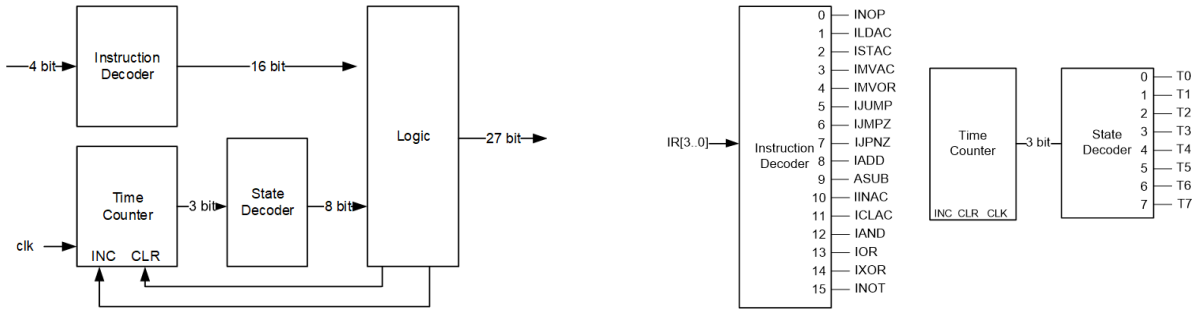
HARDWIRED ΜΟΝΑΔΑ ΕΛΕΓΧΟΥ

4

Σκοπός

Με αφορμή την σχεδίαση και την εξομοίωση με διάφορους τρόπους, απλών ψηφιακών κυκλωμάτων θα κατακτηθεί το αντικείμενο της άσκησης αυτής που είναι η σχεδίαση της μονάδας ελέγχου, χρησιμοποιώντας την hardwired λογική η οποία θα χρησιμοποιηθεί, εναλλακτικά με την microprogrammed, κατά την τελική σύνθεση της σχετικά απλής ΚΜΕ.

Η μονάδα ελέγχου είναι αυτή που παρέχει στην ΚΜΕ τα απαραίτητα σήματα ελέγχου για τη λειτουργία της. Η λογική σχεδίασής της θα είναι η hardwired λογική, η οποία θα υλοποιηθεί με μία μηχανή πεπερασμένων καταστάσεων – FSM. Η μηχανή καταστάσεων αποτελείται από δύο αποκωδικοποιητές, ένα μετρητή και ένα συνδυαστικό κύκλωμα. Ο πρώτος αποκωδικοποιητής (instruction decoder) παράγει ένα ξεχωριστό σήμα για κάθε εντολή ενώ ο δεύτερος αποκωδικοποιητής (state decoder), με τη βοήθεια ενός απαριθμητή(time counter), παρακολουθεί ποια κατάσταση του κύκλου ανάκλησης η εκτέλεσης κάθε εντολής είναι ενεργή. Τέλος μια μονάδα συνδυαστικής λογικής παράγει μέσα από τα ξεχωριστά σήματα, σήματα ελέγχου για κάθε αποκωδικοποιητή αλλά και για τον απαριθμητή. Μια τέτοια μονάδα ελέγχου θα είχε την ακόλουθη μορφή (Σχήμα 1a).



(a) ***(b)***

Σχήμα 1: *Λογικό διάγραμμα HardWired Μονάδας Ελέγχου.*

Η σχεδίαση του αποκωδικοποιητή εντολών είναι σχετικά απλή. Δέχεται σαν είσοδο την έξοδο του καταχωρητή εντολών (IR) ενώ δεδομένου ότι χρησιμοποιούμαι μόνο τα 4 bit του καταχωρητή εντολών για το ρεπερτόριο των 16 εντολών της σχετικά απλής ΚΜΕ είναι προφανές ότι ο αποκωδικοποιητής εντολών είναι ένα αποκωδικοποιητής 4 σε 16. Από την άλλη εφόσον ο μέγιστος αριθμός καταστάσεων για το ρεπερτόριο των 16 εντολών είναι 8 καταστάσεις στη σχεδίαση μας χρησιμοποιούμαι έναν απαριθμητή 3 bit με δυνατότητα αύξησης και μηδενισμού και ένα

