	3.3-V LVTTL (default)	
22	Y	Unknown	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	

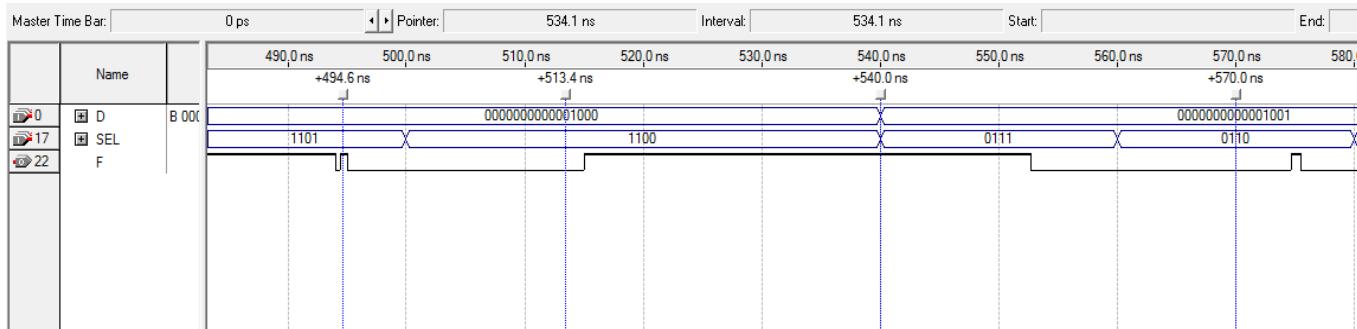
Πριν εκτελέσουμε την εξομοίωση, πρώτα κάναμε στατική χρονική ανάλυση για να βρούμε την μέγιστη καθυστέρηση, όπου φαίνεται παρακάτω:

Registered Performance						
	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	14.975 ns	SEL[0]	F	
2	N/A	None	14.856 ns	D[11]	F	
3	N/A	None	14.336 ns	D[14]	F	
4	N/A	None	14.205 ns	SEL[1]	F	
5	N/A	None	14.182 ns	D[15]	F	
6	N/A	None	13.688 ns	D[12]	F	
7	N/A	None	13.660 ns	D[13]	F	
8	N/A	None	13.460 ns	SEL[2]	F	
9	N/A	None	13.374 ns	SEL[3]	F	
10	N/A	None	10.292 ns	D[9]	F	
11	N/A	None	10.212 ns	D[5]	F	
12	N/A	None	10.185 ns	D[1]	F	
13	N/A	None	10.156 ns	D[2]	F	
14	N/A	None	9.910 ns	D[6]	F	
15	N/A	None	9.801 ns	D[10]	F	

και

	Slack	Required P2P Time	Actual P2P Time	From	To	
6	N/A	None	13.688 ns	D[12]	F	
7	N/A	None	13.660 ns	D[13]	F	
8	N/A	None	13.460 ns	SEL[2]	F	
9	N/A	None	13.374 ns	SEL[3]	F	
10	N/A	None	10.292 ns	D[9]	F	
11	N/A	None	10.212 ns	D[5]	F	
12	N/A	None	10.185 ns	D[1]	F	
13	N/A	None	10.156 ns	D[2]	F	
14	N/A	None	9.910 ns	D[6]	F	
15	N/A	None	9.801 ns	D[10]	F	
16	N/A	None	9.767 ns	D[3]	F	
17	N/A	None	9.677 ns	D[4]	F	
18	N/A	None	9.586 ns	D[7]	F	
19	N/A	None	9.494 ns	D[8]	F	
20	N/A	None	9.379 ns	D[0]	F	

Επομένως, για την παρακάτω χρονική εξομοίωση θέσαμε ως περίοδο αλλαγής στην είσοδο D 60ns και στην είσοδο SEL 20ns. Άρα παρακάτω έχουμε την εξομοίωση.



Ερώτημα 3^ο:

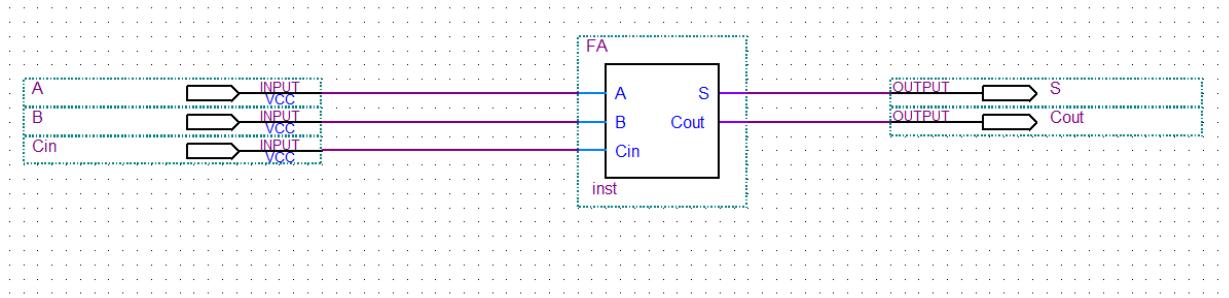
Στο 3^ο ερώτημα, αρχικά σχεδιάσαμε την οντότητα ενός πλήρη αθροιστή, ο οποίος φαίνεται παρακάτω:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity FA is
5 port(
6     A: in std_logic;
7     B: in std_logic;
8     Cin: in std_logic;
9     S: out std_logic;
10    Cout: out std_logic);
11 end FA;
12
13 architecture RTL of FA is
14 begin
15     S <= A XOR B XOR Cin;
16     Cout<= (A AND B) OR (B AND Cin) OR (A AND Cin);
17 end RTL;

```

Κατόπιν δημιουργήσαμε το block αλλά και το component του αθροιστή για το παρακάτω ερώτημα. Το σχηματικό είναι το εξής:



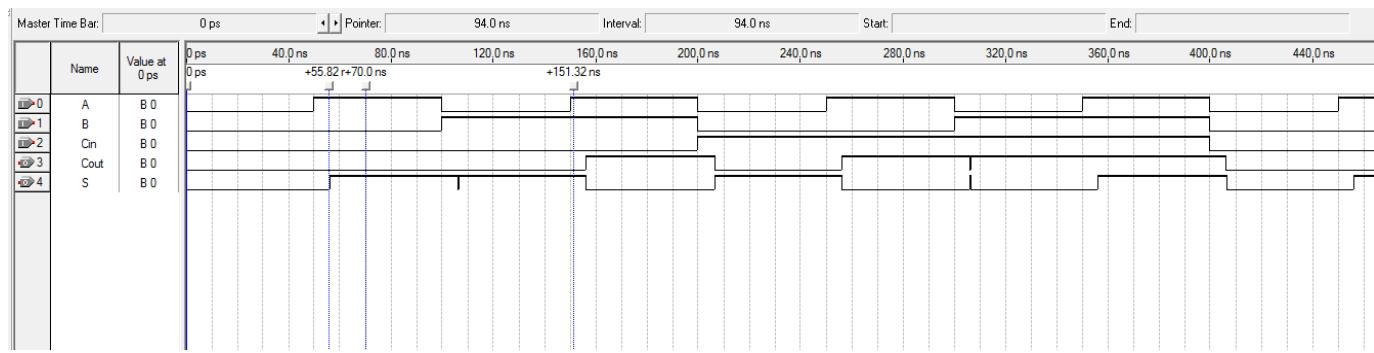
Τα pins τα οποία χρησιμοποιήσαμε είναι τα εξής:

Named:	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
1	A	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)		
2	B	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
3	Cin	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
4	Cout	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
5	S	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)		
6	<<new node>>							

Τώρα, εκτελούμε στατική χρονική ανάλυση για να βρούμε την μέγιστη καθυστέρηση, όπου φαίνεται στην παρακάτω εικόνα:

Registered Performance						
	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	6.537 ns	B	Cout	
2	N/A	None	6.518 ns	B	S	
3	N/A	None	6.357 ns	A	Cout	
4	N/A	None	6.343 ns	A	S	
5	N/A	None	6.227 ns	Cin	Cout	
6	N/A	None	6.214 ns	Cin	S	

Στη συνέχεια, δημιουργήσαμε ένα αρχείο κυματομορφών για να εκτελέσεσουμε χρονική εξομοίωση. Θέσαμε την τιμή 1 στις εισόδους A, B, Cin με περίοδο αλλαγής 50ns, 100ns και 200ns αντίστοιχα. Παρακάτω φαίνεται η εξομοίωση μαζί με τις καθυστερήσεις:



Ερώτημα 4^ο:

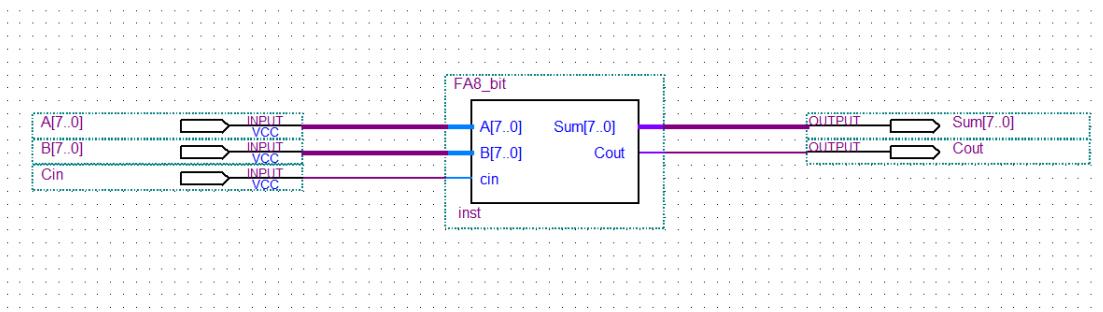
Σε αυτό το ερώτημα σχεδιάσαμε τον πλήρη αθροιστή των 8 bit με διαφορετικό τρόπο και συγκεκριμένα με τον συντελεστή ‘+’. Παρακάτω φαίνεται ο κώδικας:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_unsigned.all;
4
5 entity FA8_bit is
6   port(A: in std_logic_vector (7 downto 0);
7       B: in std_logic_vector (7 downto 0);
8       cin: in std_logic;
9       Sum: out std_logic_vector (7 downto 0);
10      Cout: out std_logic);
11 end FA8_bit;
12
13 architecture RTL of FA8_bit is
14   signal InternalSum: std_logic_vector (8 downto 0);
15 begin
16   process(A,B,cin)
17   begin
18     InternalSum<=A+B+"00000000"&cin;
19   end process;
20   Sum <= InternalSum(7 downto 0);
21   Cout <= InternalSum(8);
22 end RTL;

```

Στη συνέχεια δημιουργήσαμε το block, το οποίο φαίνεται παρακάτω:



Κατόπιν, ορίσαμε τα pins για τις εισόδους και εξόδους:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	A[7]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)	
2	A[6]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)	
3	A[5]	Input	PIN_A113	8	B8_N0	3.3-V LVTTL (default)	
4	A[4]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)	
5	A[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
6	A[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
7	A[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
8	A[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
9	B[7]	Input	PIN_U4	1	B1_N0	3.3-V LVTTL (default)	
10	B[6]	Input	PIN_U3	1	B1_N0	3.3-V LVTTL (default)	
11	B[5]	Input	PIN_T7	1	B1_N0	3.3-V LVTTL (default)	
12	B[4]	Input	PIN_P2	1	B1_N0	3.3-V LVTTL (default)	
13	B[3]	Input	PIN_P1	1	B1_N0	3.3-V LVTTL (default)	
14	B[2]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)	
15	B[1]	Input	PIN_A13	4	B4_N1	3.3-V LVTTL (default)	
16	B[0]	Input	PIN_B13	4	B4_N1	3.3-V LVTTL (default)	
17	Cin	Input	PIN_V1	1	B1_N0	3.3-V LVTTL (default)	
18	Cout	Output	PIN_AA14	7	B7_N1	3.3-V LVTTL (default)	
19	Sum[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)	
20	Sum[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)	
21	Sum[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)	
22	Sum[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)	
23	Sum[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)	
24	Sum[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)	
25	Sum[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
26	Sum[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
27	<<new node>>						

Και:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
6	A[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
7	A[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
8	A[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
9	B[7]	Input	PIN_U4	1	B1_N0	3.3-V LVTTL (default)	
10	B[6]	Input	PIN_U3	1	B1_N0	3.3-V LVTTL (default)	
11	B[5]	Input	PIN_T7	1	B1_N0	3.3-V LVTTL (default)	
12	B[4]	Input	PIN_P2	1	B1_N0	3.3-V LVTTL (default)	
13	B[3]	Input	PIN_P1	1	B1_N0	3.3-V LVTTL (default)	
14	B[2]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)	
15	B[1]	Input	PIN_A13	4	B4_N1	3.3-V LVTTL (default)	
16	B[0]	Input	PIN_B13	4	B4_N1	3.3-V LVTTL (default)	
17	Cin	Input	PIN_V1	1	B1_N0	3.3-V LVTTL (default)	
18	Cout	Output	PIN_AA14	7	B7_N1	3.3-V LVTTL (default)	
19	Sum[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)	
20	Sum[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)	
21	Sum[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)	
22	Sum[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)	
23	Sum[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)	
24	Sum[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)	
25	Sum[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
26	Sum[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
27	<<new node>>						

Επιπλέον, εκτελέσαμε στατική χρονική ανάλυση για να βρούμε τις μέγιστες καθυστερήσεις των εξόδων σε σχέση με των εισόδων. Παρακάτω φαίνονται οι καθυστερήσεις:

	Registered Performance	tpd	tsu	tco	th	Custom Delays
	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	12.529 ns	B[5]	Sum[6]	
2	N/A	None	12.238 ns	B[5]	Sum[7]	
3	N/A	None	11.730 ns	B[6]	Sum[7]	
4	N/A	None	11.564 ns	B[5]	Cout	
5	N/A	None	11.371 ns	B[6]	Cout	
6	N/A	None	10.998 ns	B[7]	Cout	
7	N/A	None	9.847 ns	Cin	Sum[0]	
8	N/A	None	9.709 ns	B[0]	Sum[6]	
9	N/A	None	9.659 ns	B[0]	Sum[3]	
10	N/A	None	9.515 ns	B[0]	Sum[2]	
11	N/A	None	9.512 ns	B[0]	Sum[4]	
12	N/A	None	9.493 ns	B[1]	Sum[6]	
13	N/A	None	9.484 ns	A[1]	Sum[6]	
14	N/A	None	9.443 ns	B[1]	Sum[3]	
15	N/A	None	9.434 ns	A[1]	Sum[3]	
..						

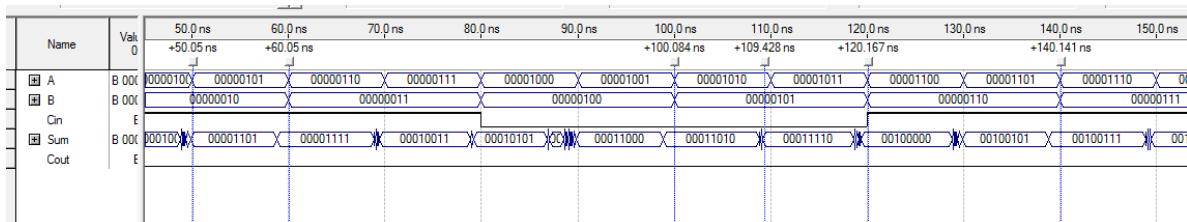
	Registered Performance	tpd	tsu	tco	th	Custom Delays
	Slack	Required P2P Time	Actual P2P Time	From	To	
16	N/A	None	9.370 ns	A[0]	Sum[6]	
17	N/A	None	9.339 ns	A[2]	Sum[6]	
18	N/A	None	9.320 ns	A[0]	Sum[3]	
19	N/A	None	9.296 ns	B[1]	Sum[4]	
20	N/A	None	9.287 ns	A[1]	Sum[4]	
21	N/A	None	9.181 ns	B[0]	Sum[1]	
22	N/A	None	9.176 ns	A[0]	Sum[2]	
23	N/A	None	9.173 ns	A[0]	Sum[4]	
24	N/A	None	9.142 ns	A[2]	Sum[4]	
25	N/A	None	9.103 ns	B[0]	Sum[7]	
26	N/A	None	8.986 ns	B[0]	Sum[5]	
27	N/A	None	8.982 ns	B[1]	Sum[2]	
28	N/A	None	8.974 ns	A[2]	Sum[3]	
29	N/A	None	8.970 ns	A[1]	Sum[2]	
30	N/A	None	8.887 ns	B[1]	Sum[7]	
..						

Registered Performance		tpd	tsu	tco	th	Custom Delays	
	Slack	Required P2P Time	Actual P2P Time	From	To		
31	N/A	None	8.878 ns	A[1]	Sum[7]		
32	N/A	None	8.877 ns	B[2]	Sum[6]		
33	N/A	None	8.846 ns	A[0]	Sum[1]		
34	N/A	None	8.812 ns	B[3]	Sum[6]		
35	N/A	None	8.770 ns	B[1]	Sum[5]		
36	N/A	None	8.764 ns	A[0]	Sum[7]		
37	N/A	None	8.761 ns	A[1]	Sum[5]		
38	N/A	None	8.752 ns	B[4]	Sum[6]		
39	N/A	None	8.733 ns	A[2]	Sum[7]		
40	N/A	None	8.680 ns	B[2]	Sum[4]		
41	N/A	None	8.647 ns	A[0]	Sum[5]		
42	N/A	None	8.616 ns	A[2]	Sum[5]		
43	N/A	None	8.515 ns	B[2]	Sum[3]		
44	N/A	None	8.429 ns	B[0]	Cout		
45	N/A	None	8.299 ns	B[3]	Sum[4]		
..							

Registered Performance		tpd	tsu	tco	th	Custom Delays	
	Slack	Required P2P Time	Actual P2P Time	From	To		
46	N/A	None	8.271 ns	B[2]	Sum[7]		
47	N/A	None	8.213 ns	B[1]	Cout		
48	N/A	None	8.206 ns	B[3]	Sum[7]		
49	N/A	None	8.204 ns	A[1]	Cout		
50	N/A	None	8.154 ns	B[2]	Sum[5]		
51	N/A	None	8.146 ns	B[4]	Sum[7]		
52	N/A	None	8.090 ns	A[0]	Cout		
53	N/A	None	8.089 ns	B[3]	Sum[5]		
54	N/A	None	8.059 ns	A[2]	Cout		
55	N/A	None	7.717 ns	B[4]	Sum[5]		
56	N/A	None	7.640 ns	A[4]	Sum[6]		
57	N/A	None	7.597 ns	B[2]	Cout		
58	N/A	None	7.578 ns	A[3]	Sum[6]		
59	N/A	None	7.532 ns	B[3]	Cout		
60	N/A	None	7.472 ns	B[4]	Cout		
..							

	Slack	Required P2P Time	Actual P2P Time	From	To	
59	N/A	None	7.532 ns	B[3]	Cout	
60	N/A	None	7.472 ns	B[4]	Cout	
61	N/A	None	7.071 ns	A[5]	Sum[6]	
62	N/A	None	7.066 ns	A[3]	Sum[4]	
63	N/A	None	7.034 ns	A[4]	Sum[7]	
64	N/A	None	6.972 ns	A[3]	Sum[7]	
65	N/A	None	6.858 ns	A[7]	Cout	
66	N/A	None	6.855 ns	A[3]	Sum[5]	
67	N/A	None	6.781 ns	A[5]	Sum[7]	
68	N/A	None	6.602 ns	A[4]	Sum[5]	
69	N/A	None	6.407 ns	A[6]	Sum[7]	
70	N/A	None	6.360 ns	A[4]	Cout	
71	N/A	None	6.298 ns	A[3]	Cout	
72	N/A	None	6.107 ns	A[5]	Cout	
73	N/A	None	6.045 ns	A[6]	Cout	

Επιπρόσθετα δημιουργήσαμε και ένα αρχείο εξομοίωσης για να δούμε τις καθυστερήσεις πάνω στις κυματομορφές. Θέσαμε ως περίοδο αλλαγής την μονάδα στην είσοδο A ανά 10ns και στην B αντίστοιχα 20ns. Στην είσοδο Cin θέσαμε περίοδο αλλαγής 20ns και παρακάτω έχουμε το διάγραμμα:



Κατά την ανάλυση και των δύο τρόπων περιγραφής, ο δεύτερος είναι πιο γρήγορος, καθώς δεν χρειάζεται να δημιουργήσουμε component ενός απλού πλήρη αθροιστή μέσα στην αρχιτεκτονική του πλήρη αθροιστή των 8 bit και να κάνουμε χρήση την εντολής port map για κάθε έξοδο. Συνεπώς ο 2^{oc} τρόπος είναι πιο βέλτιστος.

Μέρος 2^ο: Ακολουθιακά Συνδιαστικά Κυκλώματα

Ερώτημα 1^ο:

Σε αυτό το ερώτημα θα σχεδιάσουμε τη βασική μονάδα μνήμης(δηλαδή ένα D flip-flop) και ένα latch, το οποίο είναι το βασικό στοιχείο του flip-flop.

