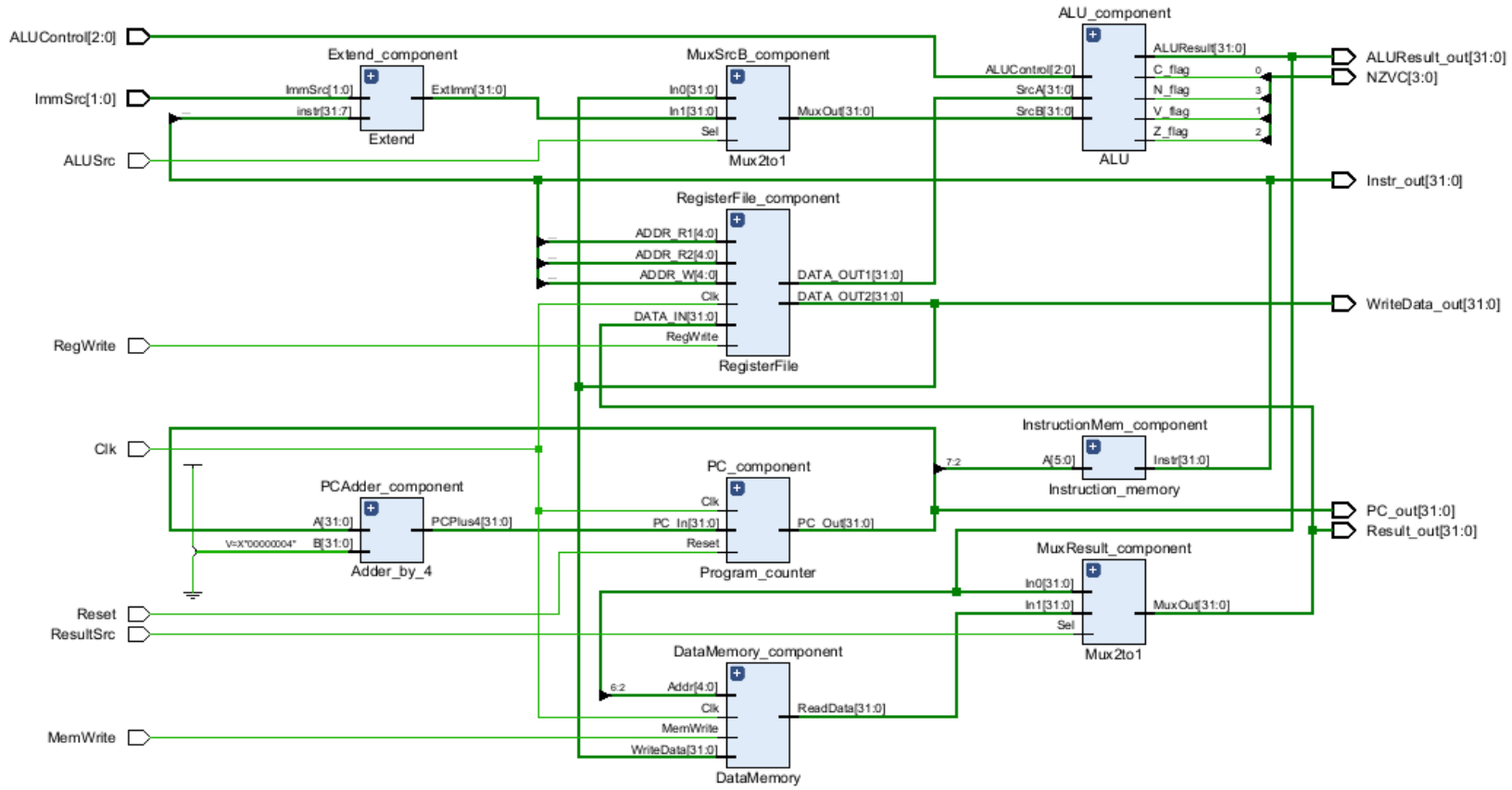
Αναφορά υλοποίησης **RISC-V** επεξεργαστή ενός κύκλου σε **VHDL**

7-1. **Περιγραφή των στοιχείων και της δομής του επεξεργαστή**

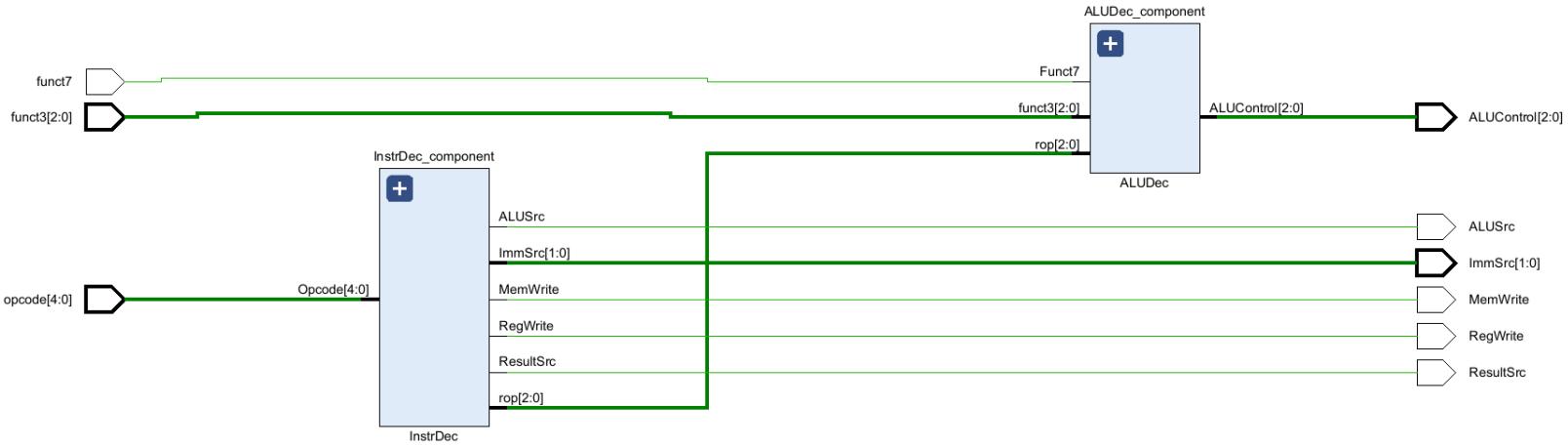
7-1.1 *RTL του* ***datapath***



Στο διάγραμμα RTL βλέπουμε τα component της διαδρομής δεδομένων. Αρχικά, υπάρχει ο καταχωρητής για τον **Program Counter.** Η **μνήμη ROM** για την αποθήκευση εντολών η οποία δέχεται την τιμή του program counter και βγάζει την αντίστοιχη εντολή η οποία πάει ως είσοδος στο register file ώστε να βρεθούν οι αντίστοιχοι καταχωρητές προέλευσης και προορισμού αλλά και στο Extend unit. Ο **αθροιστής** προσθέτει 4 στον program counter. Το **αρχείο καταχωρητών** δέχεται τα πεδία της εντολής που είπαμε πιο πάνω και έχει ως έξοδο της δύο θύρες για διάβασμα δεδομένων. Επίσης, έχει μία θύρα για γράψιμο, αν το επιτρέπει το σήμα RegWrite που έρχεται από το control unit. Στον καταχωρητή, γράφει ο επεξεργαστής σε περίπτωση που η εντολή δεν είναι τύπου Store. Έπειτα, έχουμε το **component επέκτασης προσήμου,** το οποίο κάνει τις αντίστοιχες επεκτάσεις για εντολές τύπου I και store. Ο extended τελεστής πηγαίνει ως είσοδος στο '1' του πολυπλέκτη για το SrcB της ALU, που ανάλογα με το σήμα ALUSrc, περνάει ως δεύτερη είσοδο στην ALU είτε την extended τιμή σε περίπτωση Immediate , Load και store είτε την τιμή του δεύτερου καταχωρητή εισόδου σε περίπτωση R εντολής. Το component της **ALU** δέχεται τις δύο εισόδους srcA, srcB, το σήμα ALUControl από το control unit, βάση του οποίου ξέρει τι πράξη θα κάνει, και έχει ως έξοδο το σήμα ALUResult που αποθηκεύεται στο αρχείο καταχωρητών σε περίπτωση εντολής τύπου R και I εντολής. Όταν η εντολή είναι LOAD το αποτέλεσμα της ALU είναι διεύθυνση και διαβάζει από την μνήμη την τιμή που τελικά θα γραφεί πίσω στο αρχείο καταχωρητών, ενώ στην περίπτωση της STORE το ALUResult είναι διεύθυνση και δεν αποθηκεύεται πουθενά απλά αποθηκεύεται σε αυτήν την διευθυνση το περιεχόμενο του RD2 του register file . Το σήμα SLTUorSLT απενεργοποιήθηκε, καθώς οι εντολές σύγκρισης δεν περιλαμβάνονται στο ζητούμενο instruction set όπως αναφέρεται στο τέλος της εκφώνησης πως τα αχρείαστα σήματα δεν πρέπει να υλοποιηθούν. Η **μνήμη RAM** που χρησιμοποιείται ως μνήμη δεδομένων γράφει δεδομένα στην διεύθυνση του ALUResult σε περίπτωση Store και έχει θύρα για δίαβασμα σε Load. Τέλος υπάρχει ένας **πολυπλέκτης** για να επιλεχθεί το από που θα γραφτούν τα αποτελέσματα πίσω στο αρχείο καταχωρητών βάση του σήματος ResultSrc (από RAM ή ALU), σε περίπτωση SW που δεν θέλουμε γράψιμο πίσω σε καταχωρητή έχει φροντίσει το Control Unit να έχει κάνει το RegWrite 0.

Τα component 2-5.2 και 2-5.3 ΔΕΝ υλοποιήθηκαν αφού αφορούν εντολές branch και όπως αναφέρεται στο τέλος της εκφώνησης παρόλο που αναφέρονται δεν πρέπει να υλοποιηθούν. Ο program counter είναι πάντα pc+4.

7-1.2 RTL διάγραμμα του **Control Unit**



Στο RTL διάγραμμα της Μονάδας Ελέγχου παρατηρούμε ότι υλοποιήθηκαν μόνο οι υπομονάδες **InstrDec** και **ALUDec**.

Η υπομονάδα **PCLogic** (και τα αντίστοιχα κυκλώματα 2-5.2, 2-5.3) **δεν υλοποιήθηκε**, καθώς το ζητούμενο σύνολο εντολών δεν περιλαμβάνει εντολές διακλάδωσης (Branch). Συνεπώς, η επιλογή της επόμενης διεύθυνσης δεν εξαρτάται από συνθήκες και το σήμα **PCSrc** θα είχε σταθερή τιμή '0' (επιλογή PC+4), καθιστώντας το κύκλωμα περιττό (όπως αναφέρεται και στο τέλος της εκφώνησης).

Η λειτουργία της Μονάδας Ελέγχου χωρίζεται σε :

1. **InstrDec:** Δέχεται το Opcode της εντολής και παράγει τα κύρια σήματα ελέγχου του datapath (ResultSrc, MemWrite, ALUSrc, ImmSrc, RegWrite). Παράλληλα, παράγει το εσωτερικό σήμα rop, το οποίο ενημερώνει τον ALUDec για τον τύπο της λειτουργίας.
2. **ALUDec (ALU Decoder):** Δέχεται το σήμα rop καθώς και τα πεδία funct3 και funct7 της εντολής. Βάσει αυτών, αποφασίζει την ακριβή πράξη που θα εκτελέσει η ALU και παράγει το σήμα **ALUControl**.

Τροποποιήσεις κύριου πίνακα αληθείας 4-2.1:

- Τα σήματα και η λογική για τις εντολές σύγκρισης SLTU, SLT αφαιρέθηκαν και το XOR γίνεται μόνο όταν ALUControl = 110

- Για τις εντολές **LW** και **SW**, ο ALUDec παράγει ALUControl = "000" (Πρόσθεση), ώστε η ALU να υπολογίσει τη διεύθυνση μνήμης (Base Register + Offset).

Πίνακας αληθείας για InstrDec

Op(6:2)	Rop(2:0)	ResultSrc	MemWrite	ALUsrc	ImmSrc(1:0)	RegWrite	Type
00000	000	1	0	1	00	1	*LW*
01100	010	0	0	0	00	1	*R*
00100	011	0	0	1	00	1	*I*
01000	001	0	1	1	01	0	*SW*

Πίνακας αληθείας για ALUDec

Rop(2:0)	Funct3(2:0)	Func7	ALUControl(2:0)	Εντολή
000	x	x	000	LW (ADD στην ALU
001	x	x	000	SW (ADD στην ALU)
010	000	0	000	ADD
010	000	1	001	SUB
010	100	x	110	XOR
010	110	x	101	OR
010	111	x	100	AND
011	000	x	000	ADDI
011	100	x	110	XORI
011	110	X	101	ORI
011	111	X	100	ANDI

Από τους πίνακες προκύπτει:

Εντολή LW:

- Στον **InstrDec**, το σήμα ALUSrc γίνεται 1 ώστε η ALU να αθροίσει τον καταχωρητή βάσης με το Immediate . Το ResultSrc είναι 1 για να οδηγηθούν τα δεδομένα από τη μνήμη (και όχι από την ALU) πίσω στον καταχωρητή προορισμού. Το RegWrite γίνεται 1.
- Στον **ALUDec**, παράγεται ALUControl = 000 (Πρόσθεση) για τον υπολογισμό της διεύθυνσης.

