Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Central Processing Unit (CPU)

Anestopoulos Konstantinos, Undergraduate Student

Giorgos Vasileiou, Undergraduate Student

Avlonitis Konstantinos-Odusseas, Undergraduate Student

Department of Electrical and Computer Engineering, University of the Peloponnese

Υλοποιηθικε μια απλη ΚΜΕ με έναν επιπλεον καταχορητη B (8-bits) που εκτελει τις εντολες του παρακατω πινακα.

Implement a simple CPU with an additional 8-bit B register that executes the instructions in the table below.

Instruction	Instruction Code (Hex)	Instruction Code (Bin)	**Operation**
ANDB	1C	00011100	$AC \leftarrow AC \wedge B$
ORB	1D	00011101	$AC \leftarrow AC \vee B$
XORB	1E	00011110	$AC \leftarrow AC \oplus B$

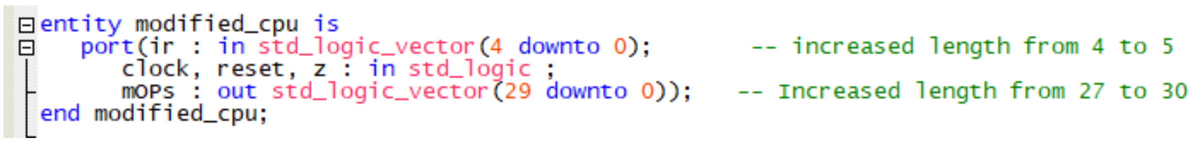
Για να υλοποιηθει η νεα ΚΜΕ,

Προκιμενου για να εκτελεστουν οι αριθμητικες ή λογικες εντολες του register B, επρεπε να γινουν αλλαγες σε καποια components.

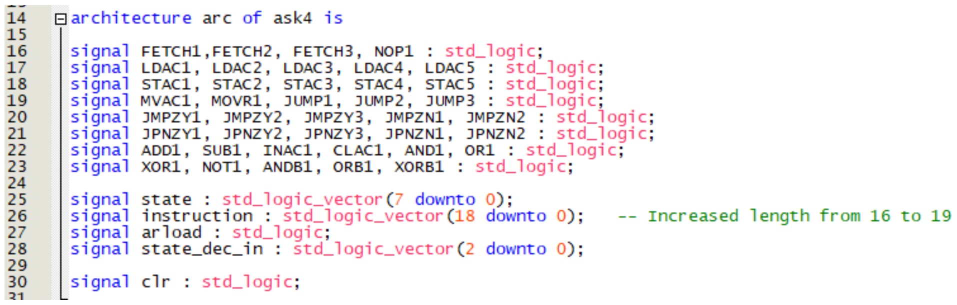
1. Στην hardwire προστεθικαν 3 επιπλεον bits στο mOPs και στο instraction, ώστε να μπορουν να αντιστιχιθουν οι εντολες ANDB,ORB,XORB, όπως φενεται παρακατω.

To implement the new CPU,
In order to execute the numerical and logical instructions in register B, several changes had to be made to several components.