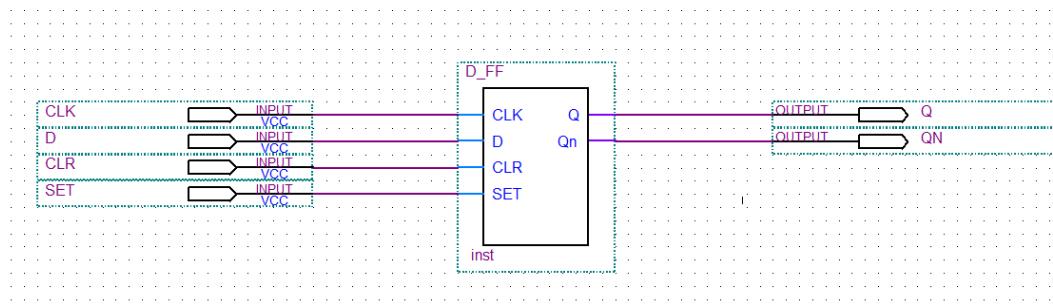
Παρακάτω φαίνεται ο κώδικας σε VHDL για την δημιουργεία του D flip-flop:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity D_FF is
5   port( CLK,D,CLR,SET: in std_logic;
6         Q,Qn: out std_logic);
7 end D_FF;
8
9 architecture RTL of D_FF is
10    signal DFF: std_logic;
11 begin
12   seq0: process(CLK,CLR,SET)
13   begin
14     if(CLR='1') then DFF<='0';
15     elsif(SET='1') then DFF<='1';
16     elsif(CLK'event and CLK = '1') then DFF<=D;
17     end if;
18   end process;
19   Q <=DFF; Qn <= not DFF;
20 end RTL;

```

Στη συνέχεια δημιουργούμε ένα το block, το οποίο φαίνεται παρακάτω:



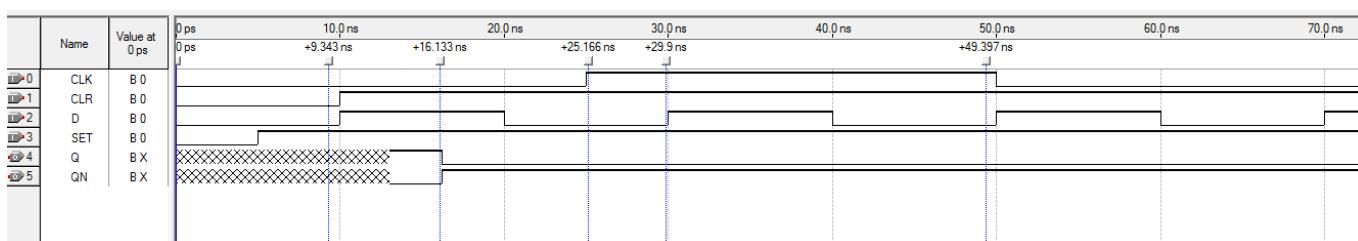
Κατόπιν θέσαμε τα pins ως εξής:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	CLK	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
2	CLR	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
3	D	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
4	Q	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
5	QN	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
6	SET	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
7	<new node>						

Επιπλέον, εκτελέσαμε στατική χρονική ανάλυση για να δουμε την μέγιστη καθυστέρηση μετάδοσης από την είσοδο στις εξόδους:

		Registered Performance	tpd	tsu	tco	th	Custom Delays
		Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None		8.060 ns	SET	QN	
2	N/A	None		8.060 ns	SET	Q	
3	N/A	None		6.246 ns	CLR	QN	
4	N/A	None		6.246 ns	CLR	Q	

Επιπρόσθετα, δημιουργήσαμε αρχείο κυματομορφών στο οποίο φαίνονται οι παραπάνω καθυστερήσεις. Θέσαμε ως περίοδο αλλαγής 10ns την τιμή 1 στην είσοδο D, η είσοδος CLR παίρνει την τιμή 0 στο διάστημα [0ps,10ns] και κατόπιν την τιμή 1. Επιπλέον, η είσοδος SET παίρνει την τιμή 0 στο διάστημα [0ps,5ns] και έπειτα την τιμή 1. Τέλος, το ρολόι θα έχει περίοδο αλλαγής 50ns. Παρακάτω φαίνεται η χρονική εξομοίωση:



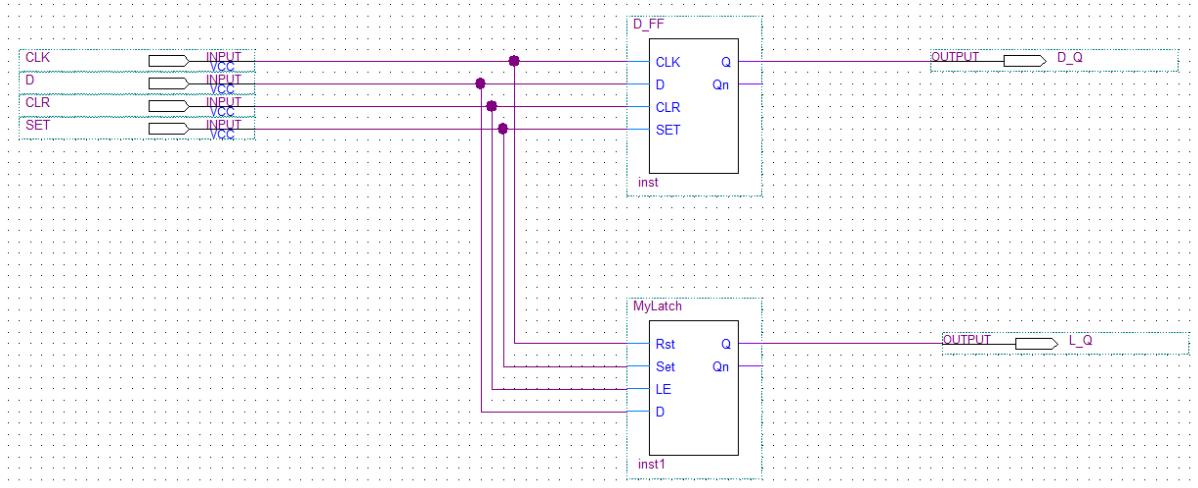
Στη συνέχεια σχεδιάσαμε το latch, του οποίου ο κώδικας φαίνεται παρακάτω:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity MyLatch is
5   port( Rst,Set,LE,D:in std_logic;
6         Q,Qn: out std_logic);
7 end MyLatch;
8
9 architecture RTL of MyLatch is
10    signal FF: std_logic;
11 begin
12   seq0: process(Rst,Set,D,LE)
13   begin
14     if Rst='1' then FF<='0';
15     elsif Set='1' then FF<='1';
16     elsif LE='1' then FF<=D;
17     end if;
18   end process;
19   Q<=FF;
20   Qn<= not FF;
21 end RTL;

```

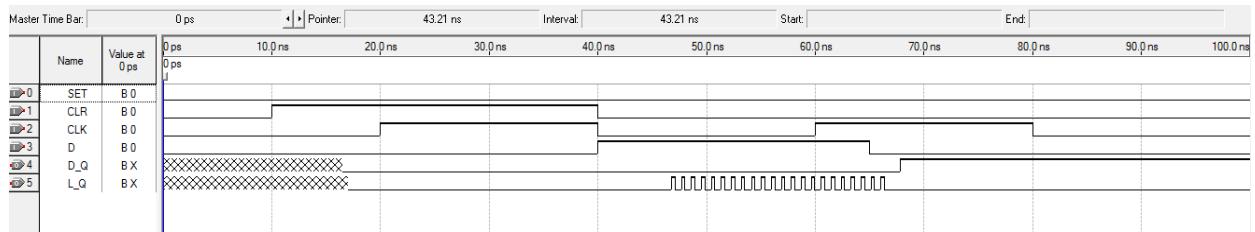
Στη συνέχεια δημιουργούμε ένα νέο block για το latch. Παρακάτω φαίνεται το σηματικό του κυκλώματος:



Κατόπιν εισάγουμε τα pins εισόδου και εξόδου:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
1	CLK	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)		
2	CLR	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
3	D	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
4	D_Q	Output	PIN_AE22	7	B7_N0	3.3-V LVTTL (default)		
5	L_Q	Output	PIN_AF22	7	B7_N0	3.3-V LVTTL (default)		
6	SET	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)		
7	<<new node>>							

Στη συνέχεια δημιουργούμε ένα αρχείο κυματομορφών. Θέσαμε την τιμή '1' στο σήμα CLR στο διάστημα [10ns,40ns] και στο ρολόι CLK θέσαμε περίοδο αλλαγής 40ns. Επιπλέον το σήμα SET είναι μόνιμα στην τιμή '0' και το σήμα D παίρνει την τιμή '1' στο διάστημα [40ns,65ns]. Σε αυτό το διάστημα παρατηρούμε πως η έξοδος του flip-flop αλλάζει τιμή στην επόμενη θετική ακμή του ρολογιού μέχρι τα 100ns. Ωστόσο, η έξοδος του latch αλλάζει σε καθ' όλη την διάρκεια της θετικής ακμής του ρολογιού. Παρακάτω φαίνεται η εξομοίωση:



Ερώτημα – 2^ο:

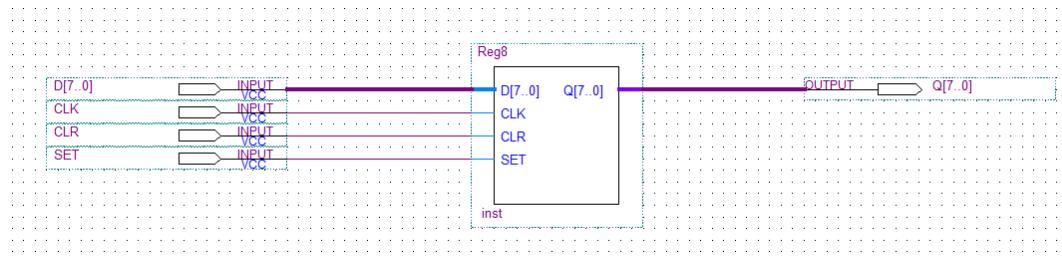
Σε αυτό το ερώτημα σχεδιάσαμε έναν καταχωρητή των 8 bit. Παρακάτω φαίνεται ο κώδικας σε VHDL:

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity Reg8 is
5    port(D: in std_logic_vector (7 downto 0);
6          CLK,CLR,SET: in std_logic;
7          Q: out std_logic_vector (7 downto 0));
8  end Reg8;
9
10 architecture RTL of Reg8 is
11   signal STORE: std_logic_vector (7 downto 0);
12 begin
13   seq0:process(CLK,CLR,SET)
14 begin
15     if (CLR='1') then STORE<=(7 downto 0 => '0');
16     elsif (SET = '1') then STORE<=(7 downto 0 => '1');
17     elsif (CLK'event and CLK='1') then
18       STORE<=D;
19     end if;
20   end process;
21   Q<=STORE;
22 end RTL;

```

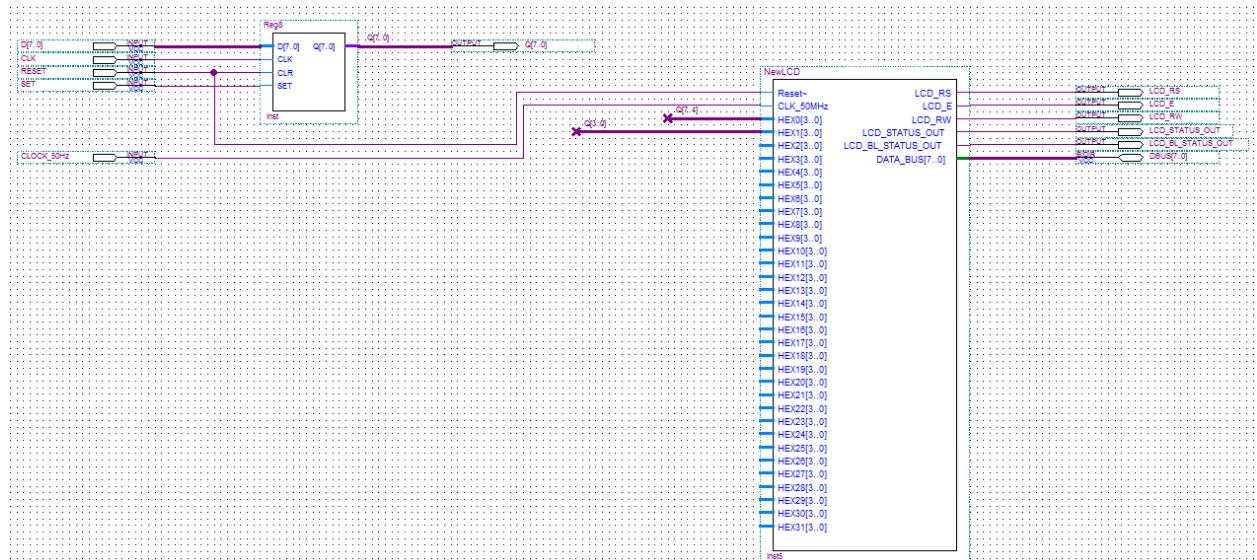
Δημιουργήσαμε ένα block του παραπάνω καταχωρητή και έχουμε το εξής σχέδιο:



Όπου τα pins είναι τα εξής:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	CLK	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
2	CLR	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
3	D[7]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)	
4	D[6]	Input	PIN_A13	4	B4_N1	3.3-V LVTTL (default)	
5	D[5]	Input	PIN_B13	4	B4_N1	3.3-V LVTTL (default)	
6	D[4]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)	
7	D[3]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)	
8	D[2]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)	
9	D[1]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)	
10	D[0]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
11	Q[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)	
12	Q[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)	
13	Q[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)	
14	Q[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)	
15	Q[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)	
16	Q[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)	
17	Q[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
18	Q[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
19	SET	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
20	<<new node>>						

Καθώς μας ζητάει να βγάλουμε τα αποτελέσματα στο LCD, επεκτείναμε το παραπάνω κύκλωμα στην εξής μορφή:



και τα pins είναι τα εξής:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
1	CLK	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)		
2	CLOCK_50Hz	Input	PIN_N2	2	B2_N1	3.3-V LVTTL (default)		
3	D[7]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)		
4	D[6]	Input	PIN_A13	4	B4_N1	3.3-V LVTTL (default)		
5	D[5]	Input	PIN_B13	4	B4_N1	3.3-V LVTTL (default)		
6	D[4]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)		
7	D[3]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)		
8	D[2]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)		
9	D[1]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)		
10	D[0]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)		
11	DBUS[7]	Bidir	PIN_H3	2	B2_N1	3.3-V LVTTL (default)		
12	DBUS[6]	Bidir	PIN_H4	2	B2_N1	3.3-V LVTTL (default)		
13	DBUS[5]	Bidir	PIN_J3	2	B2_N1	3.3-V LVTTL (default)		
14	DBUS[4]	Bidir	PIN_J4	2	B2_N1	3.3-V LVTTL (default)		
15	DBUS[3]	Bidir	PIN_H2	2	B2_N1	3.3-V LVTTL (default)		
16	DBUS[2]	Bidir	PIN_H1	2	B2_N1	3.3-V LVTTL (default)		
17	DBUS[1]	Bidir	PIN_J2	2	B2_N1	3.3-V LVTTL (default)		
18	DBUS[0]	Bidir	PIN_J1	2	B2_N1	3.3-V LVTTL (default)		
19	LCD_BL_STATUS_OUT	Output	PIN_K2	2	B2_N1	3.3-V LVTTL (default)		
20	LCD_E	Output	PIN_K3	2	B2_N1	3.3-V LVTTL (default)		
21	LCD_RS	Output	PIN_K1	2	B2_N1	3.3-V LVTTL (default)		
22	LCD_RW	Output	PIN_K4	2	B2_N1	3.3-V LVTTL (default)		
23	LCD_STATUS_OUT	Output	PIN_L4	2	B2_N1	3.3-V LVTTL (default)		
24	Q[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)		
25	Q[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)		
26	Q[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)		
27	Q[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)		
28	Q[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)		
29	Q[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)		
30	Q[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)		
31	Q[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
32	RESET	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
33	SET	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
34	<<new node>>							

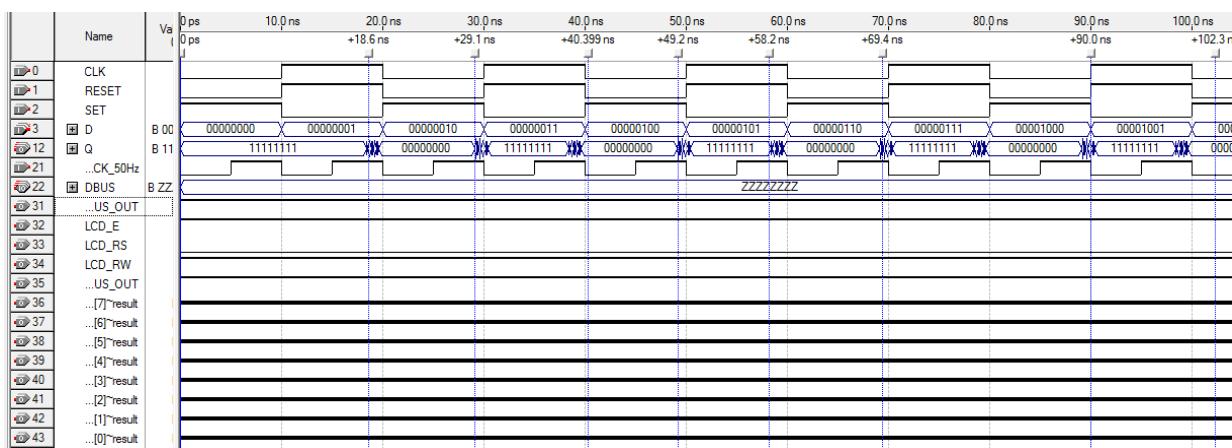
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
13	DBUS[5]	Bidir	PIN_J3	2	B2_N1	3.3-V LVTTL (default)		
14	DBUS[4]	Bidir	PIN_J4	2	B2_N1	3.3-V LVTTL (default)		
15	DBUS[3]	Bidir	PIN_H2	2	B2_N1	3.3-V LVTTL (default)		
16	DBUS[2]	Bidir	PIN_H1	2	B2_N1	3.3-V LVTTL (default)		
17	DBUS[1]	Bidir	PIN_J2	2	B2_N1	3.3-V LVTTL (default)		
18	DBUS[0]	Bidir	PIN_J1	2	B2_N1	3.3-V LVTTL (default)		
19	LCD_BL_STATUS_OUT	Output	PIN_K2	2	B2_N1	3.3-V LVTTL (default)		
20	LCD_E	Output	PIN_K3	2	B2_N1	3.3-V LVTTL (default)		
21	LCD_RS	Output	PIN_K1	2	B2_N1	3.3-V LVTTL (default)		
22	LCD_RW	Output	PIN_K4	2	B2_N1	3.3-V LVTTL (default)		
23	LCD_STATUS_OUT	Output	PIN_L4	2	B2_N1	3.3-V LVTTL (default)		
24	Q[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)		
25	Q[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)		
26	Q[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)		
27	Q[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)		
28	Q[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)		
29	Q[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)		
30	Q[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)		
31	Q[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
32	RESET	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
33	SET	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
34	<<new node>>							

Επιπλέον, κάναμε στατική χρονική ανάλυση και βρήκαμε τις εξής καθυστερύσεις:

	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	10.353 ns	RESET	Q[3]	
2	N/A	None	10.215 ns	RESET	Q[5]	
3	N/A	None	10.204 ns	RESET	Q[2]	
4	N/A	None	10.022 ns	SET	Q[3]	
5	N/A	None	9.884 ns	SET	Q[5]	
6	N/A	None	9.873 ns	SET	Q[2]	
7	N/A	None	9.759 ns	RESET	Q[6]	
8	N/A	None	9.576 ns	RESET	Q[4]	
9	N/A	None	9.428 ns	SET	Q[6]	
10	N/A	None	9.376 ns	RESET	Q[1]	
11	N/A	None	9.349 ns	RESET	Q[7]	
12	N/A	None	9.245 ns	SET	Q[4]	
13	N/A	None	9.178 ns	RESET	Q[0]	
14	N/A	None	9.045 ns	SET	Q[1]	
15	N/A	None	9.018 ns	SET	Q[7]	
..						
16	N/A	None	8.847 ns	SET	Q[0]	

Στη συνέχεια δημιουργήσαμε ένα αρχείο ROM.hex για να προβληθούν τα αποτελέσματα στο LCD. Παρακάτω φαίνεται το αρχείο:

Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών το οποίο φαίνεται παρακάτω:



Θέσαμε ως περίοδο αλλαγής της τιμής '1' τα 10ns στο σήμα εισόδου RESET και στο ρολόι CLK. Επίσης, στο CLK_50MHZ θέσαμε περίοδο 10ns και το σήμα εισόδου SET έχει περίοδο αλλαγής 10ns, ώστόσο ξεκινάει από την τιμή '1' και αλλάζει στην τιμή '0'. Επιπλέον, τα στοιχεία εισόδου του D παίρνουν την τιμή '1' ανά 10ns αντίστοιχα.

Ερώτημα – 3^ο:

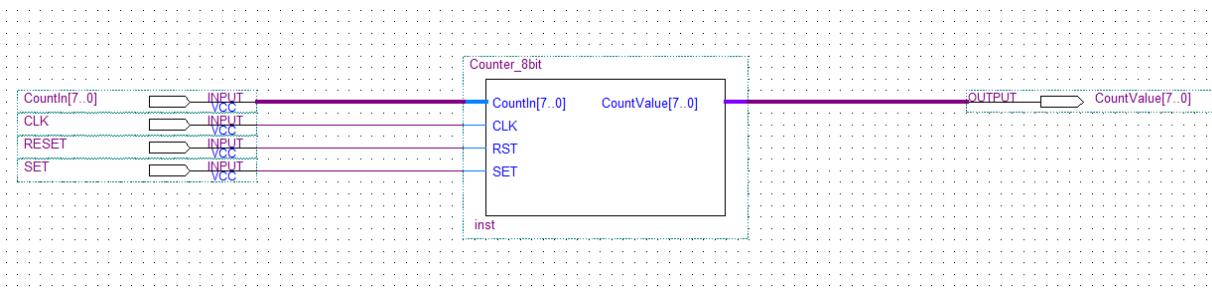
Σε αυτό το ερώτημα έχουμε να σχεδιάσουμε έναν μετρητή των 8 δυαδικών ψηφίων, ο οποίος έχει δυνατότητα μέτρησης από 0 έως 255. Παρακάτω φαίνεται ο κώδικας:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_unsigned.all;
4
5 entity Counter_8bit is
6     port(CountIn: in std_logic_vector(7 downto 0);
7           CLK,RST,SET: in std_logic;
8           CountValue: out std_logic_vector(7 downto 0));
9 end Counter_8bit;
10
11 architecture RTL of Counter_8bit is
12     signal count: std_logic_vector (7 downto 0);
13 begin
14     process(CLK,RST,SET)
15     begin
16         if (RST = '0') then
17             count<=(count'range =>'0');
18         elsif (CLK'event and CLK = '1') then
19             if(SET = '1') then
20                 count<=count+1;
21             end if;
22         end if;
23     end process;
24     CountValue <= count;
25 end RTL;

