

# HARDWIRED

## ΜΟΝΑΔΑ ΕΛΕΓΧΟΥ

4

Ονοματεπώνυμο : Παναγιώτης Αναγνωστόπουλος

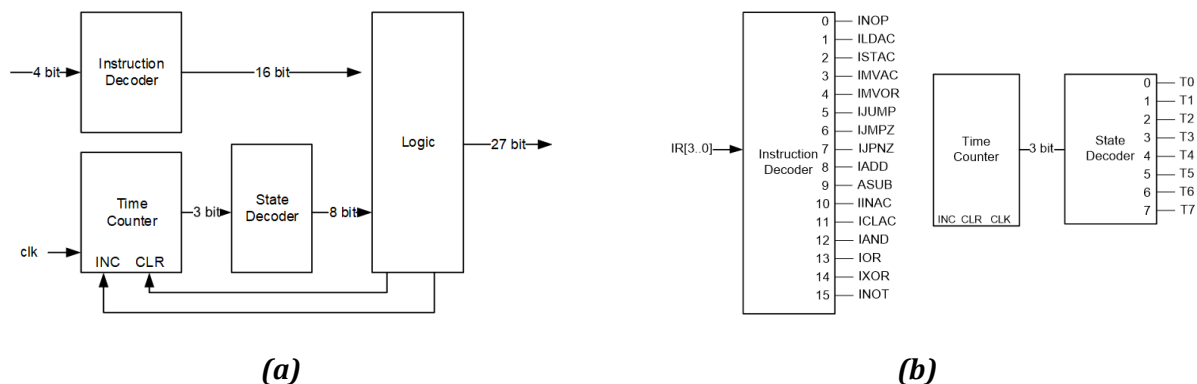
AM : 90930202100034

Ημερομηνία : 15/12/2025

### Έσκοπός

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).



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

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

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 συνδιαστική 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_dec is
```

```
port(
```

```
    din : in std_logic_vector(3 downto 0);
```

```
    dout : out std_logic_vector(15 downto 0)
```

```
);
```

```
end entity;
```

```
architecture rtl of instr_dec is
```

```
begin
```

```
    process(din)
```

```
    begin
```

```
        dout <= (others => '0');
```

```
        case din is
```

```
            when "0000" => dout(0) <= '1'; --Εντολή INOP
```

```
            when "0001" => dout(1) <= '1'; --Εντολή ILDAC
```

```
            when "0010" => dout(2) <= '1'; --Εντολή ISTAC
```

```

when "0011" => dout(3) <= '1'; --Εντολή IMVAC
when "0100" => dout(4) <= '1'; --Εντολή IMOVR
when "0101" => dout(5) <= '1'; --Εντολή IJUMP
when "0110" => dout(6) <= '1'; --Εντολή IJMPZ
when "0111" => dout(7) <= '1'; --Εντολή IJPNZ
when "1000" => dout(8) <= '1'; --Εντολή IADD
when "1001" => dout(9) <= '1'; --Εντολή SUB
when "1010" => dout(10) <= '1'; --Εντολή IINAC
when "1011" => dout(11) <= '1'; --Εντολή CLAC
when "1100" => dout(12) <= '1'; --Εντολή IAND
when "1101" => dout(13) <= '1'; --Εντολή IOR
when "1110" => dout(14) <= '1'; --Εντολή IXOR
when others => dout(15) <= '1'; --Εντολή INOT

end case;

end process;

end architecture;

```

**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_dec is

    port(
        din : in std_logic_vector(2 downto 0);
        dout : out std_logic_vector(7 downto 0)
    );
end entity state_dec;