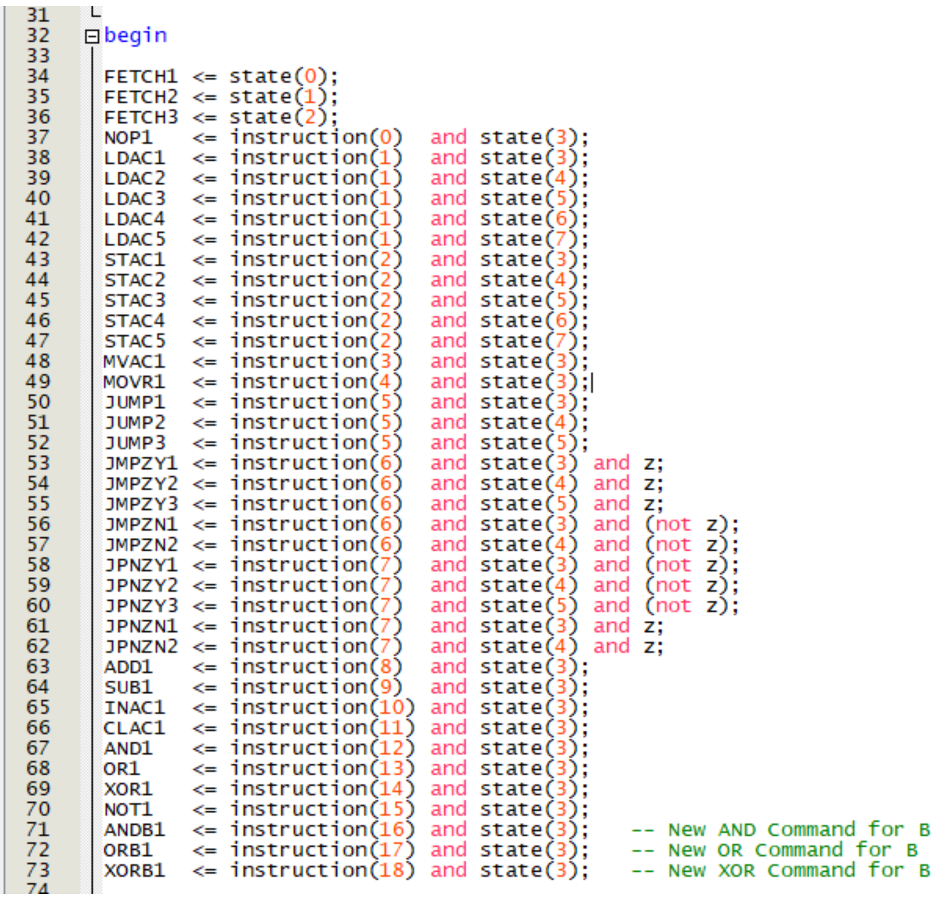
1. Hardwire added 3 bits to mOPS and instructions so that ANDB, ORB, and XORB instructions could be inverted.