αποκωδικοποιητή 3 σε 8. Τα παραπάνω στοιχεία και οι έξοδοι τους φαίνονται με μεγαλύτερη λεπτομέρεια στο Σχήμα 1b.
Η ρουτίνα FETCH είναι η μόνη ρουτίνα η οποία δεν χρησιμοποιείται από το αποκωδικοποιητή εντολών. Δεδομένου ότι κατά τη ρουτίνα αυτή η προς εκτέλεση εντολή ανακαλείται από τη μνήμη η έξοδος του αποκωδικοποιητή μπορεί να είναι οποιαδήποτε. Σε αυτή μας τη σχεδίαση αναθέτουμε την κατάσταση T0 στην FETCH1 θέλοντας να εκμεταλλευτούμε το γεγονός ότι αυτή είναι προσπελάσιμη καθαρίζοντας (clear) τον απαριθμητή καταστάσεων. Όμοια αναθέτουμε την κατάσταση T1 και T2 στην FETCH2 και FETCH3 αντίστοιχα. Οι καταστάσεις των προς εκτέλεση εντολών εξαρτώνται αφενός από το opcode κάθε εντολής και αφετέρου από την τιμή του απαριθμητή καταστάσεων. Η T3 είναι η πρώτη χρονικά κατάσταση κάθε εντολής, η T4 η δεύτερη και ούτω καθεξής. Η μονάδα ελέγχου συνδέοντας με λογική and την κατάλληλη τιμή του απαριθμητή καταστάσεων με την έξοδο του αποκωδικοποιητή εντολών παράγει τις επιμέρους καταστάσεις για κάθε εντολή. Για παράδειγμα οι δύο πρώτες καταστάσεις της εντολής LDAC είναι:

LDAC1 = ILDAC ^ T3
LDAC2 = ILDAC ^ T4

Η συνολική λίστα των επιμέρους καταστάσεων για όλες τις εντολές δίνεται στο πίνακα Γ.4.1 που ακολουθεί.

κατάσταση	**λειτουργία**	**κατάσταση**	**λειτουργία**
FETCH1	T0	**JMPZY1**	IJMPZ ^ Z ^ T3
FETCH2	T1	**JMPZY2**	IJMPZ ^ Z ^ T4
FETCH3	T3	**JMPZY3**	IJMPZ ^ Z ^ T5
NOP1	INOP ^ T3	**JMPZN1**	IJMPZ ^ Z′ ^ T3
LDAC1	ILDAC ^ T3	**JMPZN2**	IJMPZ ^ Z′ ^ T4
LDAC2	ILDAC ^ T4	**JPNZY1**	IJPNZ ^ Z′ ^ T3
LDAC3	ILDAC ^ T5	**JPNZY2**	IJPNZ ^ Z′ ^ T4
LDAC4	ILDAC ^ T6	**JPNZY3**	IJPNZ ^ Z′ ^ T5
LDAC5	ILDAC ^ T7	**JPNZN1**	IJPNZ ^ Z ^ T3
STAC1	ISTAC ^ T3	**JPNZN2**	IJPNZ ^ Z ^ T4
STAC2	ISTAC ^ T4	**ADD1**	IADD ^ T3
STAC3	ISTAC ^ T5	**SUB1**	ISUB ^ T3
STAC4	ISTAC ^ T6	**INAC1**	IINAC ^ T3
STAC5	ISTAC ^ T7	**CLAC1**	ICLAC ^ T3
MVAC1	IMVAC ^ T3	**AND1**	IAND ^ T3
MOVR1	IMOVR ^ T3	**OR1**	IOR ^ T3
JUMP1	IJUMP ^ T3	**XOR1**	IXOR ^ T3
JUMP2	IJUMP ^ T4	**NOT1**	INOT ^ T3
JUMP3	IJUMP ^ T5		

Πίνακας 1: *Παραγωγή καταστάσεων για τη σχετικά απλή ΚΜΕ*

Έχοντας δημιουργήσει τις επιμέρους καταστάσεις για κάθε εντολή είναι ανάγκη να δημιουργήσουμε τα σήματα που θα οδηγούν τις εισόδους inc και clr του απαριθμητή καταστάσεων. Για να το επιτύχουμε αυτό συνδέουμε με λογική or την τελευταία κατάσταση κάθε εντολής για να δημιουργήσουμε το σήμα που θα οδηγήσει την είσοδο clr. Δεδομένου ότι η είσοδος inc πρέπει να είναι ενεργοποιημένη σε κάθε άλλη κατάσταση , μπορεί να υλοποιηθεί συνδέοντας με λογική or όλες τις υπόλοιπες καταστάσεις (πλην της τελευταίας) κάθε εντολής. Τέλος, η συνδυαστική λογική που

χρειάζεται για να παραχθούν τα κατάλληλα σήματα ελέγχου , για τα επιμέρους τμήματα της ΚΜΕ φαίνονται στο Πίνακα 2 που ακολουθεί:

Σήμα	Συνδιαστική Λογική
ARLOAD	FETCH1∨FETCH3∨LDAC3∨STAC3
ARINC	LDAC1∨STAC1∨JMPZY1∨JPNZY1
PCLOAD	JUMP3∨JMPZY3∨JPNZY3
PCINC	FETCH2∨LDAC1∨LDAC2∨STAC1∨STAC2∨JMPZN1∨JMPZN2∨JPNZN1∨JPNZN2
DRLOAD	FETCH2∨LDAC1∨LDAC2∨LDAC4∨STAC1∨STAC2∨STAC4∨JUMP1∨JUMP2∨JMPZY1∨JMPZY2∨ JPNZY1∨JPNZY2
TRLOAD	LDAC2 ∨STAC2 ∨JUMP2 ∨JMPZY2 ∨JPNZY2
IRLOAD	FETCH3
RLOAD	MVAC1
ACLOAD	LDAC5∨MOVR1∨ADD1∨SUB1∨INAC1∨CLAC1∨AND1∨OR1∨XOR1∨NOT1
ZLOAD	LDAC5∨MOVR1∨ADD1∨SUB1∨INAC1∨CLAC1∨AND1∨OR1∨XOR1∨NOT1
READ	FETCH2∨LDAC1∨LDAC2∨LDAC4∨STAC1∨STAC2∨JUMP1∨JUMP2∨JMPZY1∨JMPZY2∨JPNZY1∨ JPNZY2
WRITE	STAC5
MEMBUS	FETCH2∨LDAC1∨LDAC2∨LDAC4∨STAC1∨STAC2∨JUMP1∨JUMP2∨JMPZY1∨JMPZY ∨JPNZY1∨JPNZY2
BUSMEM	STAC5
PCBUS	FETCH1 or FETCH3
DRBUS	LDAC2∨LDAC3∨LDAC5∨STAC2∨STAC3∨STAC5∨JUMP2∨JUMP3∨JMPZY2∨JMPZY3∨JPNZY2∨ JPNZY3
TRBUS	LDAC3∨STAC3∨JUMP3∨JMPZY3∨JPNZY3
RBUS	MOVR1∨ADD1∨SUB1∨AND1∨OR1∨XOR1
ACBUS	STAC4∨MVAC1
ANDOP	AND1
OROP	OR1
XOROP	XOR1
NOTOP	NOT1
ACINC	INAC1
ACZERO	CLAC1
PLUS	ADD1
MINUS	SUB1

Πίνακας 2: *Παραγωγή σημάτων ελέγχου για τη σχετικά απλή ΚΜΕ*

Αποκωδικοποιητής Εντολών

Γράψτε τον κώδικα για τον αποκωδικοποιητή 4 σε 16 με σήμα εισόδου D_{in} εύρους 4 bit και σήμα εξόδου D_{out} εύρους 16 bit. Το κύκλωμα αυτό όπως είναι γνωστό θα αντιστοιχεί την τιμή (opcode) κάθε μιας από τις 16 εντολές που εμφανίζεται στην είσοδο του σε μία από τις 16 εξόδους του.

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;



ENTITY decdr4to16 IS

  PORT (
    Din  : in  STD_LOGIC_VECTOR (3 downto 0);
    Dout : out STD_LOGIC_VECTOR (15 downto 0)
  );

END decdr4to16;

ARCHITECTURE bhv of decdr4to16 IS
BEGIN

  WITH Din SELECT
    Dout <= "0000000000000001" WHEN "0000", --Pinakas 1, klasikh syntaksh IR(opcode)x4(dhladh IR&00)
          "0000000000000010" WHEN "0001",
          "0000000000000100" WHEN "0010",
          "0000000000001000" WHEN "0011",
          "0000000000010000" WHEN "0100",
          "0000000000100000" WHEN "0101",
          "0000000001000000" WHEN "0110",
          "0000000010000000" WHEN "0111",
          "0000000100000000" WHEN "1000",
          "0000001000000000" WHEN "1001",
          "0000010000000000" WHEN "1010",
```

```
"0000100000000000" WHEN "1011",
"0001000000000000" WHEN "1100",
"0010000000000000" WHEN "1101",
"0100000000000000" WHEN "1110",
"1000000000000000" WHEN "1111",
"0000000000000000" WHEN OTHERS;