```

**);**

**end entity;**

**architecture rtl of state\_dec is**

**begin**

**process(din)**

**begin**

**dout <= (others => '0');**

**case din is**

**when "000" => dout(0) <= '1'; -- Κατάσταση T0**

**when "001" => dout(1) <= '1'; -- Κατάσταση T1**

**when "010" => dout(2) <= '1'; -- Κατάσταση T2**

**when "011" => dout(3) <= '1'; -- Κατάσταση T3**

**when "100" => dout(4) <= '1'; -- Κατάσταση T4**

**when "101" => dout(5) <= '1'; -- Κατάσταση T5**

**when "110" => dout(6) <= '1'; -- Κατάσταση T6**

**when others=> dout(7) <= '1'; -- Κατάσταση T7**

**end case;**

**end process;**

**end architecture;**

**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 counter3_bit is
```

```
  port(
```

```
    clock : in std_logic;
```

```
    rst  : in std_logic;
```

```
    inc  : in std_logic;
```

```
    count : out std_logic_vector(2 downto 0)
```

```
  );
```

```
end entity;
```

```
architecture rtl of counter3_bit is
```

```
  signal c : std_logic_vector(2 downto 0);
```

```
begin
```

```
  process(clock)
```

```
  begin
```

```
    if rising_edge(clock) then
```

```
      if rst = '1' then
```

```
        c <= (others => '0');
```

```

    elsif inc = '1' then
        c <= c + "001";
    end if;
end if;
end process;

```

```

count <= c;
end architecture;

```

*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 instr_dec
    port(
        din : in std_logic_vector(3 downto 0);
        dout : out std_logic_vector(15 downto 0)
    );
end component;

```

```

    component state_dec

```



```

port(
    din : in std_logic_vector(2 downto 0);
    dout : out std_logic_vector(7 downto 0)
);
end component;

```

```

component counter3_bit
port(
    clock : in std_logic;
    rst : in std_logic;
    inc : in std_logic;
    count : out std_logic_vector(2 downto 0)
);
end component;

```

```

end package;

```

[package body hardwiredlib is](#)

[end package body;](#)

**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:

ΕΠΕΙΔΗ ΤΟ ΠΡΟΓΡΑΜΜΑ ΕΙΝΑΙ ΠΟΛΛΕΣ ΣΕΙΡΕΣ ΚΑΘΩΣ ΠΕΡΑΣΤΗΚΑΝΕ ΟΛΟΙ ΟΙ ΠΙΝΑΚΕΣ ΠΟΥ ΕΧΟΥΝ ΔΩΘΕΙ, ΟΙ ΚΑΤΑΣΤΑΣΕΙΣ, ΤΑ ΣΗΜΑΤΑ, ΟΙ ΣΤΑΘΕΡΕΣ ΜΕΤΑΒΛΗΤΕΣ ΚΛΠ. ΘΑ ΔΩΘΕΙ ΤΟ ΔΙΝΚ ΠΟΥ ΕΧΕΙ ΑΝΑΕΒΕΙ ΣΤΟ [GITHUB](#)

ΝΑ ΣΗΜΕΙΩΘΕΙ ΟΤΙ ΥΠΑΡΧΕΙ ΤΥΠΟΓΡΑΦΙΚΟ ΛΑΘΟΣ ΠΟΥ ΕΔΩ ΚΑΙ 2 ΕΒΔΟΜΑΔΕΣ ΔΕΝ ΕΧΕΙ ΔΙΟΡΘΩΘΕΙ.

ΠΑΡΑΤΗΡΗΘΗΚΕ ΑΣΥΝΕΠΙΑ ΣΤΗΝ ΑΝΤΙΣΤΟΙΧΙΣΗ ΤΩΝ ΚΑΤΑΣΤΑΣΕΩΝ FETCH :

ΣΤΟ ΚΕΙΜΕΝΟ ΟΡΙΖΕΤΑΙ ΟΤΙ Η FETCH1=T0, FETCH2=T1 και FETCH3=T2., ΕΝΩ ΣΤΟΝ ΠΙΝΑΚΑ 1 ΕΜΦΑΝΙΖΕΤΑΙ FETCH3=T3.