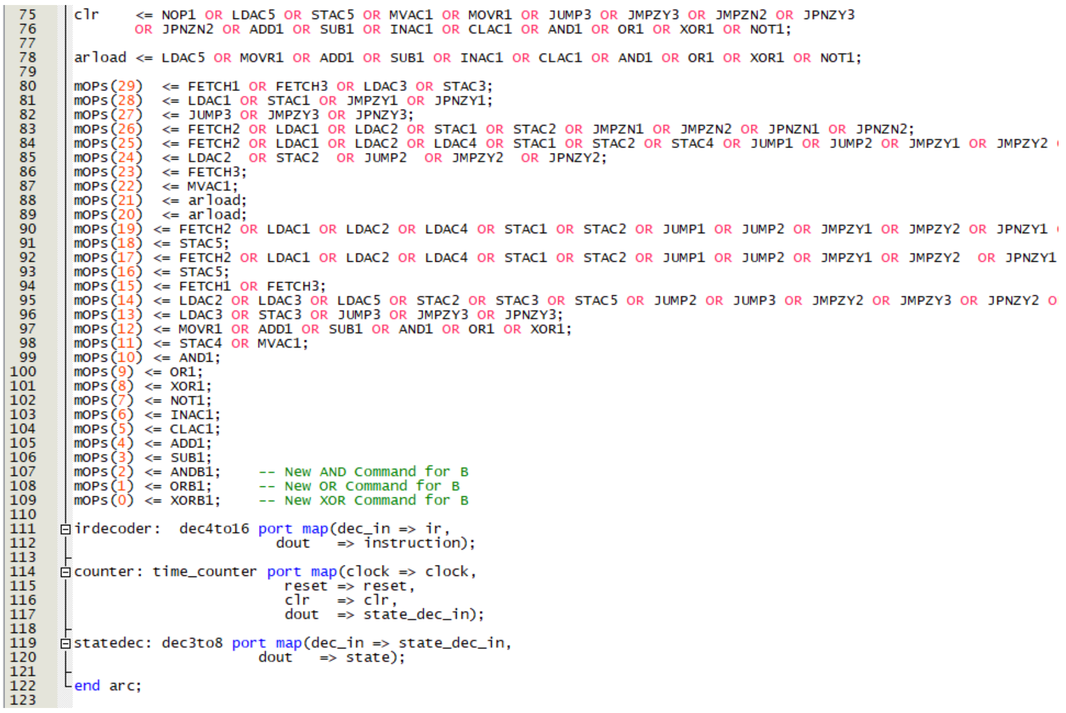
```
entity modified_cpu is
   port(ir : in std_logic_vector(4 downto 0);        -- increased length from 4 to 5
      clock, reset, z : in std_logic ;
      mOPs : out std_logic_vector(29 downto 0));    -- Increased length from 27 to 30
end modified_cpu;
```



```
architecture arc of ask4 is

signal FETCH1,FETCH2, FETCH3, NOP1 : std_logic;
signal LDAC1, LDAC2, LDAC3, LDAC4, LDAC5 : std_logic;
signal STAC1, STAC2, STAC3, STAC4, STAC5 : std_logic;
signal MVAC1, MOVR1, JUMP1, JUMP2, JUMP3 : std_logic;
signal JMPZY1, JMPZY2, JMPZY3, JMPZN1, JMPZN2 : std_logic;
signal JPNZY1, JPNZY2, JPNZY3, JPNZN1, JPNZN2 : std_logic;
signal ADD1, SUB1, INAC1, CLAC1, AND1, OR1 : std_logic;
signal XOR1, NOT1, ANDB1, ORB1, XORB1 : std_logic;

signal state : std_logic_vector(7 downto 0);
signal instruction : std_logic_vector(18 downto 0);   -- Increased length from 16 to 19
signal arload : std_logic;
signal state_dec_in : std_logic_vector(2 downto 0);

signal clr : std_logic;
```



```vhdl
begin

FETCH1 <= state(0);
FETCH2 <= state(1);
FETCH3 <= state(2);
NOP1   <= instruction(0)  and state(3);
LDAC1  <= instruction(1)  and state(3);
LDAC2  <= instruction(1)  and state(4);
LDAC3  <= instruction(1)  and state(5);
LDAC4  <= instruction(1)  and state(6);
LDAC5  <= instruction(1)  and state(7);
STAC1  <= instruction(2)  and state(3);
STAC2  <= instruction(2)  and state(4);
STAC3  <= instruction(2)  and state(5);
STAC4  <= instruction(2)  and state(6);
STAC5  <= instruction(2)  and state(7);
MVAC1  <= instruction(3)  and state(3);
MOVR1  <= instruction(4)  and state(3);
JUMP1  <= instruction(5)  and state(3);
JUMP2  <= instruction(5)  and state(4);
JUMP3  <= instruction(5)  and state(5);
JMPZY1 <= instruction(6)  and state(3) and z;
JMPZY2 <= instruction(6)  and state(4) and z;
JMPZY3 <= instruction(6)  and state(5) and z;
JMPZN1 <= instruction(6)  and state(3) and (not z);
JMPZN2 <= instruction(6)  and state(4) and (not z);
JPNZY1 <= instruction(7)  and state(3) and (not z);
JPNZY2 <= instruction(7)  and state(4) and (not z);
JPNZY3 <= instruction(7)  and state(5) and (not z);
JPNZN1 <= instruction(7)  and state(3) and z;
JPNZN2 <= instruction(7)  and state(4) and z;
ADD1   <= instruction(8)  and state(3);
SUB1   <= instruction(9)  and state(3);
INAC1  <= instruction(10) and state(3);
CLAC1  <= instruction(11) and state(3);
AND1   <= instruction(12) and state(3);
OR1    <= instruction(13) and state(3);
XOR1   <= instruction(14) and state(3);
NOT1   <= instruction(15) and state(3);
ANDB1  <= instruction(16) and state(3);    -- New AND Command for B
ORB1   <= instruction(17) and state(3);    -- New OR Command for B
XORB1  <= instruction(18) and state(3);    -- New XOR Command for B
```