2. Εντολή SW:

- Στον **InstrDec**, το MemWrite είναι 1 για την εγγραφή στη μνήμη. Το RegWrite είναι 0 για να προστατευθούν τα περιεχόμενα των καταχωρητών. Το ALUSrc είναι 1 (υπολογισμός διεύθυνσης). Το immSrc γίνεται 01 για να γίνει σωστά το extension
- Στον **ALUDec**, παράγεται επίσης ALUControl = 000 (Πρόσθεση).

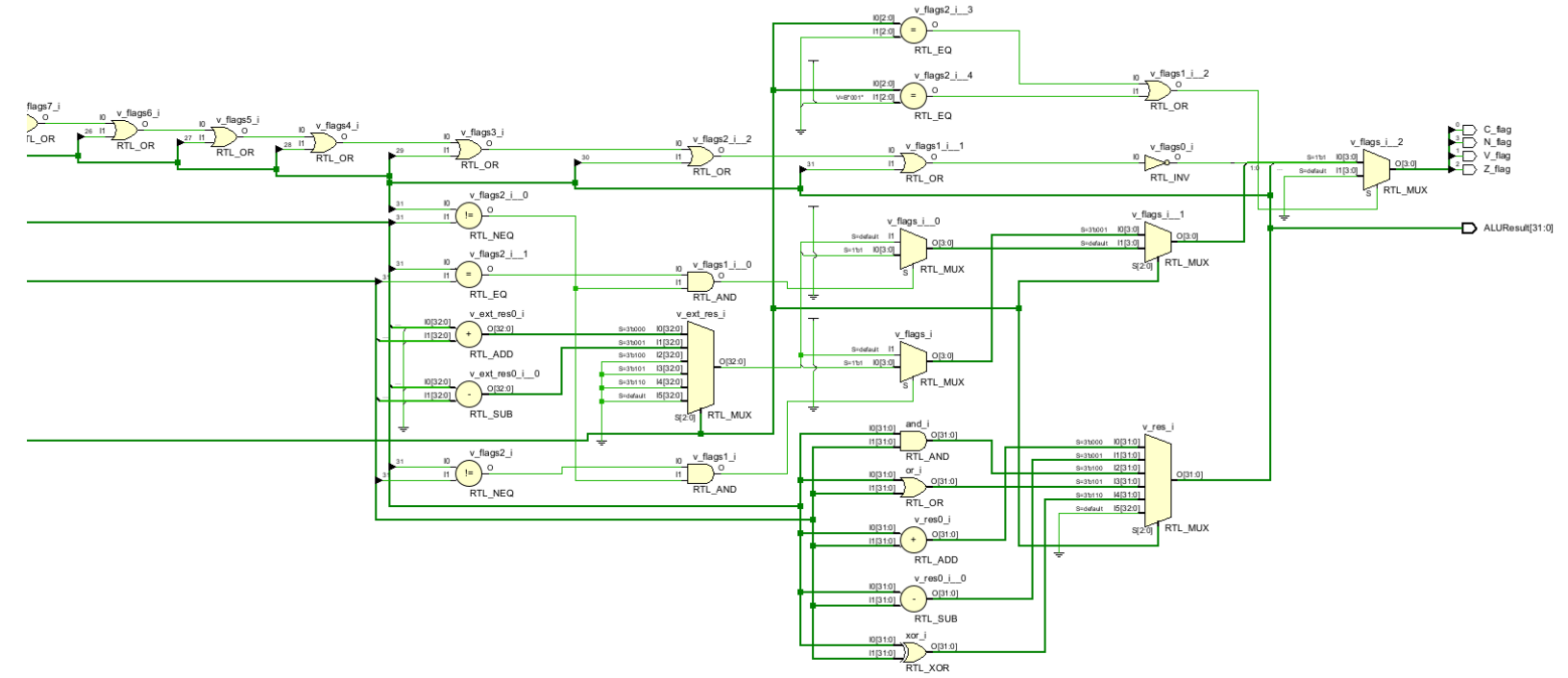
3. Εντολές R-Type:

- Στον **InstrDec**, το ALUSrc είναι '0' ώστε η ALU να επεξεργαστεί τους δύο καταχωρητές (RD1, RD2). Το ResultSrc είναι '0' (αποτέλεσμα ALU στον καταχωρητή). RegWrite = 1 για να γραφτεί το αποτέλεσμα πίσω στον καταχωρητή.
- Στον **ALUDec**, η εντολή διαφοροποιείται βάσει των πεδίων funct3 και funct7.

4. Εντολές I-Type:

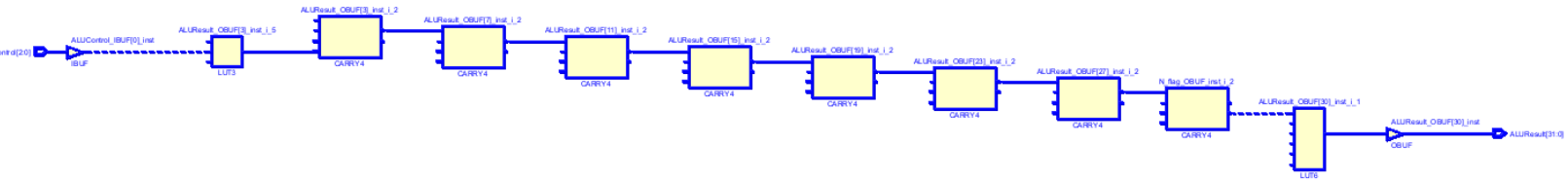
- Στον **InstrDec**, συμπεριφέρονται παρόμοια με τις R-Type ως προς την εγγραφή (RegWrite=1), αλλά το ALUSrc είναι '1' για χρήση του Immediate.
- Στον **ALUDec**, η πράξη καθορίζεται από το funct3.

7-1.3 RTL διάγραμμα της **ALU**



Το RTL διάγραμμα αποτελείται από πολλές πύλες nor συνδεδεμένες την μία μετα την άλλη γιατί για το flag Z (zero) αναφέρεται ρητά στην εκφώνηση πως πρέπει να υλοποιηθεί με πύλες nor, δοκίμασα να το κάνω και με ένα απλό if, στο elaborated design δημιουργούταν πολυπλέκτης, αλλά στο synthesized design οι πόροι ήταν ίδιοι οπότε δεν υπήρχε διαφορά. Επίσης, αποτελείται από όλα τα άλλα στοιχεία που είναι απαραίτητα για την ALU(αθροιστές, πολυπλέκτες, πύλες για τα αποτελέσματα). Δέχεται δύο τελεστές και παράγει το ALUResult καθώς και NZVC flags. Η λειτουργία της καθορίζεται από το ALUControl που έρχεται από το control unit

Καθυστέρηση διάδοσης Path 11: **15.786 ns** -> συχνότητα **63.347 MHz**

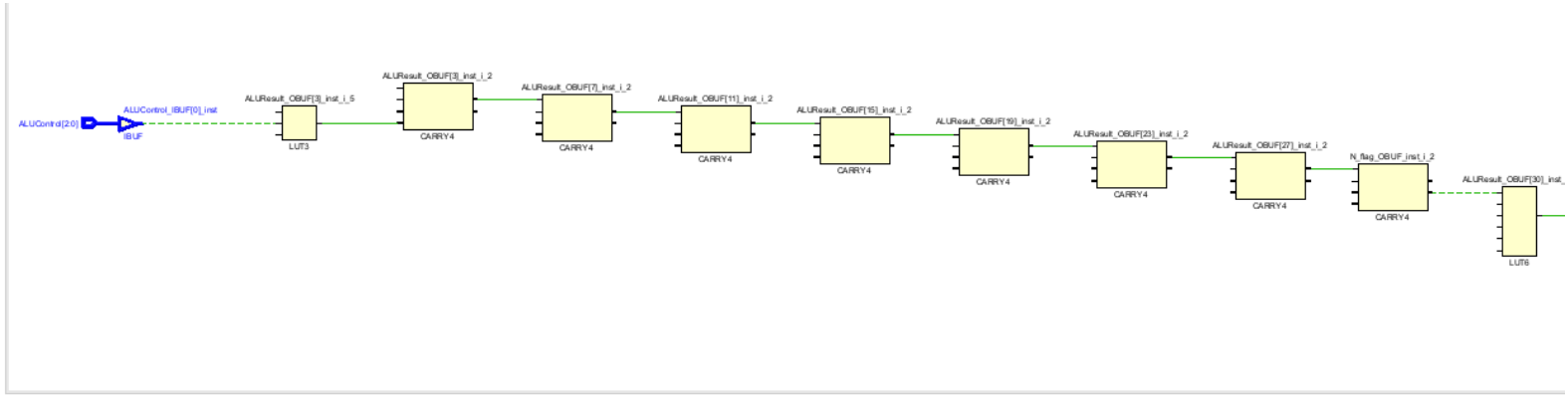


Design Runs | DRC | Power | Timing | Utilization

Timing Checks - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Destination Clock	Exception	Skew
Unconstrained Paths (1)																
(none) (10)																
Path 11	∞	12	3	98	ALUControl[0]	ALUResult[30]	15.786	5.555	10.231	35.2	64.8	∞	input port clock			
Path 12	∞	7	3	98	ALUControl[0]	ALUResult[9]	15.401	4.974	10.427	32.3	67.7	∞	input port clock			
Path 13	∞	11	3	98	ALUControl[0]	ALUResult[26]	15.373	5.384	9.989	35.0	65.0	∞	input port clock			
Path 14	∞	12	3	98	ALUControl[0]	ALUResult[28]	15.368	5.510	9.858	35.9	64.1	∞	input port clock			
Path 15	∞	9	3	98	ALUControl[0]	ALUResult[17]	15.356	5.243	10.112	34.1	65.9	∞	input port clock			
Path 16	∞	11	3	98	ALUControl[0]	ALUResult[24]	15.277	5.358	9.919	35.1	64.9	∞	input port clock			
Path 17	∞	10	3	98	ALUControl[0]	ALUResult[22]	15.264	5.282	9.983	34.6	65.4	∞	input port clock			
Path 18	∞	10	3	98	ALUControl[0]	ALUResult[20]	15.194	5.264	9.930	34.6	65.4	∞	input port clock			
Path 19	∞	6	3	98	ALUControl[0]	ALUResult[5]	15.147	4.852	10.295	32.0	68.0	∞	input port clock			
Path 20	∞	6	3	98	ALUControl[0]	ALUResult[7]	15.114	4.842	10.273	32.0	68.0	∞	input port clock			