END bhv;
```

Πρόγραμμα 1: *Ο αποκωδικοποιητής εντολών.*

Αποκωδικοποιητής Καταστάσεων

Γράψτε τον κώδικα για τον αποκωδικοποιητή 3 σε 8 με σήμα εισόδου D_{in} εύρους 3 bit και σήμα εξόδου D_{out} εύρους 8 bit. Το κύκλωμα αυτό θα αντιστοιχεί την τιμή μέτρησης από τον μετρητή που εμφανίζεται στην είσοδο του σε μία από τις 8 εξόδους του η οποίες και θα συμβολίζουν την παρούσα κατάσταση.

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;



ENTITY decdr3to8 IS

  PORT (
    Din  : in  STD_LOGIC_VECTOR (2 downto 0);
    Dout : out STD_LOGIC_VECTOR (7 downto 0)
  );

END decdr3to8;

ARCHITECTURE bhv of decdr3to8 IS
BEGIN

  WITH Din SELECT
    Dout <= "00000001" WHEN "000", --Pinakas 1(states), ginetai antistoixhsh me tis times toy IR&00 gia to kalesma twn entolwn systhmatos
          "00000010" WHEN "001",
          "00000100" WHEN "010",
          "00001000" WHEN "011",
          "00010000" WHEN "100",
          "00100000" WHEN "101",
          "01000000" WHEN "110",
          "10000000" WHEN "111",
          "00000000" WHEN OTHERS;

END bhv;
```

Πρόγραμμα 2: *Ο αποκωδικοποιητής καταστάσεων.*

Απαριθμητής

Γράψτε τον κώδικα για έναν μετρητή με εύρος 3-bits με σήματα εισόδου/ελέγχου inc για την αύξηση κατά ένα και rst για εκκαθάριση και σήμα εξόδου count .

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY counter_3bit IS   --κλασικός up counter με διαφορά το inc για buffering, δηλαδή έχουμε 3 FF
  PORT (
    clk    : IN STD_LOGIC;
    reset  : IN STD_LOGIC;
    inc    : IN STD_LOGIC;
    counter : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
  );
END counter_3bit;

ARCHITECTURE bhv OF counter_3bit IS
  SIGNAL counter_up: STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN

  PROCESS(clk) -- με συγχρονο reset, με ασυγχρονο δημιουργει προβληματα στις μεταβασεις των bit του mOPs των εντολων
  BEGIN
    IF rising_edge(clk) THEN
      IF (reset = '1') THEN
          counter_up <= "000";
```

```
        ELSIF (inc = '1') THEN
          counter_up <= counter_up + 1;
        END IF;
      END IF;
  END PROCESS;

counter <= counter_up;

END bhv;
```

Πρόγραμμα 3: *Ο απαριθμητής των 3-bits.*

Μονάδα Ελέγχου.

Έχοντας ολοκληρώσει τη συγγραφή του κώδικα για τα επιμέρους στοιχεία που συνθέτουν την μονάδα ελέγχου και αφού όλα συγκεντρωθούν σε μία βιβλιοθήκη, μπορεί πλέον να γραφεί το συνολικό πρόγραμμα περιγραφής της μονάδας ελέγχου. Σημειώνεται εδώ ότι δεδομένου ότι το κύκλωμα παραγωγής των σημάτων ελέγχου τόσο της ΚΜΕ όσο και του μετρητή καταστάσεων (σχήμα 1) είναι εξαιρετικά απλό δεν είναι απαραίτητη η συγγραφή ξεχωριστού στοιχείου για αυτό.

Γράψτε τον κώδικα για τη βιβλιοθήκη (package), με το όνομα hardwiredlib, η οποία θα περιέχει τα επιμέρους στοιχεία που συνθέτουν την μονάδα ελέγχου.

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE hardwiredlib IS

  COMPONENT decdr4to16
    PORT (
      Din  : IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
      Dout : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
    );
  END COMPONENT;

  COMPONENT decdr3to8
    PORT (
      Din  : IN  STD_LOGIC_VECTOR (2 DOWNTO 0);
      Dout : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
  END COMPONENT;



  COMPONENT counter_3bit IS
  PORT (
    clk    : IN STD_LOGIC;
    reset  : IN STD_LOGIC;
    inc    : IN STD_LOGIC;
    counter : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
   );
END COMPONENT;
```

END hardwiredlib;

Πρόγραμμα 4: *βιβλιοθήκη στοιχείων για την μονάδα ελέγχου.*

Με βάση το σκελετό που ακολουθεί (πρόγραμμα 5) γράψτε τον κώδικα περιγραφής για της μονάδας ελέγχου, δηλαδή της μηχανής πεπερασμένων καταστάσεων, έτσι όπως διαμορφώνεται από τα επιμέρους στοιχεία και το σχήμα 1. Τα σήματα που θα δέχεται σαν είσοδο το κύκλωμα, εκτός των σημάτων clock και reset, θα είναι τα τέσσερα (4) λιγότερο σημαντικά bit του καταχωρητή εντολών (ir) και η τιμή του καταχωρητή σημαίας (z). Σαν έξοδοι λαμβάνεται το σήμα mOPs που αντιστοιχεί στην κάθε μικροεντολή εύρους 27-bits.