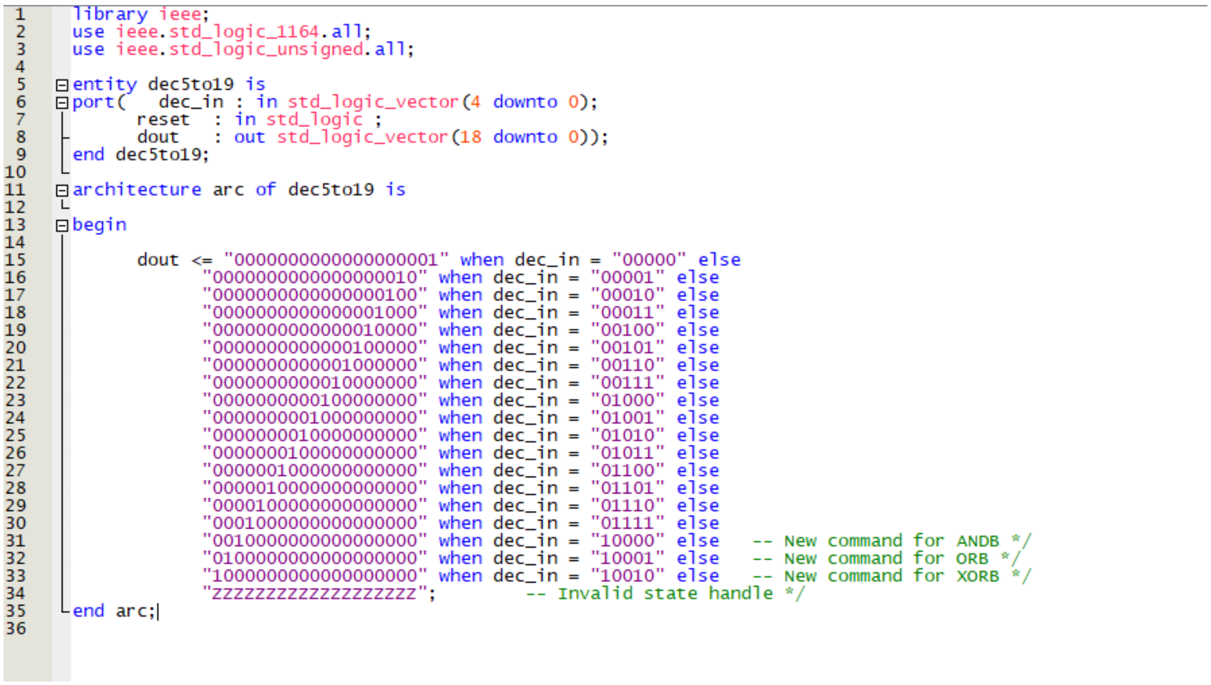


```vhdl
clr     <= NOP1 OR LDAC5 OR STAC5 OR MVAC1 OR MOVR1 OR JUMP3 OR JMPZY3 OR JMPZN2 OR JPNZY3
        OR JPNZN2 OR ADD1 OR SUB1 OR INAC1 OR CLAC1 OR AND1 OR OR1 OR XOR1 OR NOT1;

arload <= LDAC5 OR MOVR1 OR ADD1 OR SUB1 OR INAC1 OR CLAC1 OR AND1 OR OR1 OR XOR1 OR NOT1;

mOPs(29)  <= FETCH1 OR FETCH3 OR LDAC3 OR STAC3;
mOPs(28)  <= LDAC1 OR STAC1 OR JMPZY1 OR JPNZY1;
mOPs(27)  <= JUMP3 OR JMPZY3 OR JPNZY3;
mOPs(26)  <= FETCH2 OR LDAC1 OR LDAC2 OR STAC1 OR STAC2 OR JMPZN1 OR JMPZN2 OR JPNZN1 OR JPNZN2;
mOPs(25)  <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR STAC4 OR JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2
mOPs(24)  <= LDAC2  OR STAC2  OR JUMP2  OR JMPZY2  OR JPNZY2;
mOPs(23)  <= FETCH3;
mOPs(22)  <= MVAC1;
mOPs(21)  <= arload;
mOPs(20)  <= arload;
mOPs(19) <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2 OR JPNZY1
mOPs(18) <= STAC5;
mOPs(17) <= FETCH2 OR LDAC1 OR LDAC2 OR LDAC4 OR STAC1 OR STAC2 OR JUMP1 OR JUMP2 OR JMPZY1 OR JMPZY2  OR JPNZY1
mOPs(16) <= STAC5;
mOPs(15) <= FETCH1 OR FETCH3;
mOPs(14) <= LDAC2 OR LDAC3 OR LDAC5 OR STAC2 OR STAC3 OR STAC5 OR JUMP2 OR JUMP3 OR JMPZY2 OR JMPZY3 OR JPNZY2 O
mOPs(13) <= LDAC3 OR STAC3 OR JUMP3 OR JMPZY3 OR JPNZY3;
mOPs(12) <= MOVR1 OR ADD1 OR SUB1 OR AND1 OR OR1 OR XOR1;
mOPs(11) <= STAC4 OR MVAC1;
mOPs(10) <= AND1;
mOPs(9) <= OR1;
mOPs(8) <= XOR1;
mOPs(7) <= NOT1;
mOPs(6) <= INAC1;
mOPs(5) <= CLAC1;
mOPs(4) <= ADD1;
mOPs(3) <= SUB1;
mOPs(2) <= ANDB1;     -- New AND Command for B
mOPs(1) <= ORB1;      -- New OR Command for B
mOPs(0) <= XORB1;     -- New XOR Command for B

irdecoder:  dec4to16 port map(dec_in => ir,
                         dout   => instruction);

counter: time_counter port map(clock => clock,
                          reset => reset,
                          clr   => clr,
                          dout  => state_dec_in);

statedec: dec3to8 port map(dec_in => state_dec_in,
                        dout   => state);

end arc;
```