Καθυστέρηση μόλυνσης Path 1: **2.079 ns**



Reports | Design Runs | DRC | Power | Timing | Utilization

Timing Checks - Hold

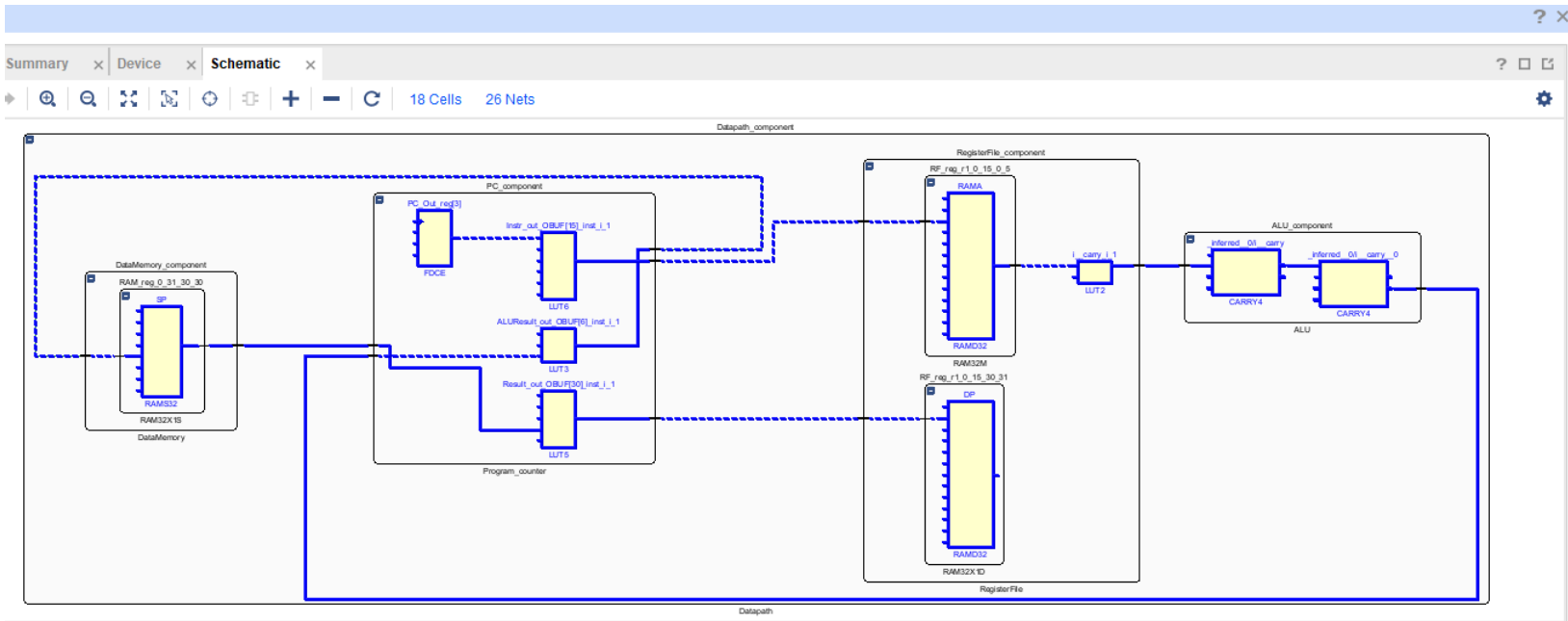
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Destinat
Unconstrained Paths (1)														
(none) (10)														
Path 1	∞	3	2	98	ALUControl[0]	V_flag	2.079	1.356	0.723	65.2	34.8	-∞	input port clock	
Path 2	∞	3	2	36	ALUControl[1]	N_flag	2.161	1.419	0.742	65.7	34.3	-∞	input port clock	
Path 3	∞	6	4	3	SrcB[18]	Z_flag	2.996	1.577	1.419	52.6	47.4	-∞	input port clock	
Path 4	∞	3	2	36	ALUControl[1]	C_flag	3.072	1.491	1.581	48.5	51.5	-∞	input port clock	
Path 5	∞	3	2	4	SrcB[31]	ALUResult[31]	3.339	1.418	1.921	42.5	57.5	-∞	input port clock	
Path 6	∞	3	2	3	SrcB[15]	ALUResult[15]	3.352	1.391	1.961	41.5	58.5	-∞	input port clock	
Path 7	∞	3	2	3	SrcB[29]	ALUResult[29]	3.361	1.455	1.906	43.3	56.7	-∞	input port clock	
Path 8	∞	3	2	3	SrcB[27]	ALUResult[27]	3.386	1.427	1.959	42.1	57.9	-∞	input port clock	
Path 9	∞	3	2	3	SrcB[21]	ALUResult[21]	3.490	1.432	2.058	41.0	59.0	-∞	input port clock	
Path 10	∞	3	2	3	SrcB[2]	ALUResult[2]	3.497	1.356	2.141	38.8	61.2	-∞	input port clock	

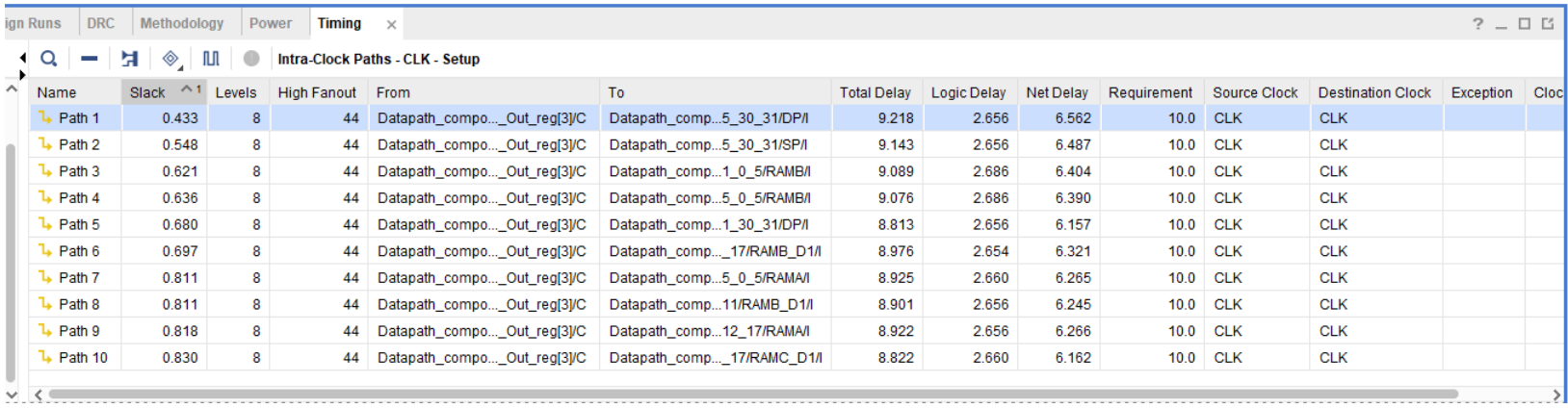
N ALU	103	32	103	103

Βλέποντας την καθυστέρηση διάδοσης της αυτόνομης ALU, βλέπουμε ότι είναι αρκετά μεγάλη και μάλιστα πιο μετά θα δούμε ότι η συχνότητα του Processor είναι μεγαλύτερη, παρόλο που προστίθεται πολύ λογική. Αυτό συμβαίνει αφού το vivado συνδέει τις θύρες σε φυσικά pins του FPGA άρα τα καλώδια έχουν μεγάλες αποστάσεις. Όμως, όταν η ALU ενσωματωθεί στο Datapath του επεξεργαστή, η επικοινωνία της με τα άλλα components (π.χ. Register File) γίνεται αποκλειστικά μέσω εσωτερικών συνδέσεων.

7-1.4

Η χειρότερη κρίσιμη διαδρομή είναι αυτή του Path 1 με **9.218 ns**

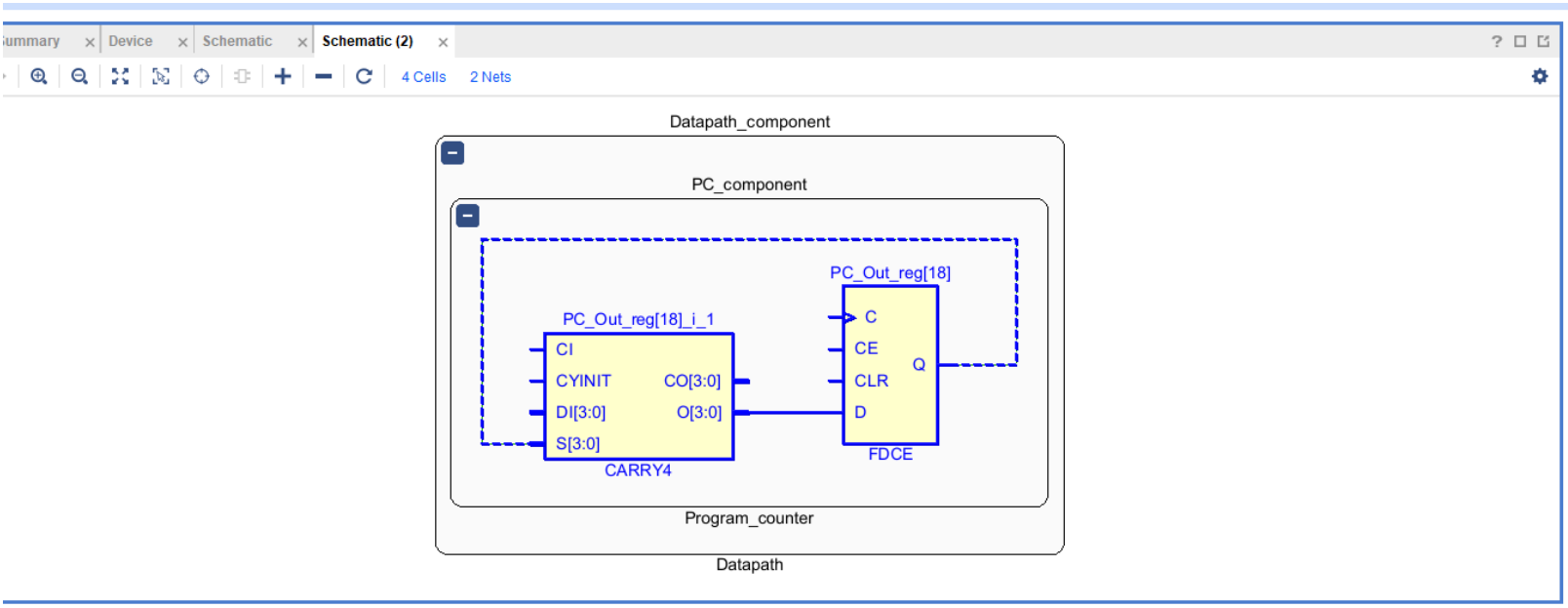




Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Cloc
Path 1	0.433	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...5_30_31/DP/I	9.218	2.656	6.562	10.0	CLK	CLK		
Path 2	0.548	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...5_30_31/SP/I	9.143	2.656	6.487	10.0	CLK	CLK		
Path 3	0.621	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...1_0_5/RAMB/I	9.089	2.686	6.404	10.0	CLK	CLK		
Path 4	0.636	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...5_0_5/RAMB/I	9.076	2.686	6.390	10.0	CLK	CLK		
Path 5	0.680	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...1_30_31/DP/I	8.813	2.656	6.157	10.0	CLK	CLK		
Path 6	0.697	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp..._17/RAMB_D1/I	8.976	2.654	6.321	10.0	CLK	CLK		
Path 7	0.811	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...5_0_5/RAMA/I	8.925	2.660	6.265	10.0	CLK	CLK		
Path 8	0.811	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...11/RAMB_D1/I	8.901	2.656	6.245	10.0	CLK	CLK		
Path 9	0.818	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp...12_17/RAMA/I	8.922	2.656	6.266	10.0	CLK	CLK		
Path 10	0.830	8	44	Datapath_compo..._Out_reg[3]/C	Datapath_comp..._17/RAMC_D1/I	8.822	2.660	6.162	10.0	CLK	CLK		

Η συχνότητα λειτουργίας του επεξεργαστή (μέγιστη) προέρχεται αν το slack του χειρότερου κρίσιμου Path γίνει 0 , δηλαδή ελάχιστη περίοδος = 10 – 0.433 = **9.567 ns,** άρα μέγιστη συχνότητα 1/9.567= **104.535 MHz.**

Η χειρότερη σύντομη διαδρομή είναι το Path 11: **0.447 ns**



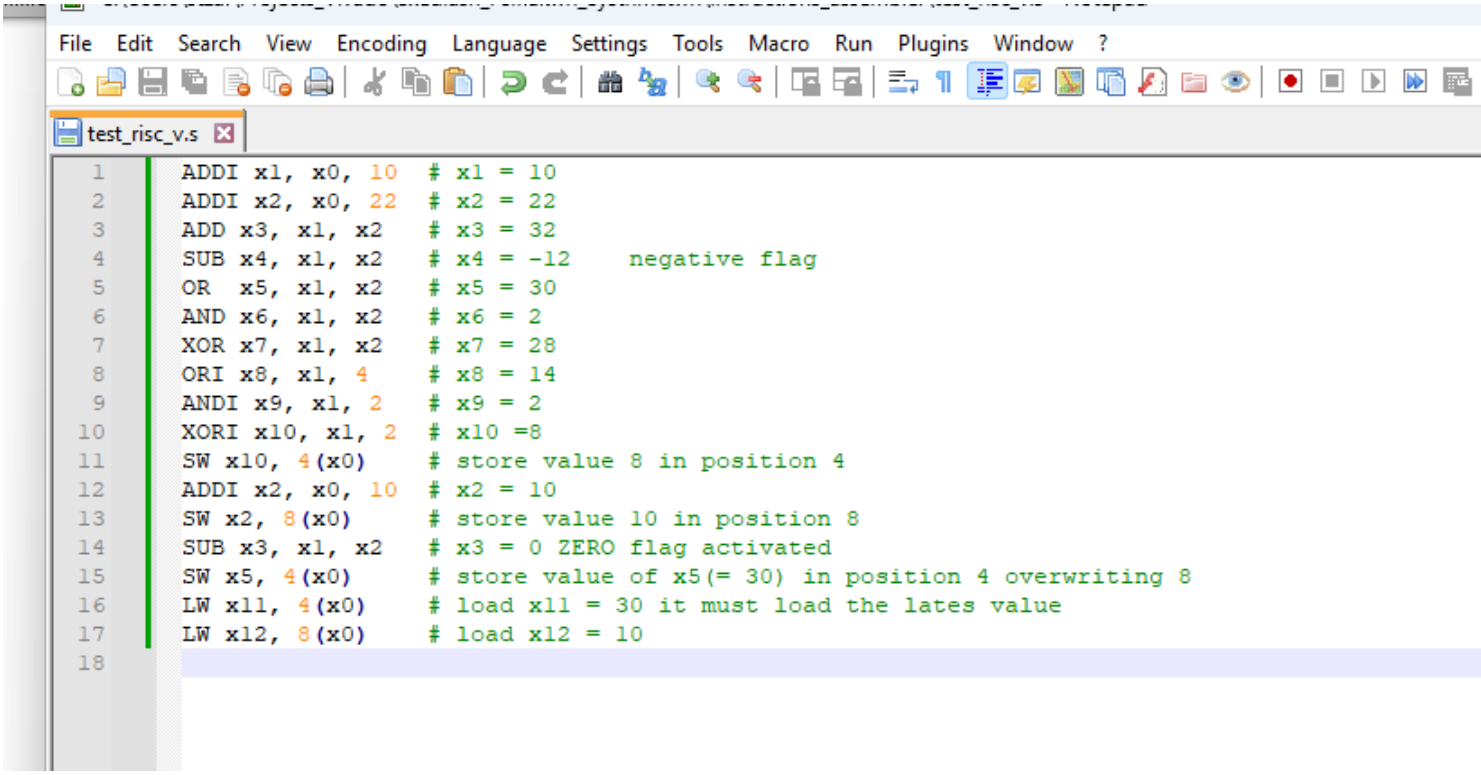
Impl Runs | DRC | Methodology | Power | Timing