Γράψτε εδώ το πρόγραμμά σας:

```
LIBRARY ieee;

USE ieee.std_logic_1164.all;

USE ieee.std_logic_unsigned.all;

USE work.hardwiredlib.all;


ENTITY hardwired is
  port(
    ir     : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
    clock,reset   : IN  STD_LOGIC;
    z      : IN  STD_LOGIC;
    mOPs    : OUT STD_LOGIC_VECTOR(26 DOWNTO 0)
  );
END hardwired;


ARCHITECTURE arc OF hardwired IS


--τα σήματα είναι από τα σχήματα/block του σχηματικού με κατάληξη out
  signal instruction_decdr_out : std_logic_vector(15 downto 0);
  signal time_counter_out : std_logic_vector(2 downto 0);
  signal state_decdr_out : std_logic_vector(7 downto 0);
  signal reset_out : std_logic;
```

```
    signal clear_cntr_out : std_logic;


    --σηματα για τα states,γινονται με την AND/^
    signal FETCH1, FETCH2, FETCH3 : std_logic;
    signal NOP1 : std_logic;
    signal LDAC1, LDAC2, LDAC3, LDAC4, LDAC5 : std_logic;
    signal STAC1, STAC2, STAC3, STAC4, STAC5 : std_logic;
    signal MVAC1, MOVR1 : std_logic;
    signal JUMP1, JUMP2, JUMP3 : std_logic;
    signal JMPZY1, JMPZY2, JMPZY3 : std_logic;
    signal JMPZN1, JMPZN2 : std_logic;
    signal JPNZY1, JPNZY2, JPNZY3 : std_logic;
    signal JPNZN1, JPNZN2 : std_logic;
    signal ADD1, SUB1, INAC1, CLAC1 : std_logic;
    signal AND1, OR1, XOR1, NOT1 : std_logic;


    -- σηματα για control με χρηση OR/πανε στην εξοδο mOPs,συγκεκριμενα σηματα εχουν sig_ γιατι
ειναι taken η ονομασιες απο το συστημα
    signal ARLOAD, ARINC, PCLOAD, PCINC, DRLOAD : std_logic;
    signal TRLOAD, IRLOAD, RLOAD, ACLOAD, ZLOAD : std_logic;
    signal sig_READ, sig_WRITE, MEMBUS, BUSMEM, PCBUS : std_logic;
    signal DRBUS, TRBUS, RBUS, ACBUS : std_logic;
    signal ANDOP, OROP, XOROP, NOTOP : std_logic;
    signal ACINC, ACZERO, PLUS, MINUS : std_logic;


BEGIN
    --reset του μετρητη, 0 για reset press η να τελειωσει η εντολη clear_cntr_out
    reset_out <= reset OR clear_cntr_out;


    INSTRUCTION_DECODER: decdr4to16
```

```
   port map (
      Din  => ir,
      Dout => instruction_decdr_out
   );

-- Συνδέουμε τον Counter
TIME_COUNTER: counter_3bit
   port map (
      clk    => clock,
      reset    => reset_out,
      inc    => '1', -- με 1 μετρα συνεχεια
      counter => time_counter_out
   );

-- Συνδέουμε τον State Decoder
STATE_DECODER: decdr3to8
   port map (
      Din  => time_counter_out ,
      Dout => state_decdr_out -- Εδώ βγαίνουν τα T0, T1, T2...
   );

--δημιουργια και αντιστοιχηση του Πινακα 1
FETCH1 <= state_decdr_out (0); -- T0
FETCH2 <= state_decdr_out (1); -- T1
FETCH3 <= state_decdr_out (2); -- T2

-- NOP
NOP1   <= instruction_decdr_out(0) AND state_decdr_out (3);
```

```
--LDAC
LDAC1  <= instruction_decdr_out(1) AND state_decdr_out (3);
LDAC2  <= instruction_decdr_out(1) AND state_decdr_out (4);
LDAC3  <= instruction_decdr_out(1) AND state_decdr_out (5);
LDAC4  <= instruction_decdr_out(1) AND state_decdr_out (6);
LDAC5  <= instruction_decdr_out(1) AND state_decdr_out (7);

--STAC
STAC1  <= instruction_decdr_out(2) AND state_decdr_out (3);
STAC2  <= instruction_decdr_out(2) AND state_decdr_out (4);
STAC3  <= instruction_decdr_out(2) AND state_decdr_out (5);
STAC4  <= instruction_decdr_out(2) AND state_decdr_out (6);
STAC5  <= instruction_decdr_out(2) AND state_decdr_out (7);

--MVAC, MOVR
MVAC1  <= instruction_decdr_out(3) AND state_decdr_out (3);
MOVR1  <= instruction_decdr_out(4) AND state_decdr_out (3);

--JUMP
JUMP1  <= instruction_decdr_out(5) AND state_decdr_out (3);
JUMP2  <= instruction_decdr_out(5) AND state_decdr_out (4);
JUMP3  <= instruction_decdr_out(5) AND state_decdr_out (5);

--JMPZ με Z variable
--Z=1 -->Yes Jump)