```

Στη συνέχεια δημιουργήσαμε ένα block, το οποίο φαίνεται παρακάτω:



Κατόπιν τοποθετήσαμε pins στις εισόδους και εξόδους:

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
1	CLK	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)		
2	Countin[7]	Input	PIN_N1	2	B2_N1	3.3-V LVTTL (default)		
3	Countin[6]	Input	PIN_A13	4	B4_N1	3.3-V LVTTL (default)		
4	Countin[5]	Input	PIN_B13	4	B4_N1	3.3-V LVTTL (default)		
5	Countin[4]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)		
6	Countin[3]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)		
7	Countin[2]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)		
8	Countin[1]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)		
9	Countin[0]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)		
10	CountValue[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)		
11	CountValue[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)		
12	CountValue[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)		
13	CountValue[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)		
14	CountValue[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)		
15	CountValue[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)		
16	CountValue[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)		
17	CountValue[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)		
18	RESET	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)		
19	SET	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)		
20	<<new node>>							

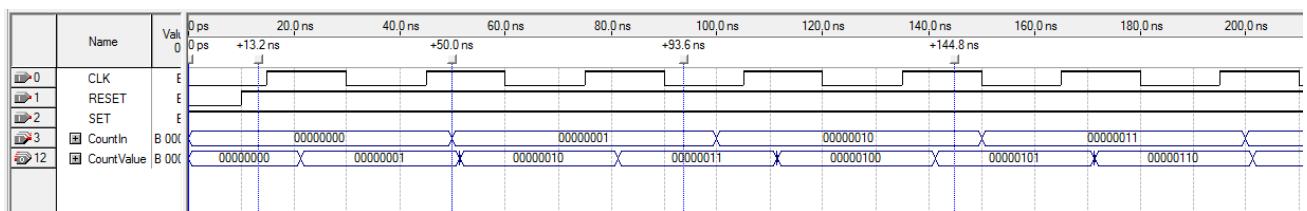
Στη συνέχεια εκτελέσαμε στατική χρονική ανάλυση, η οποία φαίνεται παρακάτω:

Registered Performance tpd tsu tco th Custom Delays							
	Slack	Required tsu	Actual tsu	From	To	To Clock	
1	N/A	None	0.355 ns	SET	Counter_8bit:inst count[7]	CLK	
2	N/A	None	0.355 ns	SET	Counter_8bit:inst count[6]	CLK	
3	N/A	None	0.355 ns	SET	Counter_8bit:inst count[5]	CLK	
4	N/A	None	0.355 ns	SET	Counter_8bit:inst count[4]	CLK	
5	N/A	None	0.355 ns	SET	Counter_8bit:inst count[3]	CLK	
6	N/A	None	0.355 ns	SET	Counter_8bit:inst count[2]	CLK	
7	N/A	None	0.355 ns	SET	Counter_8bit:inst count[1]	CLK	
8	N/A	None	-0.057 ns	SET	Counter_8bit:inst count[0]	CLK	

Registered Performance tpd tsu tco th Custom Delays							
	Slack	Required tco	Actual tco	From	To	From Clock	
1	N/A	None	6.597 ns	Counter_8bit:inst count[4]	CountValue[4]	CLK	
2	N/A	None	6.588 ns	Counter_8bit:inst count[7]	CountValue[7]	CLK	
3	N/A	None	6.584 ns	Counter_8bit:inst count[6]	CountValue[6]	CLK	
4	N/A	None	6.399 ns	Counter_8bit:inst count[1]	CountValue[1]	CLK	
5	N/A	None	6.387 ns	Counter_8bit:inst count[0]	CountValue[0]	CLK	
6	N/A	None	6.379 ns	Counter_8bit:inst count[5]	CountValue[5]	CLK	
7	N/A	None	6.361 ns	Counter_8bit:inst count[3]	CountValue[3]	CLK	
8	N/A	None	6.360 ns	Counter_8bit:inst count[2]	CountValue[2]	CLK	

	Minimum Slack	Required th	Actual th	From	To	To Clock	
1	N/A	None	0.287 ns	SET	Counter_8bit:inst count[0]	CLK	
2	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[7]	CLK	
3	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[6]	CLK	
4	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[5]	CLK	
5	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[4]	CLK	
6	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[3]	CLK	
7	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[2]	CLK	
8	N/A	None	-0.125 ns	SET	Counter_8bit:inst count[1]	CLK	

Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών, το οποίο φαίνεται παρακάτω:



Όπου θέσαμε στο ρολόι περίοδο αλλαγής 15ns, το σήμα RESET παίρνει την τιμή ‘1’ μετά από 10ns και το σήμα SET έχει πάντα την τιμή ‘1’.

5^η Εργαστηριακή Άσκηση: Σχεδίαση ενός Ακολουθιακού Πολλαπλασιαστή σε VHDL

Σε αυτή την άσκηση είχαμε να σχεδιάσουμε έναν ακολουθιακό πολλαπλασιαστή, ο οποίος απαρτίζεται από πέντε καταχωρητές, έναν αθροιστή και μία μονάδα ελέγχου. Το πρώτο κομμάτι σχεδίασης είναι ο καταχωρητής. Ωστόσο, έχουμε πέντε διαφορετικούς καταχωρητές. Για αυτό τον λόγο έχουμε την γενική οντότητα του καταχώτητη, η οποία έχει όλες τις δυνατότητες που χρειαζόμαστε για τον καθένα ξεχωριστά. Παρακάτω φαίνεται ο κώδικας σε VHDL του καταχωρητή:

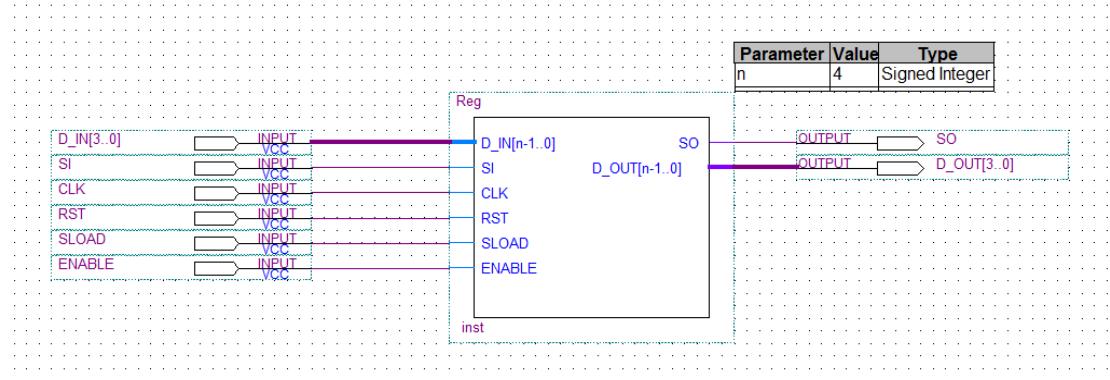
```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity Reg is
5      generic( n: integer:=4);
6      port(
7          D_IN: in std_logic_vector (n-1 downto 0);
8          SI,CLK,RST,SLOAD,ENABLE: in std_logic;
9          SO: out std_logic;
10         D_OUT: out std_logic_vector (n-1 downto 0));
11     end Reg;
12
13    architecture RTL of Reg is
14        signal F: std_logic_vector (n-1 downto 0);
15    begin
16        p0: process(RST,CLK)
17        begin
18            if (RST = '1') then F<=(n-1 downto 0 => '0');
19            elsif (CLK'event and CLK = '1') then
20                if (ENABLE = '1') then
21                    if (SLOAD = '0') then F<=D_IN;
22                    else F<=SI & F(n-1 downto 1);
23                    end if;
24                end if;
25            end if;
26        end process;
27        D_OUT <=F;
28        SO <=F(0);
29    end RTL;

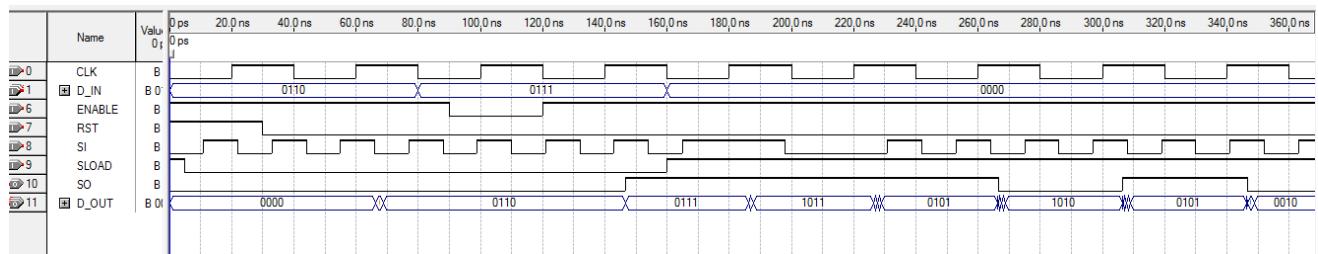
```

Στη συνέχεια, δημιουργήσαμε το block και το component της παραπάνω οντότητας.

Παρακάτω φαίνεται το κύκλωμα:



Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών στο οποίο φαίνονται όλες οι λειτουργίες του:



Όπου θέσαμε ως περίοδο ρολογιού 40ns, στην είσοδο επίτρεψης ENABLE θέσαμε την τιμή ‘1’ σε όλη την διάρκεια της εξομοίωσης εκτός από το διάστημα [90ns, 120ns] όπου θέσαμε την τιμή ‘0’, το σήμα RST έχει την τιμή ‘1’ μόνο για τα πρώτα 30ns και στην είσοδο D_IN θέσαμε την τιμή ‘0110’ για τα πρώτα 80ns, έπειτα θέσαμε την τιμή ‘0111’ για ακόμα 80ns και τέλος, θέσαμε την τιμή ‘0000’ για τον υπόλοιπο χρόνο της εξομοίωσης. Επίσης, το σήμα SLOAD έχει την τιμή ‘1’ για τα πρώτα 5ns, μετά μηδενίζεται και μετά ξανά παίρνει την τιμή ‘1’ από τα 160ns μέχρι να τελειώσει η εξομοίωση. Τέλος, η σειριακή είσοδος SI παίρνει την τιμή ‘1’ ανά 11ns, εκτός από το χρονικό διάστημα [165ns, 230ns] στο οποίο παίρνει την τιμή ‘1’ στο διάστημα [165ns, 198ns] και την τιμή ‘0’ στο διάστημα [198ns, 230ns].