Intra-Clock Paths - CLK - Hold

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Cloc
Path 11	0.342	1	2	Datapath_comp...Out_reg[18]/C	Datapath_comp...Out_reg[18]/D	0.447	0.256	0.191	0.0	CLK	CLK		
Path 12	0.342	1	2	Datapath_comp...Out_reg[26]/C	Datapath_comp...Out_reg[26]/D	0.447	0.256	0.191	0.0	CLK	CLK		
Path 13	0.342	1	2	Datapath_comp...Out_reg[30]/C	Datapath_comp...Out_reg[30]/D	0.447	0.256	0.191	0.0	CLK	CLK		
Path 14	0.378	1	2	Datapath_comp...Out_reg[18]/C	Datapath_comp...Out_reg[19]/D	0.483	0.292	0.191	0.0	CLK	CLK		
Path 15	0.378	1	2	Datapath_comp...Out_reg[26]/C	Datapath_comp...Out_reg[27]/D	0.483	0.292	0.191	0.0	CLK	CLK		
Path 16	0.378	1	2	Datapath_comp...Out_reg[30]/C	Datapath_comp...Out_reg[31]/D	0.483	0.292	0.191	0.0	CLK	CLK		
Path 17	0.391	1	2	Datapath_comp...Out_reg[29]/C	Datapath_comp...Out_reg[29]/D	0.496	0.249	0.247	0.0	CLK	CLK		
Path 18	0.398	1	2	Datapath_comp...Out_reg[14]/C	Datapath_comp...Out_reg[14]/D	0.503	0.256	0.247	0.0	CLK	CLK		
Path 19	0.398	1	2	Datapath_comp...Out_reg[22]/C	Datapath_comp...Out_reg[22]/D	0.503	0.256	0.247	0.0	CLK	CLK		
Path 20	0.398	1	2	Datapath_comp...Out_reg[10]/C	Datapath_comp...Out_reg[10]/D	0.503	0.256	0.247	0.0	CLK	CLK		

7-2. Επαλήθευση της ορθής σχεδίασης και λειτουργίας του επεξεργαστή

7-2.1



Το παραπάνω πρόγραμμα σε συμβολική γλώσσα επαληθεύει την σωστή λειτουργία με τον εξής τρόπο :

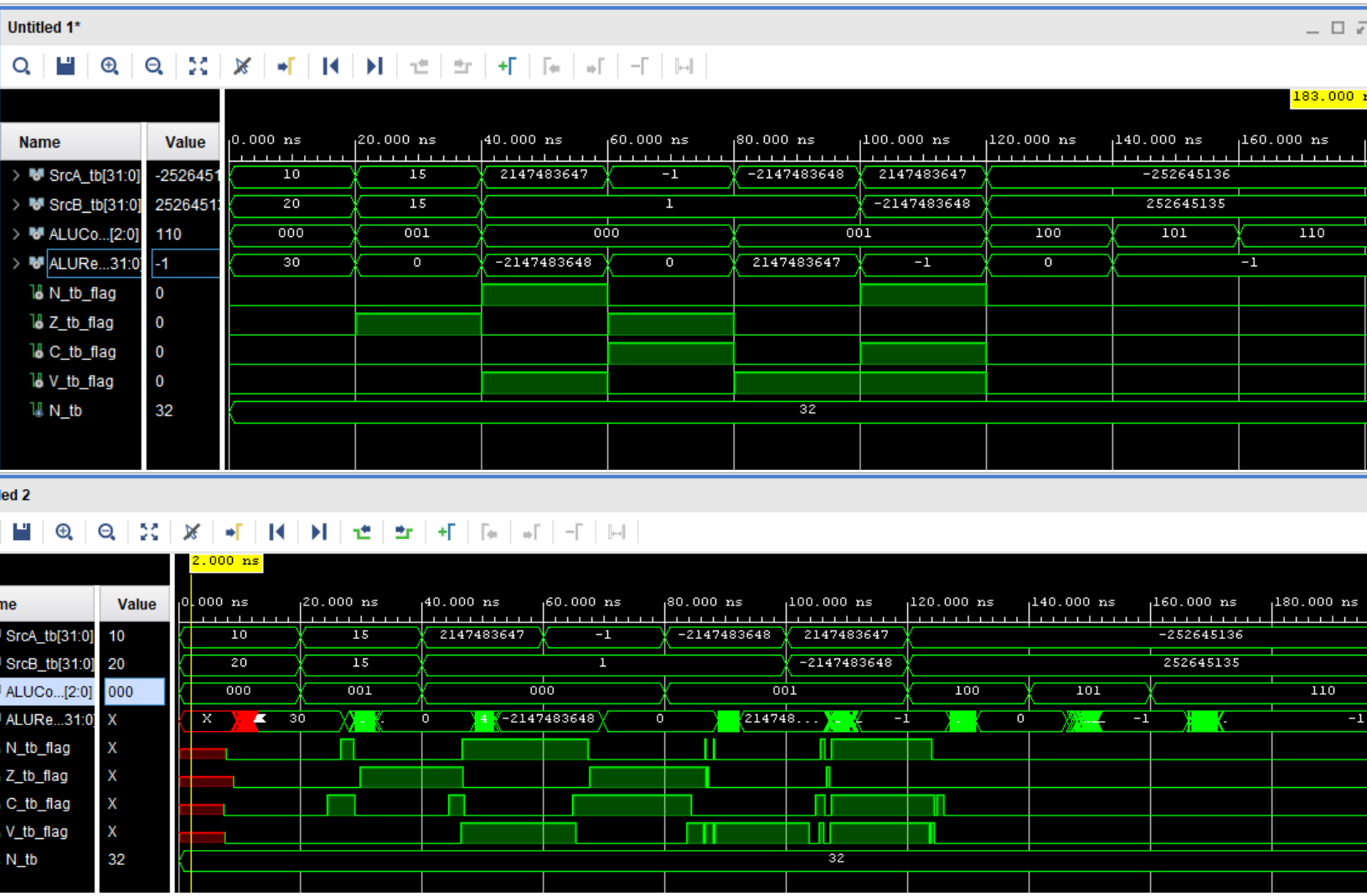
Οι εντολές **ADDI** χρησιμοποιούνται για την εισαγωγή αρχικών τιμών στους καταχωρητές. Αυτό μπορεί να ελεγχθεί από το ALUResult καθώς και το Result του επεξεργαστή. Επίσης επιβεβαιώνεται η σωστή λειτουργία του extend unit για τον δεύτερο τελεστή της ALU.

Έπειτα, με την εκτέλεση της εντολής **ADD** μπορούμε να ελέγξουμε αν οι τιμές της ADDI αποθηκεύτηκαν σωστά στους καταχωρητές καθώς και αν η πράξη ADD δουλεύει σωστά. Σε αυτό το σημείο επιβεβαιώνεται και η σωστή λειτουργία του πολυπλέκτη ALUSrc ο οποίος μέχρι τώρα (στις εντολές immediate) επέλεγε την δεύτερη είσοδο της ALU από το Extend unit ενώ τώρα πρέπει να επιλέξει από το

RegisterFile για να λειτουργήσει σωστά. Επίσης, βλέπουμε την ορθή ενημέρωση του REGwrite από το control unit. Παρόμοιους ελέγχους πετυχαίνουμε και με τις υπόλοιπες R-type εντολές, οι οποίες εκτελούνται για να ελέγξουμε τα αποτελέσματα όλων των πράξεων. Στην **SUB** ελέγχουμε και το Negative flag και το Zero flag στην εντολή 14. Το V flag ήταν δύσκολο να ελεγχθεί γιατί θα έπρεπε να κάνουμε ADDI με έναν πολύ μεγάλο σταθερό αριθμό και καθώς δεν έχουμε στο instruction set εντολή LUI θα έπρεπε να καλείται επαναληπτικά. Επομένως, άφησα τον έλεγχο για το αν δουλεύει σωστά το OVERFLOW flag μόνο στο testbench της ALU. Με τις εντολές **XORI**, **ANDI**, **ORI** και **XOR**, **AND**, **OR** ελέγχουμε τα σωστά αποτελέσματα των πράξεων, δηλαδή ότι το control unit κάνει σωστά το decoding και στέλνει τα σωστά σήματα στο datapath και άρα παίρνουμε τα σωστά αποτελέσματα. Τέλος, ελέγχουμε τις εντολές **LW**, **SW**. Αρχικά φορτώνουμε στην θέση μνήμης 4 μία τιμή και μετά στην θέση μνήμης 8. Η λειτουργία της Store αυτή κάθε αυτή, μπορεί να ελεγχθεί μέσω του ALUResult, έτσι καταλαβαίνουμε αν το Extend έγινε σωστά (άρα και αν το σωστό immsrc σήμα ήρθε από το control unit) και αν το σήμα ALUSrc ενημερώθηκε σωστά ώστε το ALUResult να είναι η σωστή διεύθυνση που έχουμε δώσει. Επιπλέον, μπορούμε να ελέγξουμε τα writedata για να δούμε αν τα δεδομένα που θέλουμε να γράψουμε πάνε όντως προς εγγραφή στην RAM. Για να ελένξουμε αν όντως γράφτηκαν στην μνήμη RAM, εφόσον δεν έχουμε πρόσβαση ούτε στην μνήμη ούτε στο MemWrite σήμα (σε implemented design), μπορούμε να κάνουμε LW και να δούμε αν θα πάρουμε την αναμενόμενη τιμή. Αυτό συμβαίνει στις δύο τελευταίες εντολές, και μάλιστα κάνω Load από την θέση 4 που μόλις έκανα overwrite την παλιά τιμή για να δω αν έχει προλάβει να ενημερωθεί και παίρνω την σωστή τιμή. Έπειτα, κάνω load και από την επόμενη θέση για να δω ότι πολλαπλές τιμές στην μνήμη έχουν αποθηκευτεί σωστά.

Ταυτόχρονα με όλα αυτά μπορούμε να βλέπουμε ποια εντολή εκτελείται από το Instr_out output του επεξεργαστή καθώς και αν ενημερώνεται σωστά ο Program Counter.