JMPZY1 <= instruction_decdr_out(6) AND z AND state_decdr_out (3);
JMPZY2 <= instruction_decdr_out(6) AND z AND state_decdr_out (4);
JMPZY3 <= instruction_decdr_out(6) AND z AND state_decdr_out (5);
--Z=0 -->No Jump)
```

```
JMPZN1 <= instruction_decdr_out(6) AND (NOT z) AND state_decdr_out (3);
JMPZN2 <= instruction_decdr_out(6) AND (NOT z) AND state_decdr_out (4);

--JPNZ με NOT Z variable
--Z=0 -->Yes Jump
JPNZY1 <= instruction_decdr_out(7) AND (NOT z) AND state_decdr_out (3);
JPNZY2 <= instruction_decdr_out(7) AND (NOT z) AND state_decdr_out (4);
JPNZY3 <= instruction_decdr_out(7) AND (NOT z) AND state_decdr_out (5);
--Z=1 -->No Jump
JPNZN1 <= instruction_decdr_out(7) AND z AND state_decdr_out (3);
JPNZN2 <= instruction_decdr_out(7) AND z AND state_decdr_out (4);

--εντολες απο alu
ADD1   <= instruction_decdr_out(8)  AND state_decdr_out (3);
SUB1   <= instruction_decdr_out(9)  AND state_decdr_out (3);
INAC1  <= instruction_decdr_out(10) AND state_decdr_out (3);
CLAC1  <= instruction_decdr_out(11) AND state_decdr_out (3);
AND1   <= instruction_decdr_out(12) AND state_decdr_out (3);
OR1    <= instruction_decdr_out(13) AND state_decdr_out (3);
XOR1   <= instruction_decdr_out(14) AND state_decdr_out (3);
NOT1   <= instruction_decdr_out(15) AND state_decdr_out (3);

--ενωνουμε ολα τα final states με OR, μη σπαταλη χρονου και οδηγηση σε T0/FETCH για να παρουμε την επομενη
clear_cntr_out <= NOP1 OR LDAC5 OR STAC5 OR MVAC1 OR MOVR1 OR JUMP3 OR
                  JMPZY3 OR JMPZN2 OR JPNZY3 OR JPNZN2 OR
                  ADD1 OR SUB1 OR INAC1 OR CLAC1 OR
                  AND1 OR OR1 OR XOR1 OR NOT1;
```

```
        --σηματα απο Πινακα 2-μεσω OR
ARLOAD <= FETCH1 OR FETCH3 OR LDAC3 OR STAC3;
PCLOAD <= JUMP3 OR JMPZY3 OR JPNZY3;
DRLOAD <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR STAC4 OR
          JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2 OR JPNZY1 OR JPNZY2;
IRLOAD <= FETCH3;
TRLOAD <= LDAC2 OR STAC2 OR JUMP2 OR JMPZY2 OR JPNZY2;
RLOAD  <= MVAC1;
ACLOAD <= LDAC5 OR MOVR1 OR ADD1 OR SUB1 OR INAC1 OR CLAC1 OR
          AND1 OR OR1 OR XOR1 OR NOT1;
ZLOAD  <= LDAC5 OR MOVR1 OR ADD1 OR SUB1 OR INAC1 OR CLAC1 OR
          AND1 OR OR1 OR XOR1 OR NOT1;
ARINC  <= LDAC1 OR STAC1 OR JMPZY1 OR JPNZY1;
PCINC  <= FETCH2 OR LDAC1 OR LDAC2 OR STAC1 OR STAC2 OR
          JMPZN1 OR JMPZN2 OR JPNZN1 OR JPNZN2;
sig_READ   <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR
          JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2 OR JPNZY1 OR JPNZY2;
sig_WRITE  <= STAC5;
PCBUS  <= FETCH1 OR FETCH3;
DRBUS  <= LDAC2 OR LDAC3 OR LDAC5 OR STAC2 OR STAC3 OR STAC5 OR
          JUMP2 OR JUMP3 OR JMPZY2 OR JMPZY3 OR JPNZY2 OR JPNZY3;
TRBUS  <= LDAC3 OR STAC3 OR JUMP3 OR JMPZY3 OR JPNZY3;
RBUS   <= MOVR1 OR ADD1 OR SUB1 OR AND1 OR OR1 OR XOR1;
ACBUS  <= STAC4 OR MVAC1;
MEMBUS <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR
          JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2 OR JPNZY1 OR JPNZY2;
```

```
    BUSMEM <= STAC5;
    ANDOP  <= AND1;
    OROP   <= OR1;
    XOROP  <= XOR1;
    NOTOP  <= NOT1;
    ACINC  <= INAC1;
    ACZERO <= CLAC1;
    PLUS   <= ADD1;
    MINUS  <= SUB1;

--συνδεδη σηματων με mOPs, απο 0-26 bits
    mOPs(0)  <= ARLOAD;
    mOPs(1)  <= PCLOAD;
    mOPs(2)  <= DRLOAD;
    mOPs(3)  <= IRLOAD;
    mOPs(4)  <= TRLOAD;
    mOPs(5)  <= RLOAD;
    mOPs(6)  <= ACLOAD;
    mOPs(7)  <= ZLOAD;
    mOPs(8)  <= ARINC;
    mOPs(9)  <= PCINC;
    mOPs(10) <= sig_READ;
    mOPs(11) <= sig_WRITE;
    mOPs(12) <= PCBUS;
    mOPs(13) <= DRBUS;
    mOPs(14) <= TRBUS;
    mOPs(15) <= RBUS;
    mOPs(16) <= ACBUS;
    mOPs(17) <= MEMBUS;
```