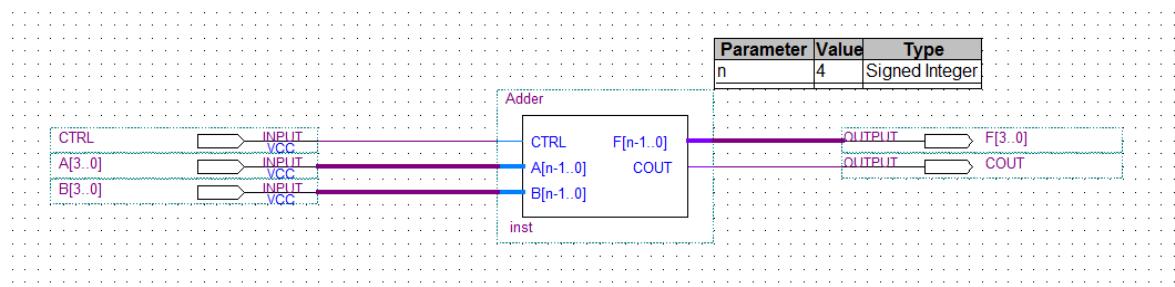
Η επόμενη οντότητα που είχαμε να σχεδιάσουμε ήταν ο αθροιστής. Παρακάτω φαίνεται ο κώδικας σε VHDL:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  ENTITY Adder IS
7      generic (n: integer :=4);
8      PORT(
9          CTRL : IN STD_LOGIC;
10         A,B : IN STD_LOGIC_VECTOR (n-1 downto 0);
11         F   : OUT STD_LOGIC_VECTOR (n-1 downto 0);
12         COUT : OUT STD_LOGIC
13     );
14 END Adder;
15
16 ARCHITECTURE RTL OF Adder IS
17     SIGNAL Interm : STD_LOGIC_VECTOR(n downto 0);
18 BEGIN
19
20     p0: process(A, B, CTRL)
21     begin
22         if (CTRL = '0') then Interm <='0' &B;
23         else Interm<=('0'& A)+('0' & B);
24         end if;
25     end process;
26     F<=Interm(n-1 downto 0);
27     COUT<=Interm(n);
28 END RTL;

```

Στη συνέχεια δημιουργήσαμε ένα block και ένα component του παραπάνω αθροιστή.
Παρακάτω φαίνεται το κύκλωμα:



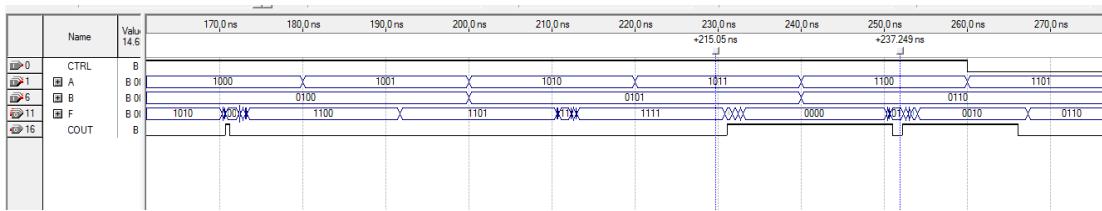
Επίσης, εκτελέσαμε στατική χρονική ανάλυση για να υπολογίσουμε την καθυστέρηση, η οποία φαίνεται παρακάτω:

	Slack	Required P2P Time	Actual P2P Time	From	To	
1	N/A	None	14.126 ns	B[2]	F[3]	
2	N/A	None	14.046 ns	A[2]	F[3]	
3	N/A	None	13.277 ns	B[2]	F[2]	
4	N/A	None	13.200 ns	A[2]	F[2]	
5	N/A	None	13.087 ns	A[1]	F[3]	
6	N/A	None	12.983 ns	A[0]	F[3]	
7	N/A	None	12.890 ns	B[0]	F[3]	
8	N/A	None	12.843 ns	B[1]	F[3]	
9	N/A	None	12.553 ns	A[1]	F[2]	
10	N/A	None	12.449 ns	A[0]	F[2]	
11	N/A	None	12.356 ns	B[0]	F[2]	
12	N/A	None	12.317 ns	A[3]	F[3]	
13	N/A	None	12.309 ns	B[1]	F[2]	
14	N/A	None	12.260 ns	B[2]	COUT	
15	N/A	None	12.180 ns	A[2]	COUT	
..						

	Slack	Required P2P Time	Actual P2P Time	From	To	
16	N/A	None	11.715 ns	A[0]	F[0]	
17	N/A	None	11.626 ns	B[0]	F[0]	
18	N/A	None	11.221 ns	A[1]	COUT	
19	N/A	None	11.117 ns	A[0]	COUT	
20	N/A	None	11.024 ns	B[0]	COUT	
21	N/A	None	10.977 ns	B[1]	COUT	
22	N/A	None	10.818 ns	A[0]	F[1]	
23	N/A	None	10.767 ns	A[3]	COUT	
24	N/A	None	10.725 ns	B[0]	F[1]	
25	N/A	None	10.607 ns	A[1]	F[1]	
26	N/A	None	10.362 ns	B[1]	F[1]	
27	N/A	None	8.607 ns	B[3]	F[3]	
28	N/A	None	7.776 ns	CTRL	F[3]	
29	N/A	None	7.348 ns	CTRL	F[0]	
30	N/A	None	7.274 ns	CTRL	F[2]	
..						

31	N/A	None	7.056 ns	B[3]	COUT	
32	N/A	None	6.153 ns	CTRL	COUT	
33	N/A	None	5.926 ns	CTRL	F[1]	

Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών, το οποίο φαίνεται παρακάτω:



Όπου στο σήμα CTRL θέσαμε την τιμή '1' μέχρι τα 260ns και ύστερα παίρνει την τιμή '0', στη συνέχεια θέσαμε ως περίοδο αλλαγής στο A και στο B, 20ns και 40ns αντίστοιχα.

Η τελευταία οντότητα που δημιουργήσαμε ήταν η μονάδα ελέγχου, η οποία θα ελέγχει αν γίνει φόρτωση των καταχωριτών ή αν γίνει πολλαπλασιασμός των δεδομένων εισόδου. Παρακάτω φαίνεται ο κώδικας:

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity CtrlLogic is
6    generic (n: integer :=4);
7    port(
8      Rst,CLK: in std_logic;
9      SL_A,SL_B,SL_H,SL_L,SL_C: out std_logic;
10     EN_A,EN_B,EN_H,EN_L,EN_C: out std_logic);
11   end CtrlLogic;
12
13  architecture RTL of CtrlLogic is
14    type state_type is (LOAD,MULT,HOLD);
15    type mult_type is (SHIFT,ADD);
16    signal state: state_type;
17    signal m_state: mult_type;
18    signal count : std_logic_vector(n-1 downto 0);
19  begin
20    p0: process(Rst,CLK)
21    begin
22      if (Rst = '1') then
23        state<=LOAD;
24        count<=(n-1 downto 0 =>'0');
25      elsif (CLK'event and CLK ='1') then
26        case state is
27          when LOAD => if (conv_integer(count) < 2*n-1) then count <= count + 1;
28          else count <= (n-1 downto 0 => '0');
29          state <= MULT;
30          m_state <= ADD;
31          end if;
32          when MULT => if (m_state = ADD) then m_state <= SHIFT;
33          elsif (m_state = SHIFT) then
34            m_state <= ADD;
35            if (conv_integer(count) < n-1) then count <= count + 1;
36            else state <= HOLD;
37            end if;
38          when HOLD => null;
39        end case;
40      end if;
41    end process;

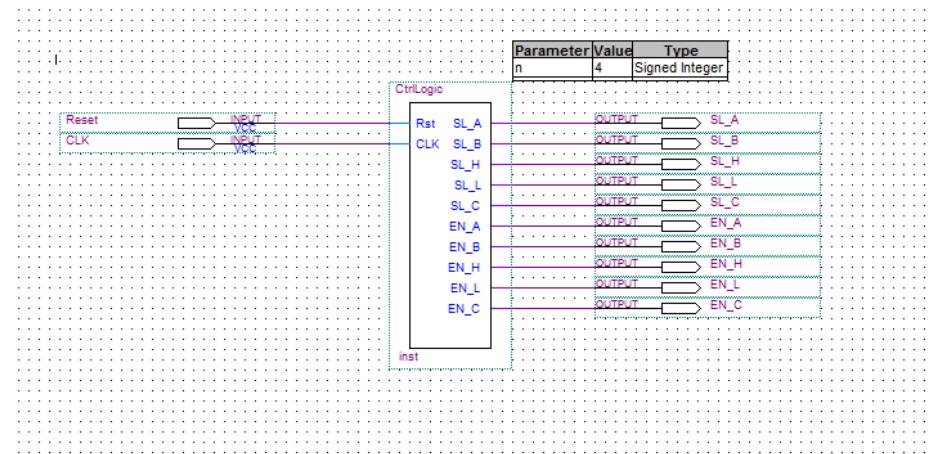
```

```

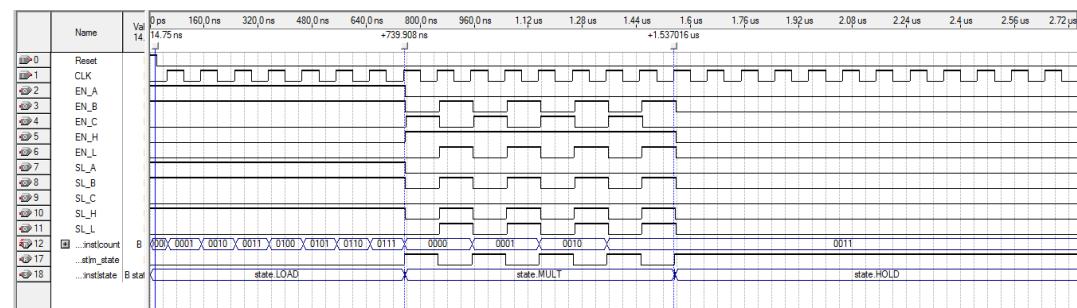
42      end process;
43
44      --Register A
45      EN_A <= '1' when state = LOAD else '0';
46      SL_A <= '1' when state = LOAD else '0';
47
48      --Register B
49      EN_B <= '1' when (state = LOAD or (state = MULTI and m_state = SHIFT)) else '0';
50      SL_B <= '1' when (state = LOAD or (state = MULTI and m_state = SHIFT)) else '0';
51
52      --Register H
53      EN_H <= '1' when state = MULT else '0';
54      SL_H <= '1' when m_state = SHIFT else '0';
55
56      --Register L
57      EN_L <= '1' when (state = MULT and m_state = SHIFT) else '0';
58      SL_L <= '1' when (state = MULT and m_state = SHIFT) else '0';
59
60      --Register C
61      EN_C <= '1' when (state = MULT and m_state = ADD) else '0';
62      SL_C <= '0';
63
64  end RTL;

```

Στη συνέχεια δημιουργήσαμε ένα block και VHDL component για το συνολικό κύκλωμα. Το σχηματικό της μονάδας είναι το εξής:



Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών, το οποίο φαίνεται παρακάτω:



Όπου το σήμα Reset έχει την τιμή '1' για τα πρώτα 20ns και μετά παίρνει την τιμή '0'. Επίσης, το ρολόι έχει περίοδο αλλαγής 50ns.