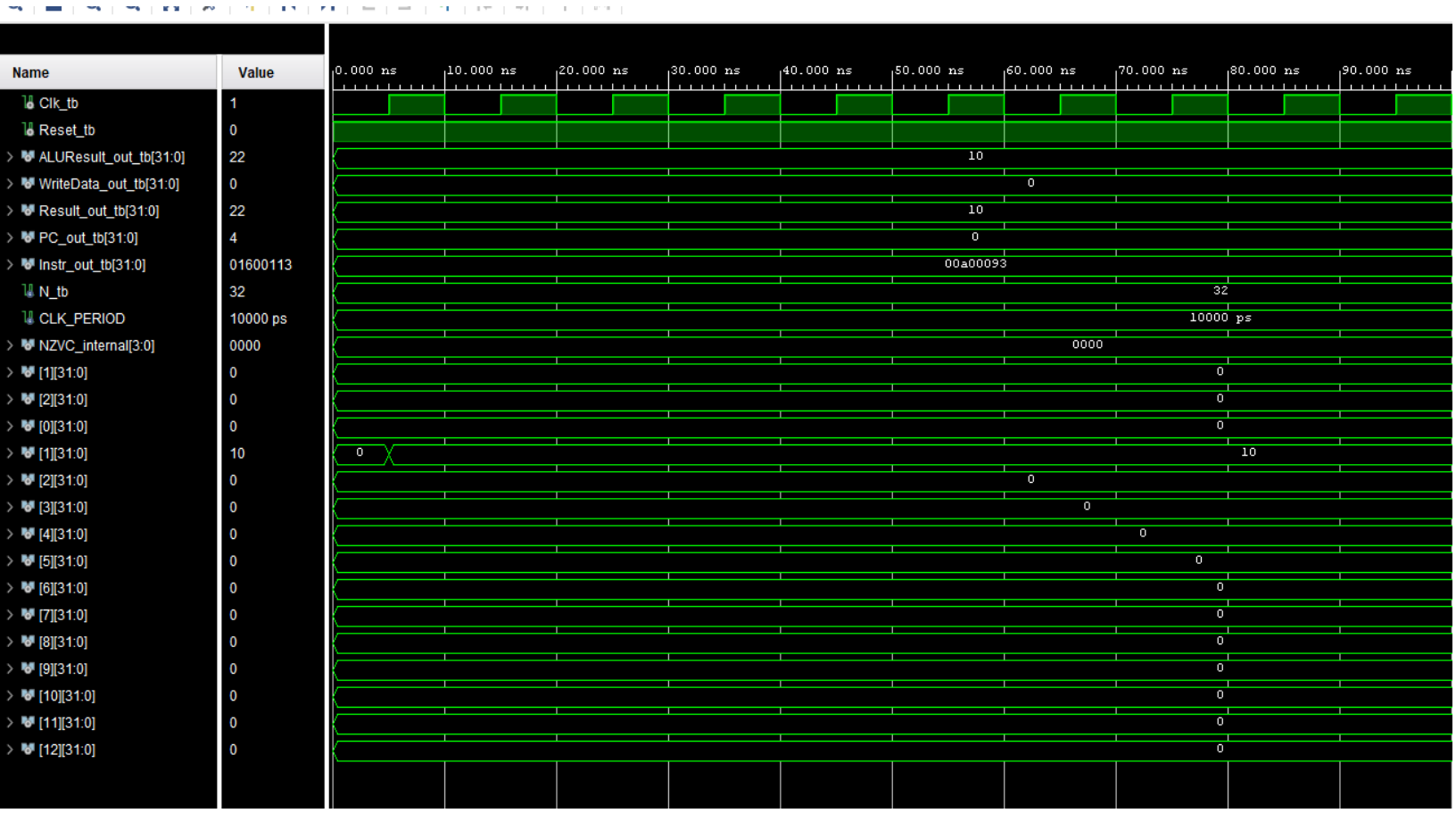
7-2.2 Διαγράμματα χρονισμού της **ALU**

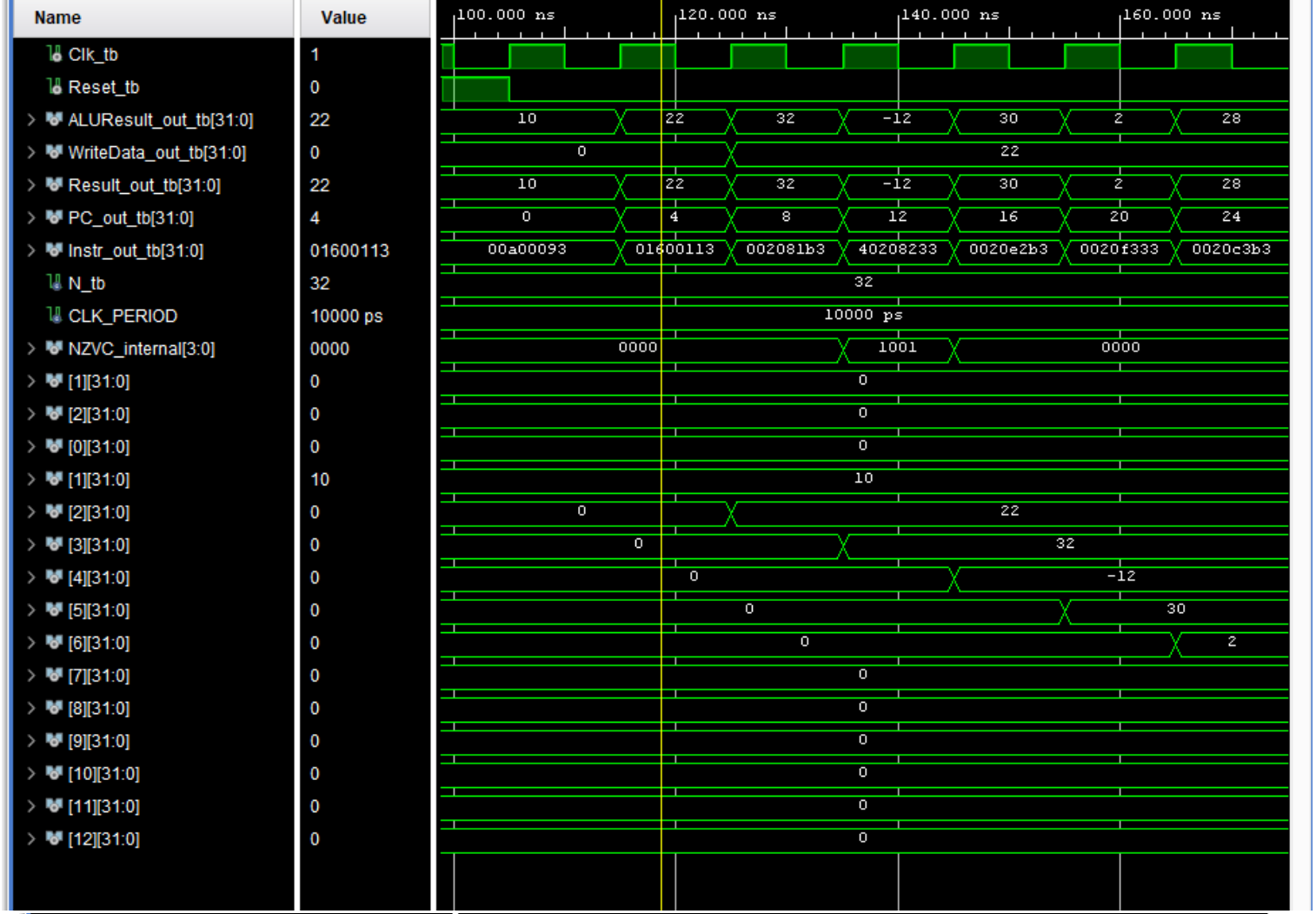


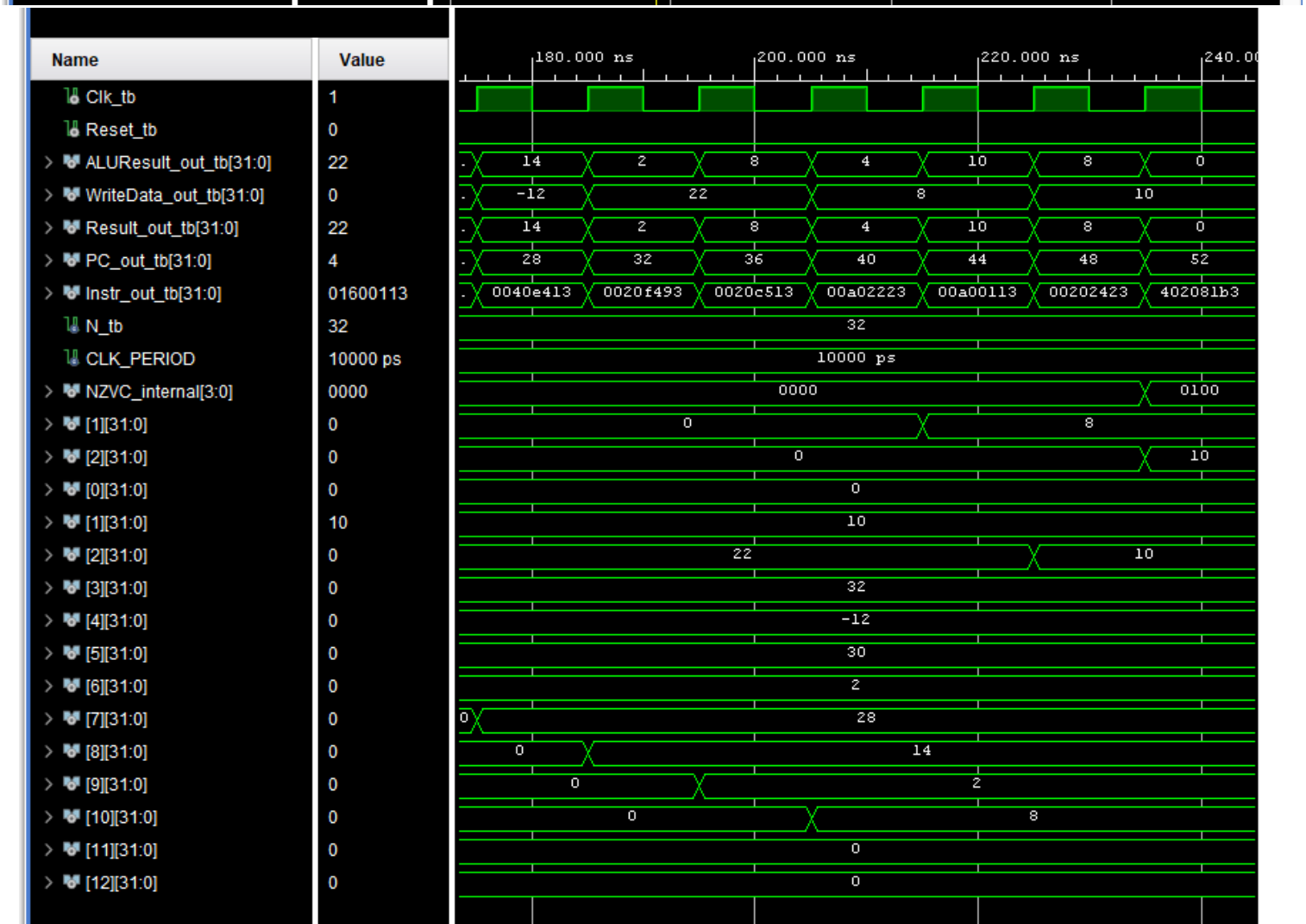
Παραπάνω βλέπουμε την εκτέλεση του behavioral και Post-Implementation Timing simulation της ALU. Ελέγχονται με αυτή τη σειρά οι πράξεις: **add** (ALUControl = 000), **sub** (ALUControl = 001) με flag Z =1, **add** με overflow, **add** με carry και zero flag, **sub** με overflow (και υπερχείληση και υποχείληση). Τέλος ελέγχονται οι πράξεις **AND**, **OR** και **XOR** όπου όλα τα flags είναι 0 και τα αποτελέσματα πρέπει να είναι **0** στο AND και **FFFFFFFF** = -1 στα άλλα 2 αφού οι είσοδοι στο 16αδικό είναι **F0F0F0F0 0F0F0F0F.** Στο Post-implementation-timing simulation βλέπουμε τις πραγματικές

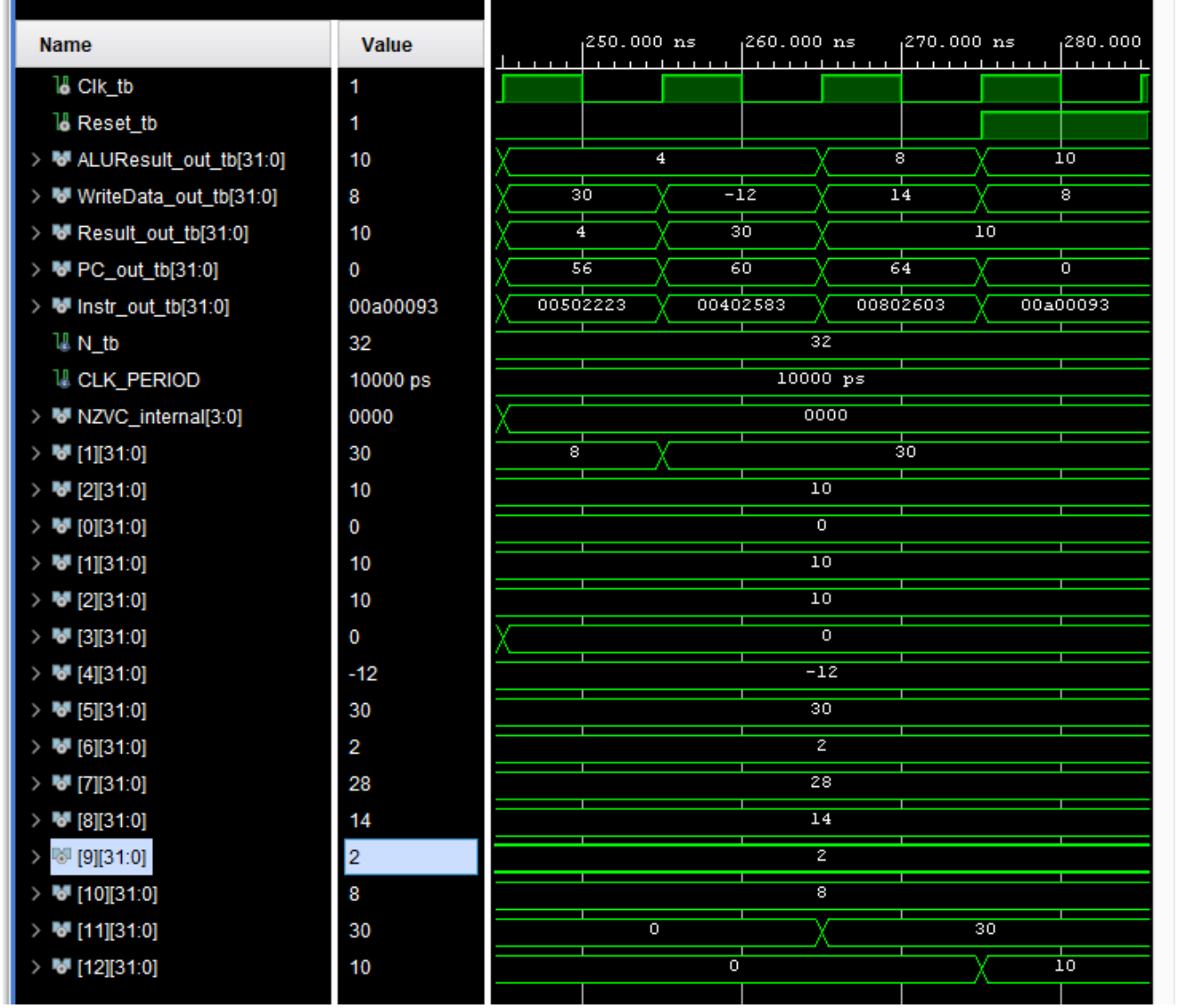
καθυστερήσεις και κάποια glitches που είναι λογικό να υπάρχουν αφού δεν ενημερώνονται όλα αμέσως όπως στο ιδεατό behavioral simulation. Το σημαντικό είναι, ότι οι τιμές ενημερώνονται και σταθεροποιούνται πριν την αλλαγή της επόμενης τιμής. Αλλάζω τιμές κάθε 20 ns γιατί οι καθυστερήσεις διάδοσης της ALU ως μεμονωμένο unit, είναι 15 ns οπότε η περίοδος 10 ns θα προκαλούσε προβλήματα.