ΤΟ T3 ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ ΩΣ ΠΡΩΤΗ ΚΑΤΑΣΤΑΣΗ ΕΚΤΕΛΕΣΗΣ ΤΗΣ ΕΝΤΟΛΗΣ, ΣΥΝΕΠΩΣ ΤΟ FETCH3=T3 ΑΠΟΤΕΛΕΙ ΛΑΘΟΣ.

ΕΠΟΜΕΝΩΣ ΧΡΗΣΙΜΟΠΟΙΟΥΜΕ FETCH3=T2

```
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

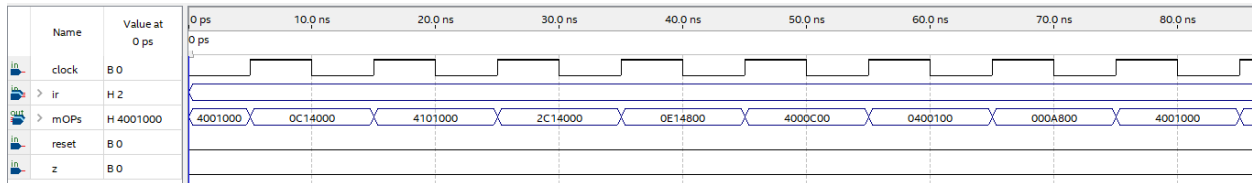
end arc;
```

**Πρόγραμμα 5:** Μονάδα Ελέγχου.

## Εξομοίωση της Μονάδας Ελέγχου.

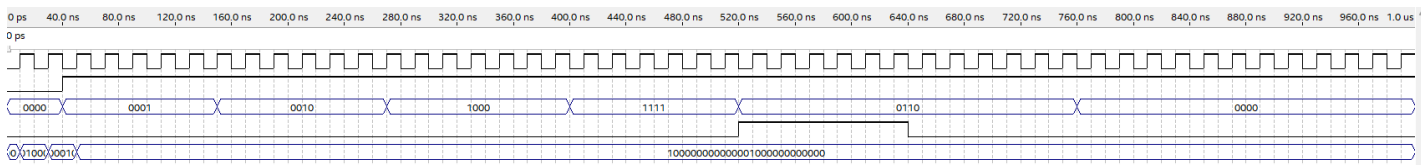
Το επόμενο στάδιο περιλαμβάνει την εξομοίωση της μονάδας ελέγχου με τον Waveform Editor με σκοπό τον έλεγχο της λειτουργίας της. Με οδηγό τις προηγούμενες ασκήσεις, δημιουργήστε ένα καινούργιο project και εξομοιώστε τη λειτουργία της μονάδας ελέγχου με τη βοήθεια του Waveform Editor για έξι (6) εντολές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

Στην Εικόνα 2 παρουσιάζονται οι κυματομορφές εξομοίωσης της hardwired μονάδας ελέγχου της ΚΜΕ. Η λειτουργία είναι η εξής :

- Αρχικά ενεργοποιείται το σήμα reset, ώστε το σύστημα να οδηγηθεί στην αρχική κατάσταση, και στη συνέχεια εφαρμόζονται έξι διαφορετικές εντολές μέσω του καταχωρητή εντολών ir[3..0].
- Για κάθε εντολή πραγματοποιείται διαδικασία FETCH, ακολουθούμενη από τις αντίστοιχες μικροκαταστάσεις εκτέλεσης. Τα σήματα ελέγχου mOPs[26..0] μεταβάλλεται σε κάθε μικροκατάσταση, ενεργοποιώντας τα κατάλληλα σήματα ελέγχου, όπως ανάλογα με την εκτελούμενη εντολή.
- Συνολικά, από τις κυματομορφές προκύπτει ότι η μονάδα ελέγχου λειτουργεί σύμφωνα με την προαναφερθείσα θεωρία και τους πίνακες που έχουν παραθετηθεί.