

HARDWIRED

ΜΟΝΑΔΑ ΕΛΕΓΧΟΥ

4

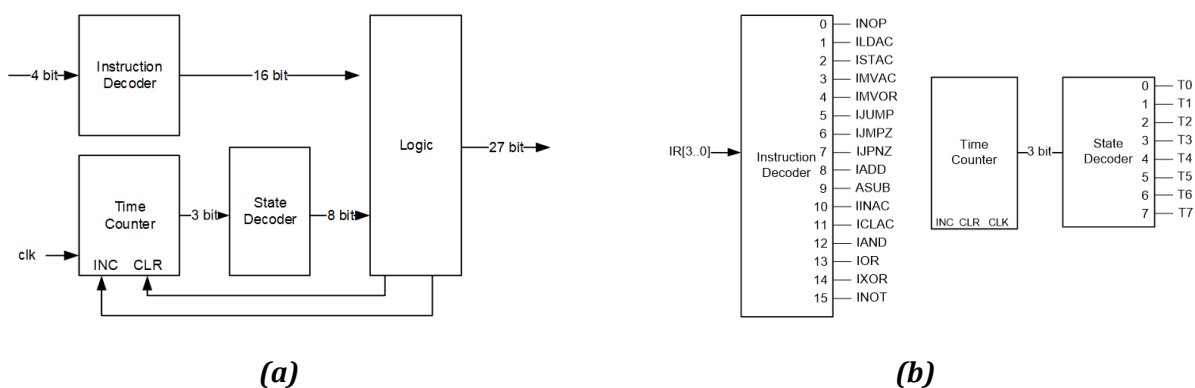
MPASTIALI SERIF

Ece20094

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 1α).



(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 χρησιμοποιούμε 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 χρησιμοποιούμε 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:

$$LDAC1 = ILDAC \wedge T3$$

$$LDAC2 = ILDAC \wedge T4$$

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

| Σήμα | Συνδυαστική 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∨JMPZY2∨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 |

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:

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 instr_decoder is
port(
Din : in std_logic_vector(3 downto 0);
Dout : out std_logic_vector(15 downto 0)
);
end instr_decoder;

architecture rtl of instr_decoder is
begin
process(Din)
begin
Dout <= (others => '0');
case Din is
when "0000" => Dout(0) <= '1'; -- NOP
when "0001" => Dout(1) <= '1'; -- LDAC
when "0010" => Dout(2) <= '1'; -- STAC
when "0011" => Dout(3) <= '1'; -- ADD
when "0100" => Dout(4) <= '1'; -- SUB
when "0101" => Dout(5) <= '1'; -- INAC
when "0110" => Dout(6) <= '1'; -- CLAC
when "0111" => Dout(7) <= '1'; -- MVAC
when "1000" => Dout(8) <= '1'; -- MOVR
when "1001" => Dout(9) <= '1'; -- AND
when "1010" => Dout(10) <= '1'; -- OR
```

```
when "1011" => Dout(11) <= '1'; -- XOR
when "1100" => Dout(12) <= '1'; -- NOT
when "1101" => Dout(13) <= '1'; -- JUMP
when "1110" => Dout(14) <= '1'; -- JMPZ
when "1111" => Dout(15) <= '1'; -- JPNZ
when others => null;

end case;

end process;

end rtl;
```

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 state_decoder is
port(
Din : in std_logic_vector(2 downto 0);
Dout : out std_logic_vector(7 downto 0)
);
end state_decoder;

architecture rtl of state_decoder is
begin
process(Din)
begin
Dout <= (others => '0');
case Din is
when "000" => Dout(0) <= '1'; -- T0
when "001" => Dout(1) <= '1'; -- T1
when "010" => Dout(2) <= '1'; -- T2
when "011" => Dout(3) <= '1'; -- T3
when "100" => Dout(4) <= '1'; -- T4
when "101" => Dout(5) <= '1'; -- T5
when "110" => Dout(6) <= '1'; -- T6
when "111" => Dout(7) <= '1'; -- T7
when others => null;
end case;
end process;
end rtl;
```

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:](#)

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:](#)

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 3627-bits.

[Γράψτε εδώ το πρόγραμμά σας:](#)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
library lpm;
use lpm.lpm_components.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

```

```

-- Instruction & state decoding

```

```

signal I : std_logic_vector(15 downto 0);

```

```

signal T : std_logic_vector(7 downto 0);

```

```

signal tc : std_logic_vector(2 downto 0);

```

```

signal inc_tc, clr_tc : std_logic;

```

```

-- FETCH states

```

```

signal FETCH1, FETCH2, FETCH3 : std_logic;

```

```

-- Instruction states

```

```

signal LDAC1,LDAC2,LDAC3,LDAC4,LDAC5 : std_logic;

```

```

signal STAC1,STAC2,STAC3,STAC4,STAC5 : std_logic;

```

```

signal ADD1,SUB1,INAC1,CLAC1,MVAC1 : std_logic;

```

```

signal AND1,OR1,XOR1,NOT1 : 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;

```