7.2.3 Behavioral Simulation για ολόκληρο τον επεξεργαστή.




Name
Value
Clk_tb
1
Reset_tb
0
ALUResult_out_tb[31:0]
22
WriteData_out_tb[31:0]
0
Result_out_tb[31:0]
22
PC_out_tb[31:0]
4
Instr_out_tb[31:0]
01600113
N_tb
32
CLK_PERIOD
10000 ps
NZVC_internal[3:0]
0000
[1][31:0]
0
[2][31:0]
0
[0][31:0]
0
[1][31:0]
10
[2][31:0]
0
[3][31:0]
0
[4][31:0]
0
[5][31:0]
0
[6][31:0]
0
[7][31:0]
0
[8][31:0]
0
[9][31:0]
0
[10][31:0]
0
[11][31:0]
0
[12][31:0]
0
100.000 ns
120.000 ns
140.000 ns
160.000 ns


Name
Value
Clk_tb
1
Reset_tb
0
ALUResult_out_tb[31:0]
22
WriteData_out_tb[31:0]
0
Result_out_tb[31:0]
22
PC_out_tb[31:0]
4
Instr_out_tb[31:0]
01600113
N_tb
32
CLK_PERIOD
10000 ps
NZVC_internal[3:0]
0000
[1][31:0]
0
[2][31:0]
0
[0][31:0]
0
[1][31:0]
10
[2][31:0]
0
[3][31:0]
0
[4][31:0]
0
[5][31:0]
0
[6][31:0]
0
[7][31:0]
0
[8][31:0]
0
[9][31:0]
0
[10][31:0]
0
[11][31:0]
0
[12][31:0]
0
180.000 ns
200.000 ns
220.000 ns
240.00



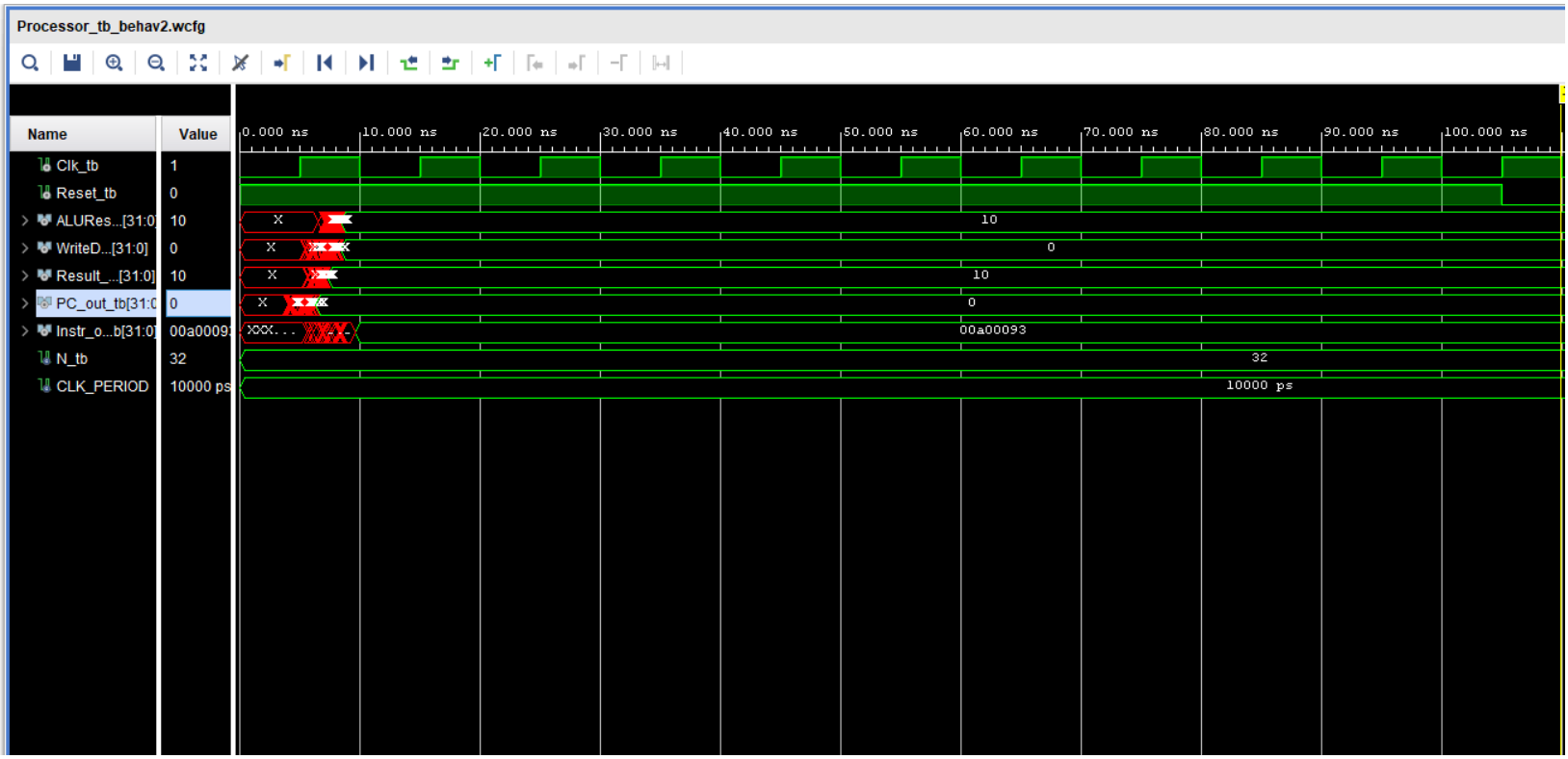
Στα διαγράμματα χρονισμού του επεξεργαστή (behavioral model) βλέπουμε για τις πράξεις που έχουμε αναφέρει στο 7.2.1 τα εξής αποτελέσματα:

Αρχικά τα πρώτα 100ns έχει τεθεί το Reset σε 1 για να είμαστε σίγουροι πως όλα τα στοιχεία του συστήματος έχουν μπει στην αρχική τους κατάσταση (Global reset). Έπειτα, βλέπουμε πως καθ' όλη την διάρκεια της προσομοίωσης ο Program Counter αυξάνεται κατά 4 όπως αναμένεται και πως οι εντολές Instr_out φορτώνονται με την σειρά. Πιο συγκεκριμένα, στις 2 πρώτες πράξεις που είναι ADDI, βλέπουμε την τιμή στο ALUResult και στο Result ενώ τα write data δεν μας ενδιαφέρουν γιατί το αντίστοιχο σήμα writeEnable είναι 0 από το Control unit. Έπειτα, στις υπόλοιπες R-type πράξεις αλλά και στις I-type (μέχρι και την 10 εντολή XORI) τα αποτελέσματα είναι τα επιθυμητά όπως έχουν σχολιαστεί δίπλα από κάθε εντολή συμβολικής γλώσσας στο 7.2.1 βήμα. Η SW έπειτα βλέπουμε ότι αποθηκεύει την σωστή τιμή (write data = 8) στην σωστή θέση (AluResult = 4). Η επόμενη εντολή ADDI συμπεριφέρεται όπως η προηγούμενες καθώς και η SW που αποθηκεύει την τιμή 10 στην θέση 8. Έπειτα έχουμε μία αφαίρεση που μας δείχνει και το flag Zero = 1

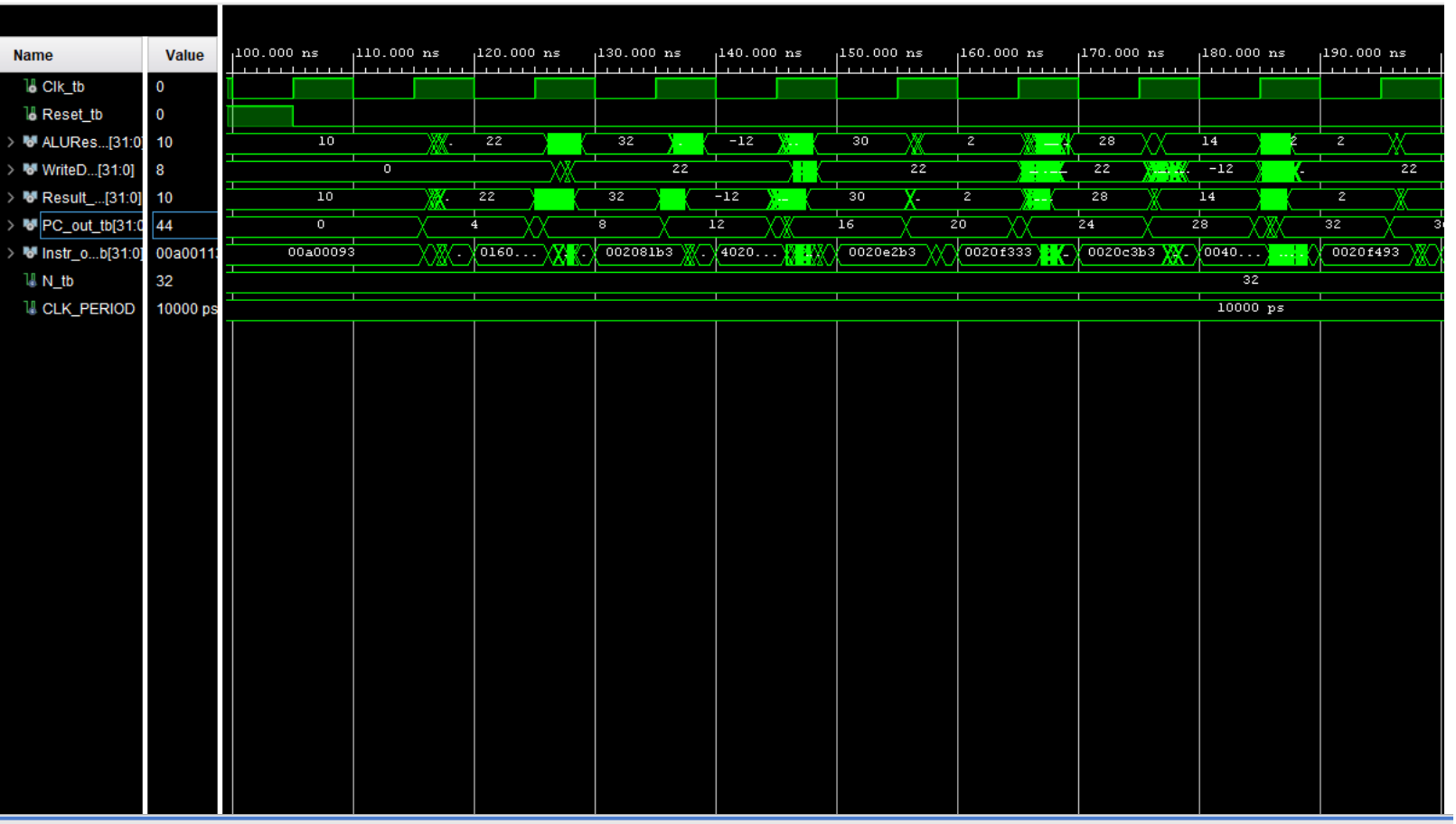
ορθώς. Τέλος, κάνουμε μία SW ακόμα στην θέση μνήμης 4 στην οποία είχαμε ήδη αποθηκεύσει δεδομένα για να δούμε αν θα γίνει σωστά η LW έπειτα. Η αποθήκευση της τιμής γίνεται σωστά και οι 2 τελευταίες εντολές LOAD φέρνουν τις αναμενόμενες τιμές μόνο στο Result πεδίο αυτήν την φορά (γιατί στο ALU result είναι η διεύθυνση της LOAD).