Τελικό βήμα ήταν να σχεδιάσουμε τον πολλαπλασιαστή χρησιμοποιώντας τις παραπάνω οντότητες ως components. Οι καταχωρητές A και B θα φορτώνονται σειριακά από την είσοδο SI. Στη συνέχεια, οι έξοδοι των καταχωρητών θα εισέρχονται στον αθροιστή, και το αποτέλεσμα θα αποθηκευθεί στους καταχωρητές H και L, όπου στον καταχωρητή H αποθηκεύονται τα ψηφία με την υψηλότερη σημαντικότητα, ενώ τα ψηφία χαμηλότερης σημαντικότητας αποθηκεύονται στον καταχωρητή L. Επίσης, το κρατούμενο εξόδου της πρόσθεσης cout θα αποθηκευτεί στον καταχωρητή C, το οποίο θα ελέγχει αν γίνεται η πρόσθεση των δεδομένων των καταχωρητών A και B. Τέλος, όλη η παραπάνω διεργασία θα ελέγχεται από την μονάδα ελέγχου, όπου θα αποφασίζει αν θα γίνει φόρτωση, πολλαπλασιασμός ή διατήρηση. Παρακάτω φαίνεται ο κώδικας του πολλαπλασιαστή:

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_arith.all;
4  use IEEE.std_logic_unsigned.all;
5
6  entity Multiplier is
7      generic (n: integer := 4);
8      port(
9          Rst,CLK,SI: in std_logic;
10         Low,High : out std_logic_vector(n-1 downto 0));
11 end Multiplier;
12
13 architecture RTL of Multiplier is
14     component CtrlLogic
15         generic (n: integer := 4);
16         port(
17             Rst,CLK: in std_logic;
18             SL_A,SL_B,SL_H,SL_L,SL_C: out std_logic;
19             EN_A,EN_B,EN_H,EN_L,EN_C: out std_logic);
20     end component;
21     component Adder
22         generic (n: integer :=4);
23         port(
24             CTRL : in std_logic;
25             A,B : in std_logic_vector (n-1 downto 0);
26             F : out std_logic_vector (n-1 downto 0);
27             COUT : out std_logic);
28     end component;
29     component Reg
30         generic( n: integer:=4);
31         port(
32             D_IN: in std_logic_vector (n-1 downto 0);
33             SI,CLK,RST,SLOAD,ENABLE: in std_logic;
34             SO: out std_logic;
35             D_OUT: out std_logic_vector (n-1 downto 0));
36     end component;
37     signal SL_A,EN_A,SO_A,SL_B,EN_B,SO_B,SL_C,EN_C,SO_C: std_logic;
38     signal SL_H,EN_H,SO_H,SL_L,EN_L,SO_L: std_logic;
39     signal A,B,F_ADD,H: std_logic_vector (n-1 downto 0);
40     signal C,COUT: std_logic_vector (0 downto 0);
41 begin

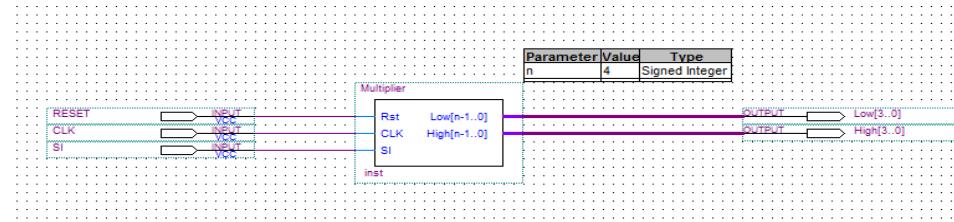
```

```

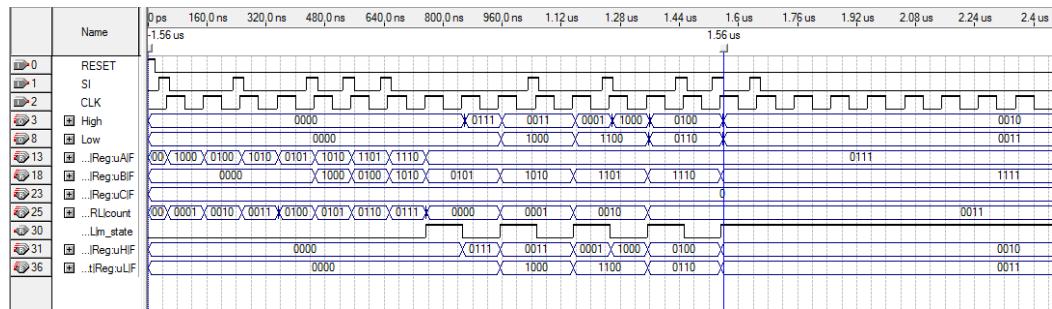
41 begin
42   uA: Reg generic map (n=>4)
43     port map (D_IN => (n-1 downto 0 => '0'), SI => SI, CLK => CLK, RST => RST, SLOAD => SL_A,
44       ENABLE => EN_A, SO => SO_A, D_OUT => A);
45   uB: Reg generic map (n=>4)
46     port map (D_IN => (n-1 downto 0 => '0'), SI => SO_A, CLK => CLK, RST => RST, SLOAD => SL_B,
47       ENABLE => EN_B, SO => SO_B, D_OUT => B);
48   uAdder: Adder generic map (n=>4)
49     port map (CTRL => B(0), A => A, B => H, F => F_ADD, COUT => COUT(0));
50   uC: Reg generic map (n=>1)
51     port map (D_IN => COUT, SI => '0', CLK => CLK, RST => RST, SLOAD => SL_C, ENABLE => EN_C,
52       SO => SO_C, D_OUT => C);
53   uH: Reg generic map (n=>4)
54     port map (D_IN => F_ADD, SI => SO_C, CLK => CLK, RST => RST, SLOAD => SL_H, ENABLE => EN_H,
55       SO => SO_H, D_OUT => H);
56   uL: Reg generic map (n=>4)
57     port map (D_IN => (n-1 downto 0 => '0'), SI => SO_H, CLK => CLK, RST => RST, SLOAD => SL_L,
58       ENABLE => EN_L, SO => SO_L, D_OUT => Low);
59   uCTRL: CtrlLogic generic map (n=>4)
60     port map (Rst => RST, CLK => CLK, SL_A => SL_A, SL_B => SL_B, SL_H => SL_H, SL_L => SL_L,
61       SL_C => SL_C, EN_A => EN_A, EN_B => EN_B, EN_H => EN_H, EN_L => EN_L, EN_C => EN_C);
62   High <= H;
63 end RTL;

```

Στη συνέχεια δημιουργήσαμε ένα block και ένα component το οποίο φαίνεται παρακάτω:



Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών, το οποίο φαίνεται παρακάτω:



όπου το σήμα εισόδου RESET έχει την τιμή '1' μόνο για τα πρώτα 20ns και το ρολόι έχει περίοδο αλλαγής 100ns. Η πράξη $7 \times 5 = 35$ φαίνεται παραπάνω όπου είναι και το τελικό αποτέλεσμα, δηλαδή $0111 \times 0101 = 00100011$. Η μέγιστη επιτρεπτή συχνότητα φαίνεται παρακάτω:

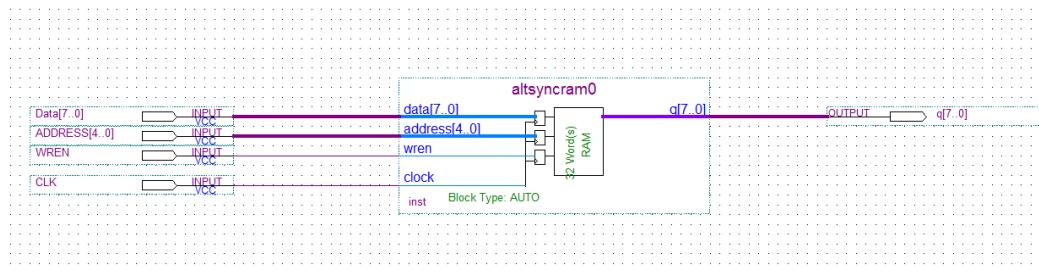
	Value
From	Multiplier:inst Reg:uA F[1]
To	Multiplier:inst Reg:uH F[3]
Clock period	2.415 ns
Frequency	414.08 MHz

Εργαστηριακή Άσκηση 6: Σχεδίαση με VHDL και μνήμες

Σε αυτή την άσκηση είχαμε να σχεδιάσουμε ένα ολοκληρωμένο σύστημα χρησιμοποιώντας την γλώσσα VHDL αλλά και μνήμες.

Ερώτημα – 1º:

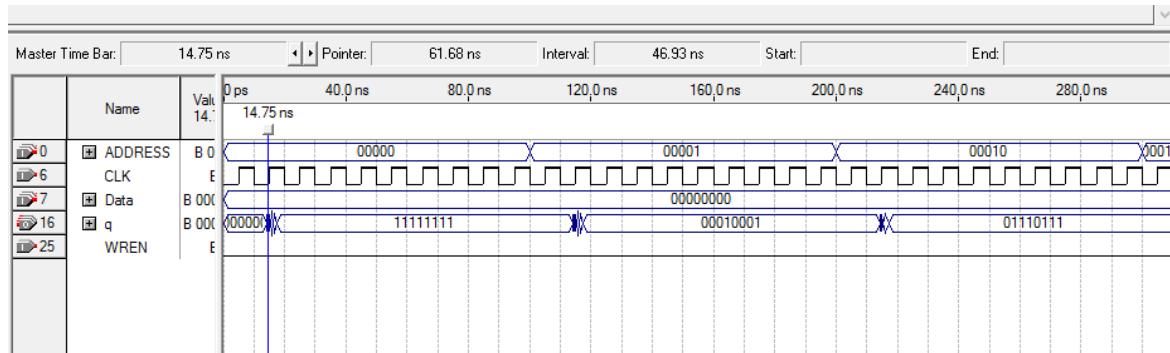
Είχαμε να σχεδιάσουμε το ένα block μνήμης, το οποίο έχει 32 λέξεις των 8 bit ως είσοδο και 8 bit έξοδο. Την παραπάνω μνήμη τη σχεδιάσαμε από το MegaWizard Plug-In Manager και χρησιμοποιήσαμε το σύμβολο Altsyncram. Παρακάτω φαίνεται το σύμβολο:



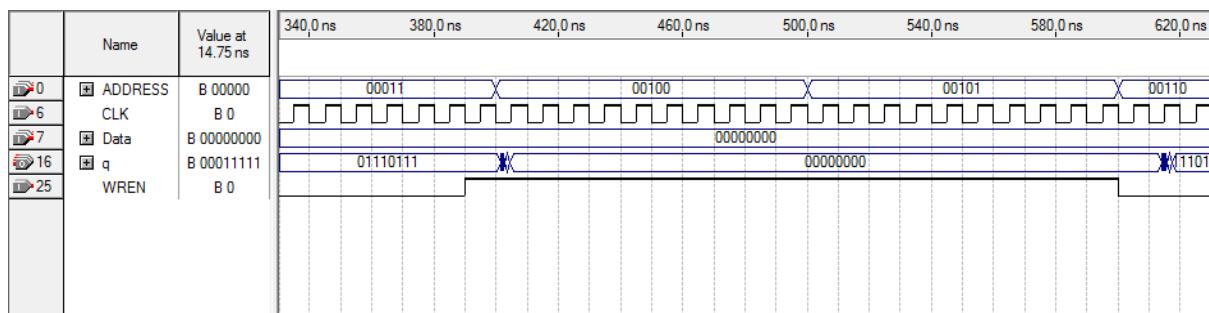
Οι είσοδοι Data[7..0] και address[4..0] τοποθετούν τα δεδομένα στην αντίστοιχη θέση μέσα στην μνήμη και η είσοδος WREN αποφασίζει αν θα κάνει εγγραφή ή ανάγνωση τα στοιχεία. Επίσης, για να λειτουργήσει η μνήμη πρέπει να φτιάξουμε ένα αρχείο .mif το οποίο είναι το αρχείο αρχικοποίησης της παραπάνω μνήμης. Παρακάτω φαίνεται αυτό το αρχείο:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	FF	11	77	77	77	77	77	77
8	77	77	77	77	77	77	77	77
16	77	77	77	77	77	77	77	77
24	77	77	77	77	77	77	77	77

Στη συνέχεια κάναμε χρονική εξομοίωση για να δούμε τις παραπάνω λειτουργίες της μνήμης. Στις παρακάτω χρονικές εξισώσεις θέσαμε ως περίοδο αλλαγής τα 5ns για το ρολόι και για το σήμα εισόδου address θέσαμε ως περίοδο αλλαγής τα 100ns. Παρακάτω φαίνονται οι δύο λειτουργίες:



Το σήμα εισόδου WREN έχει την τιμή '0', το οποίο σημαίνει πως η μνήμη κάνει ανάγνωση των στοιχείων και τα εμφανίζει στην έξοδο q.



Το σήμα εισόδου WREN έχει την τιμή '1', το οποίο σημαίνει πως η μνήμη βρίσκεται σε κύκλο εγγραφής των δεδομένων.

Ερώτημα – 2^o:

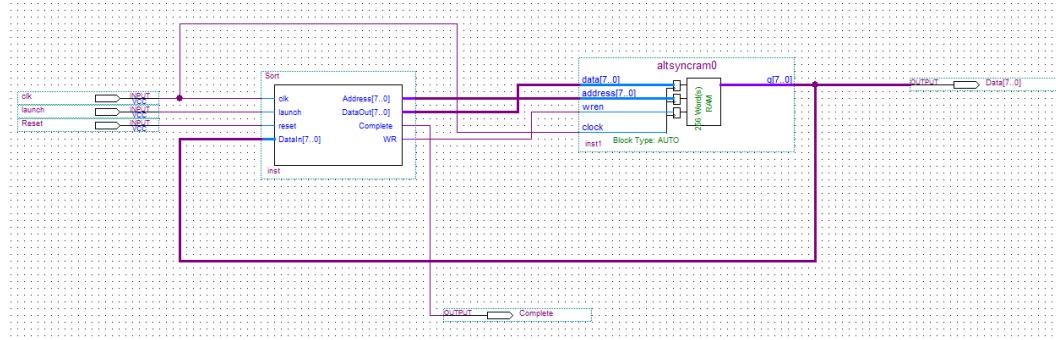
Σε αυτό το ερώτημα είχαμε να σχεδιάσουμε σε VHDL ένα σύστημα ταξινόμησης σε αύξουσα σειρά, το οποίο θα διατάσσει ένα μονοδιάστατο διάνυσμα το οποίο θα βρίσκεται αποθηκευμένο σε μνήμη RAM 256x8 χρησιμοποιώντας τον αλγόριθμο διάταξης φυσαλίδας (Bubble Sort). Παρακάτω φαίνεται ο κώδικας:

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity Sort is
6    port(
7      clk, launch, reset : in std_logic;
8      DataIn : in std_logic_vector (7 downto 0);
9      Address, DataOut : out std_logic_vector (7 downto 0);
10     Complete, WR : out std_logic
11   );
12 end Sort;
13
14 architecture RTL of Sort is
15 type state_type is (Waiting, SendAddrA_r, GetA, SendAddrB_r, GetB, Compare, SendAddrA_w, WriteA, SendAddrB_w, WriteB, CheckLoop, CheckFlag);
16 signal state: state_type := Waiting;
17 signal count : std_logic_vector (7 downto 0);
18 signal delay : std_logic;
19 signal dataA, dataB : std_logic_vector (7 downto 0);
20 signal Flag, Swap, CountEnd : std_logic;
21
22 BEGIN
23 process (clk, reset)
24 begin
25   if (reset = '1') then
26     state <= Waiting;
27   elsif (clk'event and clk = '1') then
28     case state is
29       when Waiting => if (launch = '1') then state <= SendAddrA_r; end if;
30       when SendAddrA_r => state <= GetA; delay <= '0';
31       when GetA => if (delay = '1') then state <= SendAddrB_r; else delay <= '1'; end if;
32       when SendAddrB_r => state <= GetB; delay <= '0';
33       when GetB => if (delay = '1') then state <= Compare; else delay <= '1'; end if;
34       when Compare => if (Swap = '1') then state <= SendAddrB_w; else state <= CheckLoop; end if;
35       when SendAddrB_w => state <= WriteB;
36       when WriteB => state <= SendAddrA_w;
37       when SendAddrA_w => state <= WriteA;
38       when WriteA => state <= CheckLoop;
39       when CheckLoop => if (CountEnd = '1') then state <= CheckFlag; else state <= SendAddrA_r; end if;
40       when CheckFlag => if (Flag = '0') then state <= Waiting; else state <= SendAddrA_r; end if;
41     end case;
42   end if;
43 end process;
44
45 process (clk)
46 begin
47   if (clk'event and clk = '1') then
48     case state is
49       when Waiting => count <= "00000000"; Flag <= '0';
50       when SendAddrA_r => Address <= count;
51       when GetA => if (delay = '1') then dataA <= DataIn; count <= count+1; end if;
52       when SendAddrB_r => Address <= count;
53       when GetB => if (delay = '1') then dataB <= DataIn; end if;
54       when Compare => if (Swap = '1') then count <= count-1; Flag <= '1'; end if;
55       when SendAddrB_w => Address <= count; DataOut <= dataB;
56       when WriteB => count <= count+1;
57       when SendAddrA_w => Address <= count; DataOut <= dataA;
58       when WriteA => null;
59       when CheckLoop => null;
60       when CheckFlag => if (Flag = '1') then Flag <= '0'; count <= "00000000"; end if;
61     end case;
62   end if;
63 end process;
64
65 WR <= '1' when state = WriteB or state = WriteA else '0';
66 Swap <= '1' when dataA > dataB else '0';
67 CountEnd <= '1' when count = "00000111" else '0';
68 Complete <= '1' when state = Waiting else '0';
69 end RTL;