```

begin

```


ID: instr_decoder port map(ir, I);

SD: state_decoder port map(tc, T);

TC_1: counter3 port map(clock, clr_tc, inc_tc, tc);

-- FETCH

FETCH1 <= T(0);

FETCH2 <= T(1);

FETCH3 <= T(2);

-- Instruction states

LDAC1 <= I(1) and T(3); LDAC2 <= I(1) and T(4); LDAC3 <= I(1) and T(5);

LDAC4 <= I(1) and T(6); LDAC5 <= I(1) and T(7);

STAC1 <= I(2) and T(3); STAC2 <= I(2) and T(4); STAC3 <= I(2) and T(5);

STAC4 <= I(2) and T(6); STAC5 <= I(2) and T(7);

ADD1 <= I(3) and T(3); SUB1 <= I(4) and T(3);

INAC1 <= I(5) and T(3); CLAC1 <= I(6) and T(3);

MVAC1 <= I(7) and T(3);

AND1 <= I(9) and T(3); OR1 <= I(10) and T(3);

XOR1 <= I(11) and T(3); NOT1 <= I(12) and T(3);

JUMP1 <= I(13) and T(3); JUMP2 <= I(13) and T(4); JUMP3 <= I(13) and T(5);

JMPZY1 <= I(14) and z and T(3); JMPZY2 <= I(14) and z and T(4); JMPZY3 <= I(14) and z and T(5);

JMPZN1 <= I(14) and not z and T(3); JMPZN2 <= I(14) and not z and T(4);

JPNZY1 <= I(15) and not z and T(3); JPNZY2 <= I(15) and not z and T(4); JPNZY3 <= I(15) and not z and T(5);

JPNZN1 <= I(15) and z and T(3); JPNZN2 <= I(15) and z and T(4);

-- Counter control

clr_tc <= LDAC5 or STAC5 or JUMP3 or JMPZY3 or JPNZY3;

inc_tc <= not clr_tc;

-- =====

-- mOPs mapping

-- =====

mOPs(0) <= FETCH1 or FETCH3 or LDAC3 or STAC3; -- ARLOAD

mOPs(1) <= LDAC1 or STAC1 or JMPZY1 or JPNZY1; -- ARINC

mOPs(2) <= JUMP3 or JMPZY3 or JPNZY3; -- PCLOAD

mOPs(3) <= FETCH2 or LDAC1 or LDAC2 or STAC1 or STAC2 or JMPZN1 or JMPZN2 or JPNZN1 or JPNZN2; -- PCINC

mOPs(4) <= 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; -- DRLOAD

mOPs(5) <= LDAC2 or STAC2 or JUMP2 or JMPZY2 or JPNZY2; -- TRLOAD

mOPs(6) <= FETCH3; -- IRLOAD

mOPs(7) <= MVAC1; -- RLOAD

mOPs(8) <= LDAC5 or MVAC1 or ADD1 or SUB1 or INAC1 or CLAC1 or AND1 or OR1 or XOR1 or NOT1; -- ACLOAD

mOPs(9) <= LDAC5 or MVAC1 or ADD1 or SUB1 or INAC1 or CLAC1 or AND1 or OR1 or XOR1 or NOT1; -- ZLOAD

mOPs(10) <= FETCH2 or LDAC1 or LDAC2 or LDAC4 or STAC1 or STAC2 or JUMP1 or JUMP2 or JMPZY1 or JMPZY2 or JPNZY1 or JPNZY2; -- READ

mOPs(11) <= STAC5; -- WRITE

mOPs(12) <= FETCH2 or LDAC1 or LDAC2 or LDAC4 or STAC1 or STAC2 or JUMP1 or JUMP2 or JMPZY1 or JMPZY2 or JPNZY1 or JPNZY2; -- MEMBUS

mOPs(13) <= STAC5; -- BUSMEM

mOPs(14) <= FETCH1 or FETCH3; -- PCBUS

mOPs(15) <= LDAC2 or LDAC3 or LDAC5 or STAC2 or STAC3 or STAC5 or JUMP2 or JUMP3 or JMPZY2 or JMPZY3 or JPNZY2 or JPNZY3; -- DRBUS

mOPs(16) <= LDAC3 or STAC3 or JUMP3 or JMPZY3 or JPNZY3; -- TRBUS

mOPs(17) <= MVAC1 or ADD1 or SUB1 or AND1 or OR1 or XOR1; -- RBUS

mOPs(18) <= STAC4 or MVAC1; -- ACBUS

mOPs(19) <= AND1; -- ANDOP

mOPs(20) <= OR1; -- OROP

```

mOPs(21) <= XOR1; -- XOROP

mOPs(22) <= NOT1; -- NOTOP

mOPs(23) <= INAC1; -- ACINC

mOPs(24) <= CLAC1; -- ACZERO

mOPs(25) <= ADD1; -- PLUS

mOPs(26) <= SUB1; -- MINUS

end arc;

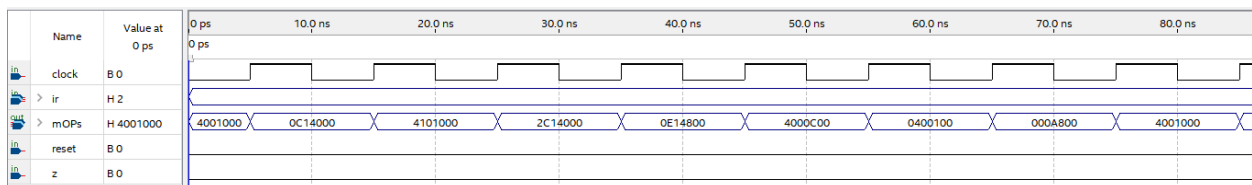
```

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:](#)

Εικόνα 2: Κυματομορφές εξομοίωσης της μονάδας ελέγχου