Επειδή είναι behavioral simulation μπορούμε να έχουμε πρόσβαση στα εσωτερικά σήμετα και στις θέσεις μνήμης – καταχωρητές , άρα να βλέπουμε και εκεί τις ενημερώσεις όπως αναφέρει το βήμα **6-2.5**. Συγκεκριμένα, τα σήματα που έχω προσθέσει κάτω από το NZVC_internal είναι: οι θέσεις μνήμης της μνήμης δεδομένων (θέση 4 και 8) και μετά οι πρώτοι 12 καταχωρητές που χρησιμοποιώ στο συμβολικό μου πρόγραμμα. Βλέπουμε πως η ενημέρωση του αρχείου καταχωρητών πραγματοποιείται στην ανοδική παρυφή του ρολογιού στο τέλος του κύκλου εκτέλεσης (δηλαδή στην αρχή του επόμενου). Άρα, η νέα τιμή καθίσταται διαθέσιμη προς ανάγνωση από την αρχή του επόμενου κύκλου ρολογιού αφού γίνεται ασύγχρονο διάβασμα.

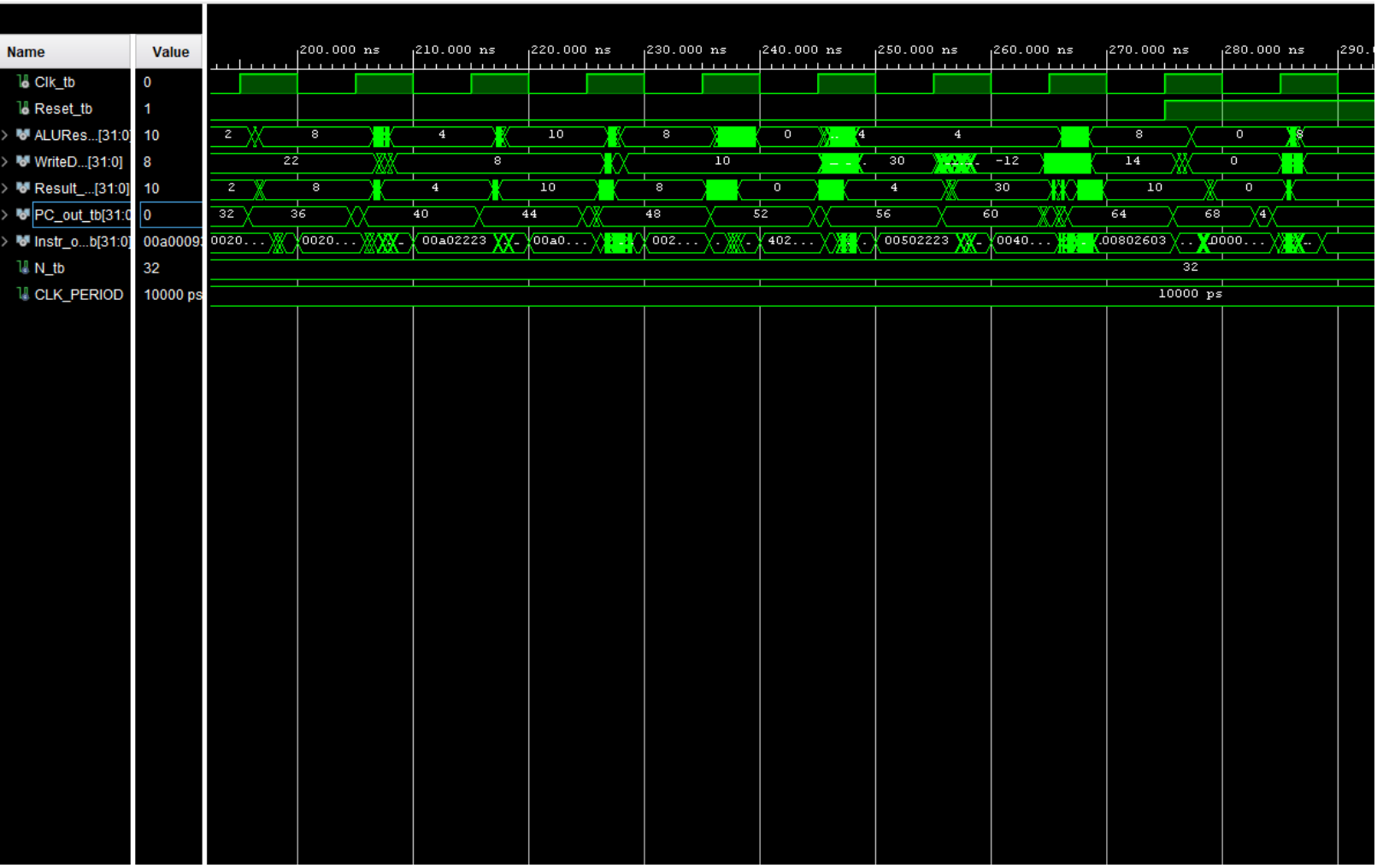
Post-implementation-Timing simulation



Processor_tb_behav2.wcfg



Processor_tb_behav2.wcfg

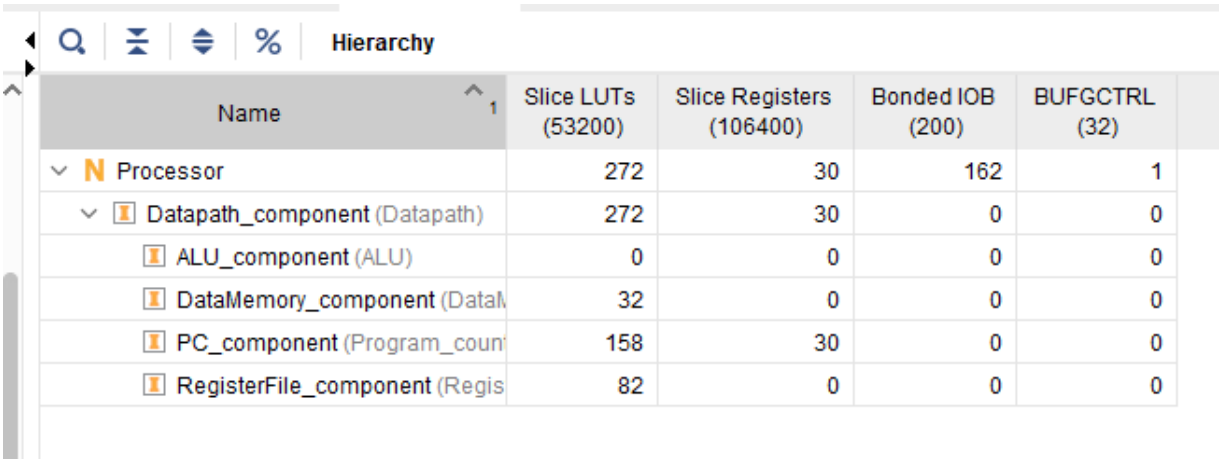


Η ίδια συμπεριφορά παρατηρείται και στο Post Implementation Timing simulation, με μόνη διαφορά τα glitches που υπάρχουν μέχρι την σταθεροποίηση των εξόδων και επίσης βλέπουμε τις πραγματικές καθυστερήσεις του κυκλώματος, αφού δεν γίνονται όλα αμέσως όπως στο behavioral simulation. Παρόλα αυτά, δεν έχουμε κάποιο πρόβλημα αφού η περίοδος έχει τεθεί σε 10ns και οι τιμές αλλάζουν κάθε 10 ns οπότε έτσι τηρούμε τους χρονικούς περιορισμούς του βήματος 7.1.4. Επίσης, δεν έχουμε πρόσβαση στα εσωτερικά σήματα στο Post Implementation Timing simulation, γιατί τα σήματα έχουν συγχωνευτεί με άλλα, και γενικά το εργαλείο έχει φτιάξει την βέλτιστη netlist λογική οπότε δεν μπορούμε να βρούμε ευκολά ποιο καλώδιο αντιστοιχεί σε κάθε εσωτερικό σήμα. Οι έξοδοι ακολουθούν την ορθότητα αποτελεσμάτων του behavioral simulation και σε αυτό το στάδιο ελέγχου.

7-3. **Ανάλυση των αποτελεσμάτων της σύνθεσης και της υλοποίησης του επεξεργαστή**

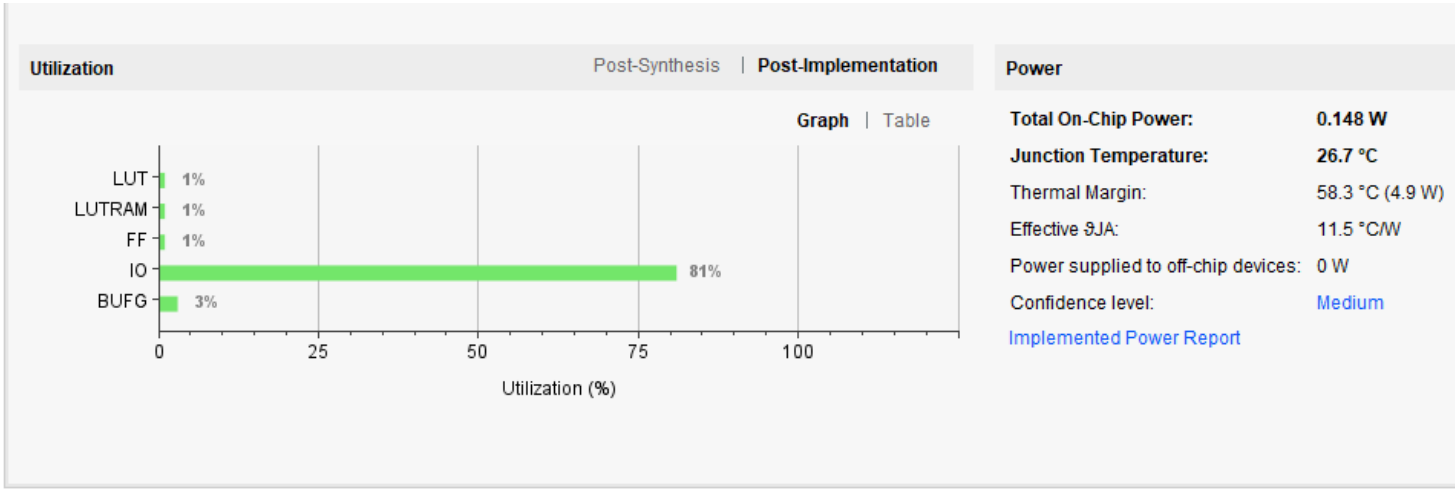
7-3.1

Utilization μετά την **σύνθεση**



Name	Slice LUTs (53200)	Slice Registers (106400)	Bonded IOB (200)	BUFGCTRL (32)
Processor	272	30	162	1
Datapath_component (Datapath)	272	30	0	0
ALU_component (ALU)	0	0	0	0
DataMemory_component (DataM	32	0	0	0
PC_component (Program_coun	158	30	0	0
RegisterFile_component (Regis	82	0	0	0

Utilization μετά την **υλοποίηση**



Design Runs | Utilization | DRC | Methodology | Power

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Bonded IOB (200)	BUFGCTRL (32)
Processor	250	30	82	174	76	162	1
Datapath_component (Datapath)	250	30	82	174	76	0	0
ALU_component (ALU)	0	0	8	0	0	0	0
DataMemory_component (DataM	32	0	8	0	32	0	0
PC_component (Program_coun	157	30	62	157	0	0	0
RegisterFile_component (Regis	62	0	28	18	44	0	0