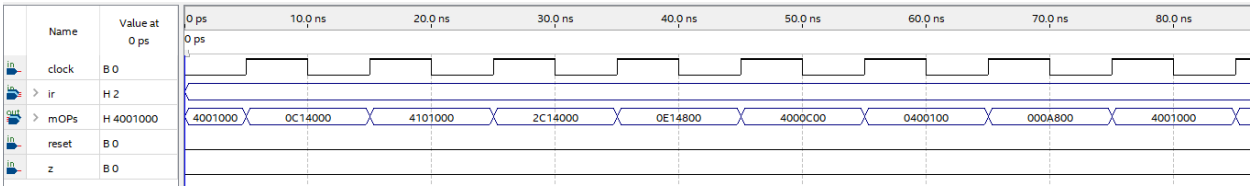
```
mOPs(18) <= BUSMEM;
mOPs(19) <= ANDOP;
mOPs(20) <= OROP;
mOPs(21) <= XOROP;
mOPs(22) <= NOTOP;
mOPs(23) <= ACINC;
mOPs(24) <= ACZERO;
mOPs(25) <= PLUS;
mOPs(26) <= MINUS;

END arc;
```

Πρόγραμμα 5: *Μονάδα Ελέγχου.*

Εξομοίωση της Μονάδας Ελέγχου.

Το επόμενο στάδιο περιλαμβάνει την εξομοίωση της μονάδας ελέγχου με τον Waveform Editor με σκοπό τον έλεγχο της λειτουργίας της. Με οδηγό τις προηγούμενες ασκήσεις, δημιουργήστε ένα καινούργιο project και εξομοιώστε τη λειτουργία της μονάδας ελέγχου με τη βοήθεια του Waveform Editor για έξι (6) εντολές της ΚΜΕ , της επιλογής σας.

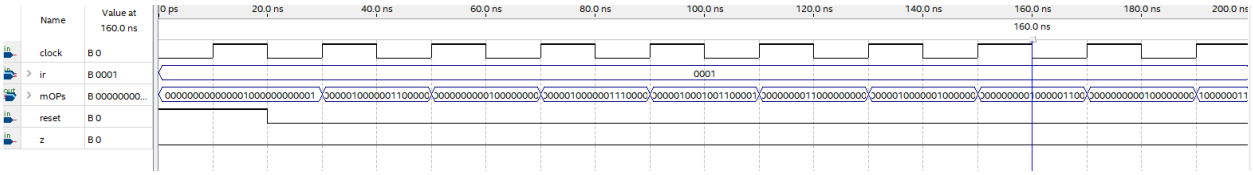
Σαν παράδειγμα ακολουθούν οι κυματομορφές εξομοίωσης για την εντολή STAC (ir=0x2).