```

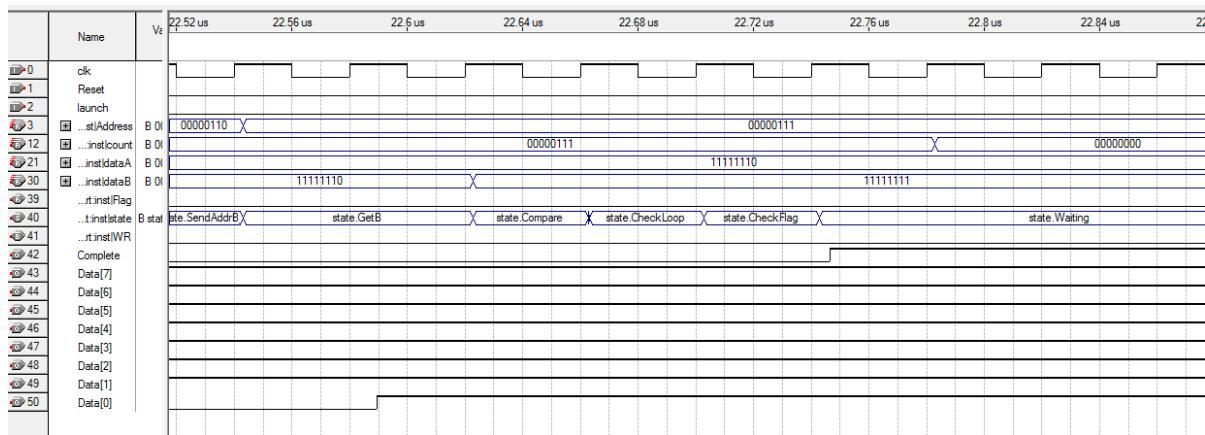
Στη συνέχεια δημιουργήσαμε το block της παραπάνω περιγραφής. Επίσης, στο νέο σχηματικό, προσθέσαμε ένα block μνήμης, το Altsyncram, το οποίο δέχεται 256 λέξεις. Παρακάτω φαίνεται το σχηματικό:



Επιπλέον δημιουργήσαμε ένα αρχείο .mif το οποίο φαίνεται παρακάτω:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	FF	FE	FD	FC	FB	FA	F9	F8
8	00	00	00	00	00	00	00	00
16	00	00	00	00	00	00	00	00
24	00	00	00	00	00	00	00	00
32	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00
48	00	00	00	00	00	00	00	00
56	00	00	00	00	00	00	00	00
64	00	00	00	00	00	00	00	00
72	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00
88	00	00	00	00	00	00	00	00
96	00	00	00	00	00	00	00	00
104	00	00	00	00	00	00	00	00
112	00	00	00	00	00	00	00	00
120	00	00	00	00	00	00	00	00

Τέλος, δημιουργήσαμε ένα αρχείο κυματομορφών, στο οποίο θέσαμε περίοδο αλλαγής 40ns στο ρολόι και το σήμα εισόδου Reset παίρνει την τιμή ‘1’ μόνο τα πρώτα 10ns. Επίσης, το σήμα εισόδου launch παίρνει την τιμή ‘1’ μόνο για λίγο, ώστε να ξεκινήσει η λειτουργία της ταξινόμησης. Επιπλέον η ταξινόμηση σταμάτησε στα 22,75ns. Παρακάτω φαίνεται η κυματομορφή:



Παρακάτω φαίνεται η ταξινόμηση της μνήμης:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	F8	F9	FA	FB	FC	FD	FE	FF
8	00	00	00	00	00	00	00	00
16	00	00	00	00	00	00	00	00
24	00	00	00	00	00	00	00	00
32	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00
48	00	00	00	00	00	00	00	00
56	00	00	00	00	00	00	00	00
64	00	00	00	00	00	00	00	00
72	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00
88	00	00	00	00	00	00	00	00
96	00	00	00	00	00	00	00	00
104	00	00	00	00	00	00	00	00
112	00	00	00	00	00	00	00	00
120	00	00	00	00	00	00	00	00
128	00	00	00	00	00	00	00	00
136	00	00	00	00	00	00	00	00
144	00	00	00	00	00	00	00	00
152	00	00	00	00	00	00	00	00