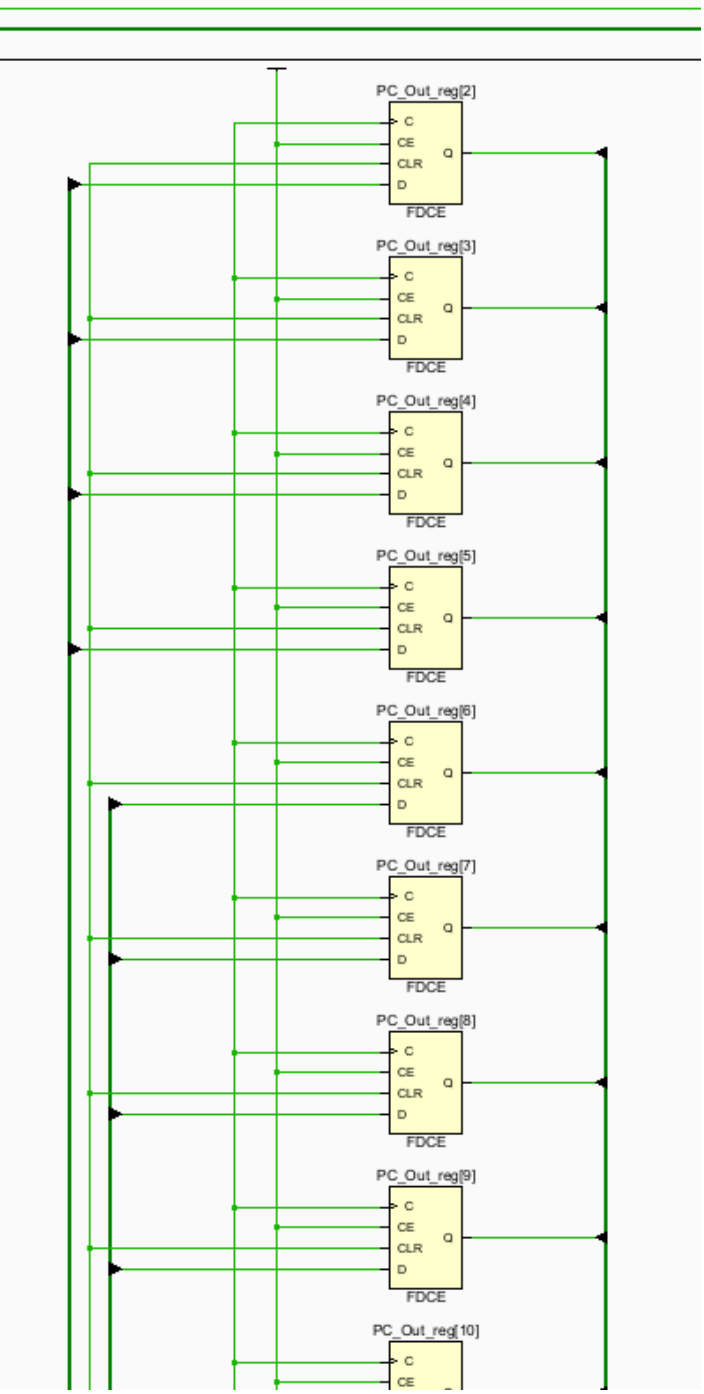
Primitives

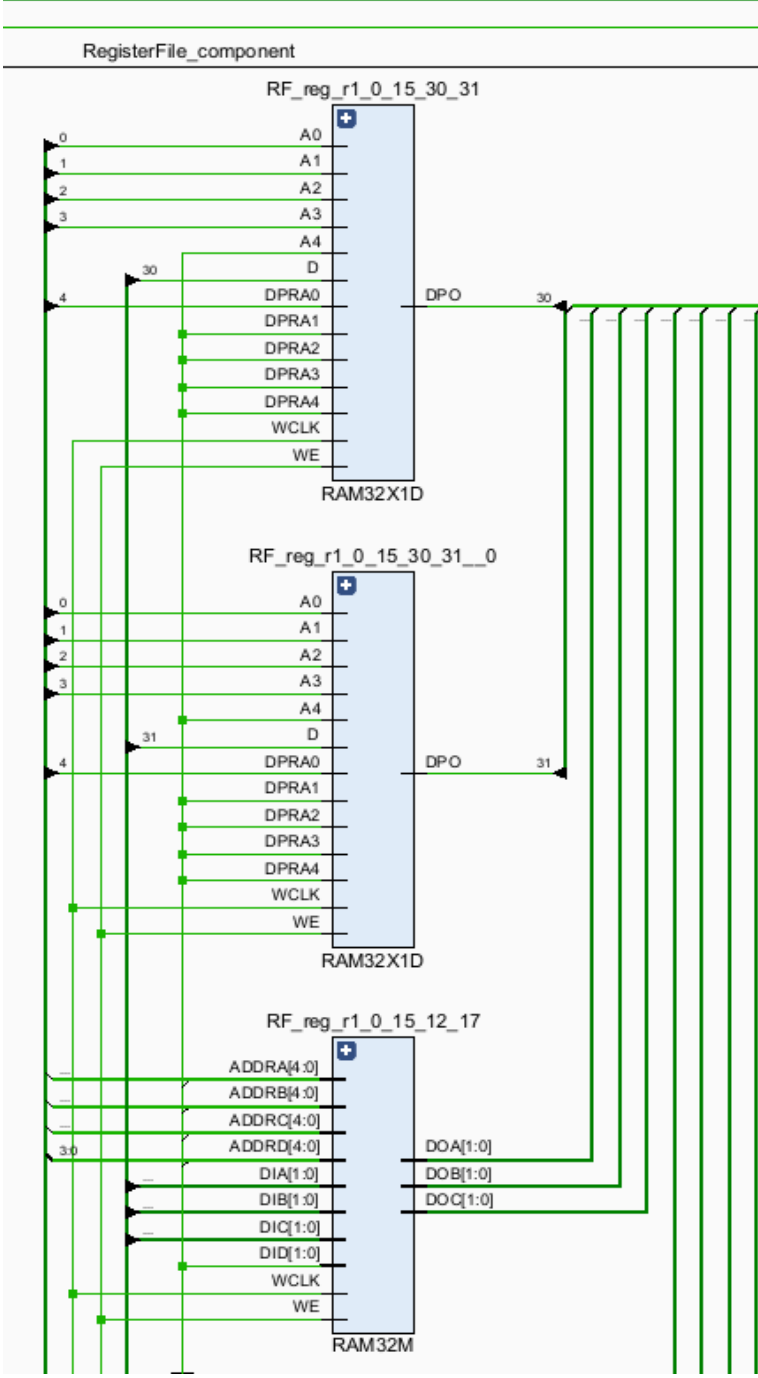
Ref Name	Used	Functional Category
OBUF	160	IO
LUT6	74	LUT
RAMD32	68	Distributed Memory
LUT5	61	LUT
LUT3	58	LUT
RAMS32	52	Distributed Memory
LUT2	34	LUT
FDCE	30	Flop & Latch
CARRY4	16	CarryLogic
LUT4	5	LUT
IBUF	2	IO
LUT1	1	LUT
BUFG	1	Clock

Βλέπουμε την **μη** χρησιμοποίηση
Latches από τα Primitives του utilization

Βλέπουμε ότι χρησιμοποιούνται (στο implemented design):

- **250** LUTs.
    - Από αυτά, τα **174** χρησιμοποιήθηκαν ως **Logic** για την υλοποίηση της συνδυαστικής λογικής (ALU, Πολυπλέκτες, Control Unit).
    - Τα υπόλοιπα **76** χρησιμοποιήθηκαν ως **Memory (LUTRAM)** Distributed RAM και όχι Block RAMs.
    - Η ύπαρξη **16 μονάδων CARRY4** (όπως φαίνεται στα Primitives) επιβεβαιώνει την υλοποίηση των αθροιστών των 32-bit για την ALU και τον υπολογισμό του PC (8 CARRY4 ανά αθροιστή 32-bit).
- **30** Flip-Flops.
    - Ο αριθμός αυτός αντιστοιχεί αποκλειστικά στον καταχωρητή του **Program Counter (PC)**. Επειδή οι εντολές είναι των 32-bit (4 bytes) και η διευθυνσιοδότηση είναι ευθυγραμμισμένη, τα 2 λιγότερο σημαντικά bits του PC είναι πάντα 0 και δεν χρειάζεται να αποθηκευτούν σε φυσικά Flip-Flops, για αυτό και τα 30 αντί για 32 FFs.
    - Το **Register File δεν υλοποιήθηκε με Flip-Flops**, αλλά ως Distributed RAM (LUTRAM), εξοικονομώντας σημαντικούς πόρους καταχωρητών.





Οι παραπάνω φωτογραφίες δείχνουν τα FF's του PC και την RAM υλοποίηση των Register Files.

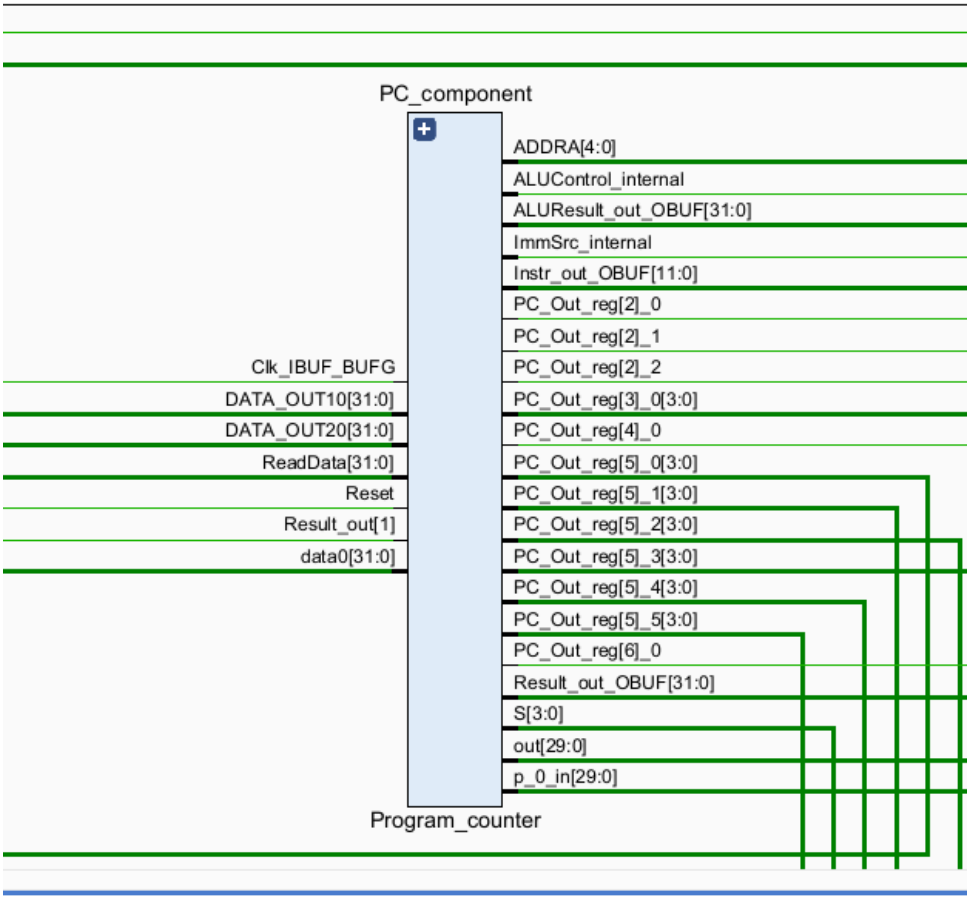
Και στις δύο αναφορές (Synthesis & Implementation), το **Control Unit** δεν εμφανίζεται ως ξεχωριστή οντότητα στην ιεραρχία. Επειδή το Control Unit αποτελείται αποκλειστικά από συνδυαστική λογική, το εργαλείο σύνθεσης ενσωμάτωσε την λογική του απευθείας στο Datapath_component.

Μία ακόμη παρατήρηση είναι ότι δεν είδα ROM blocks για το Instruction Memory όμως βλέποντας το Synthesis Log φαίνεται ότι δεν υπήρξαν ανεπιθύμητες απλοποιήσεις στη Μνήμη Εντολών αφού υπάρχει το :

INFO: [Synth 8-638] synthesizing module 'Instruction_memory' ...

INFO: [Synth 8-256] done synthesizing module 'Instruction_memory' (0#1)

Η Μνήμη Εντολών υλοποιήθηκε ως **συνδυαστική λογική** χρησιμοποιώντας Logic LUTs και όχι ως μνήμη. Γι' αυτόν τον λόγο οι πόροι της ενσωματώνονται στη γενική λογική του Datapath_component. Η ενσωμάτωση επιβεβαιώνεται από το ότι οι πόροι LUT του PC_component (157 LUTs) είναι αυξημένοι, υποδεικνύοντας ότι η συνδυαστική λογική της ROM συγχωνεύτηκε με τη λογική του PC. Επιπλέον, η ορθή λειτουργία της φαίνεται και από την σωστή εκτέλεση του προγράμματος στην προσομοίωση.



Παρατηρούμε ότι το **PC_component** οδηγεί απευθείας τα σήματα εξόδου **Instr_out**. Αυτό αποδεικνύει ότι η συνδυαστική λογική που αντιστοιχίζει διευθύνσεις σε εντολές δηλαδή η μνήμη εντολών, έχει ενσωματωθεί εντός του PC_component, δικαιολογώντας την αυξημένη χρήση LUTs.

Σημείωση: Εκτός από τα ζητούμενα simulation έχω κάνει και κάποια και σε **RegisterFile, ControlUnit** ως unit tests πριν προχωρήσω παρακάτω. Μιας και η δομή της αναφοράς ήταν αυστηρή έχω βάλει κάποια screenshots από αυτές τις κυματομορφές σε έναν φάκελο extra_simulations.

Τζαφέρης Στέφανος

1115202200183