Εικόνα 1: *Κυματομορφές εξομοίωσης εντολής STAC.*

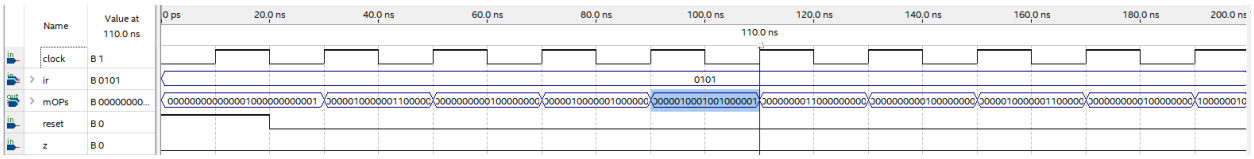
Τοποθετήστε εδώ τις κυματομορφές σας:

LDAC



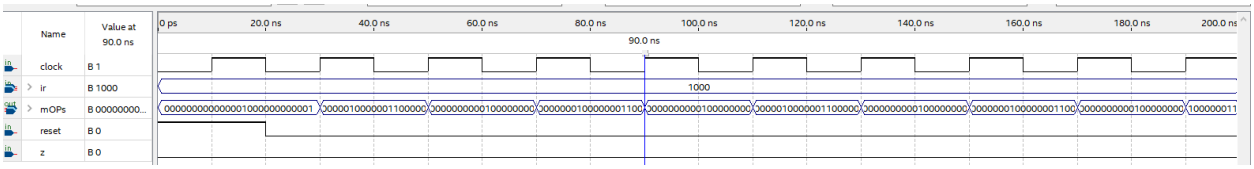
Τερματίζει στα 160ns (LDAC5), το επιβεβαιώνουμε παρακολουθώντας τις μεταβάσεις των mOPs bits έως την T7, οι μεταβάσεις μετά από αυτό το state είναι η αρχή της επόμενης εντολής(δουλεύει συνέχεια η CPU).

JUMP



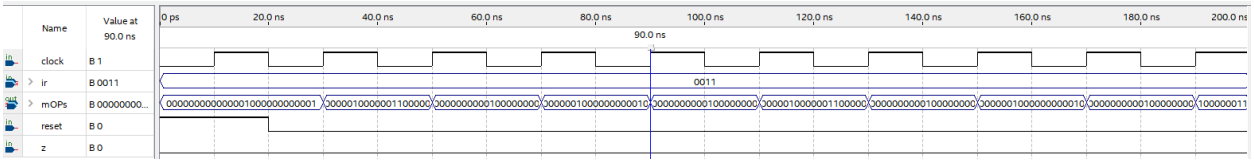
Τερματίζει στα 100-110ns , δηλαδή στο T4, μετά συνεχίζει για επόμενη εντολή.

ADD



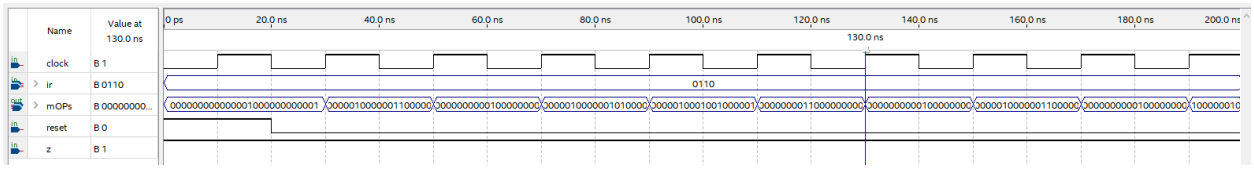
Τερματίζει στα 90ns, δηλαδή στο T3.

MVAC



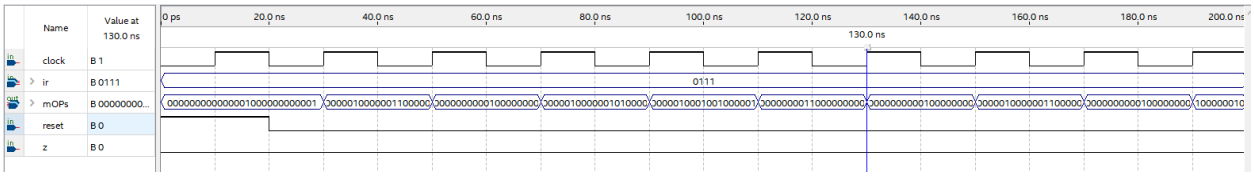
Τερματίζει στα 90ns, δηλαδή στο T3.

JMPZ



Τερματίζει στα 130ns, δηλαδή στο T5(z=1).

JPNZ



Τερματίζει στα 130ns, δηλαδή στο T5.

Εικόνα 2: *Κυματομορφές εξομοίωσης της μονάδας ελέγχου*