Hardwire

Χρησιμοποιηθικε Decoder 5 σε 19 αντι του 4 σε 16, για την καληψη των νεων εντολων.

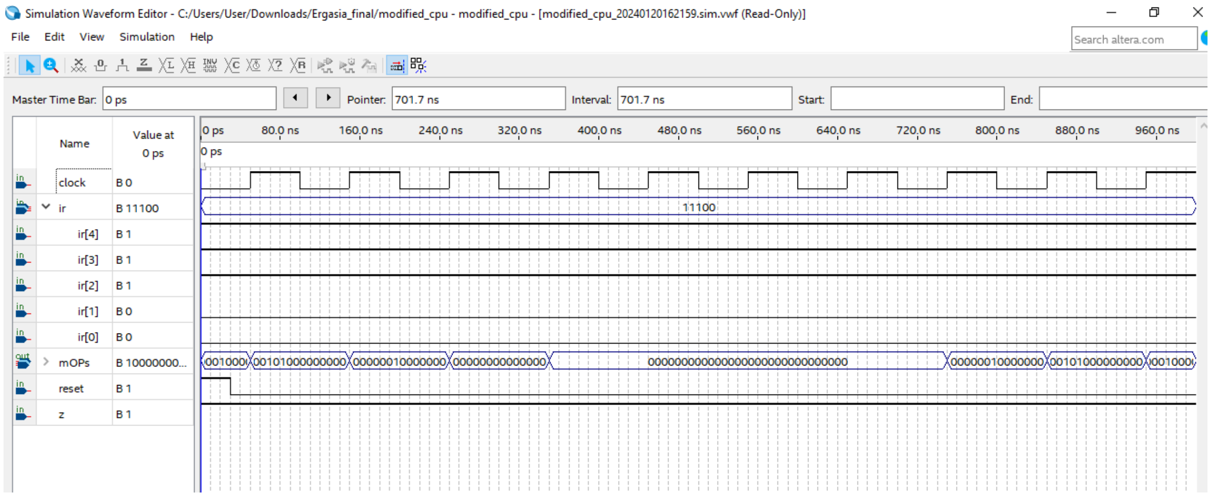
2. Use a 5/19 decoder instead of 4/16 to capture new commands.



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

entity dec5to19 is
port(   dec_in : in std_logic_vector(4 downto 0);
      reset   : in std_logic ;
      dout    : out std_logic_vector(18 downto 0));
end dec5to19;

architecture arc of dec5to19 is

begin

      dout <= "0000000000000000001" when dec_in = "00000" else
            "0000000000000000010" when dec_in = "00001" else
            "0000000000000000100" when dec_in = "00010" else
            "0000000000000001000" when dec_in = "00011" else
            "0000000000000010000" when dec_in = "00100" else
            "0000000000000100000" when dec_in = "00101" else
            "0000000000001000000" when dec_in = "00110" else
            "0000000000010000000" when dec_in = "00111" else
            "0000000000100000000" when dec_in = "01000" else
            "0000000001000000000" when dec_in = "01001" else
            "0000000010000000000" when dec_in = "01010" else
            "0000000100000000000" when dec_in = "01011" else
            "0000001000000000000" when dec_in = "01100" else
            "0000010000000000000" when dec_in = "01101" else
            "0000100000000000000" when dec_in = "01110" else
            "0001000000000000000" when dec_in = "01111" else
            "0010000000000000000" when dec_in = "10000" else   -- New command for ANDB */
            "0100000000000000000" when dec_in = "10001" else   -- New command for ORB */
            "1000000000000000000" when dec_in = "10010" else   -- New command for XORB */
            "ZZZZZZZZZZZZZZZZZZZ";        -- Invalid state handle */
end arc;
```

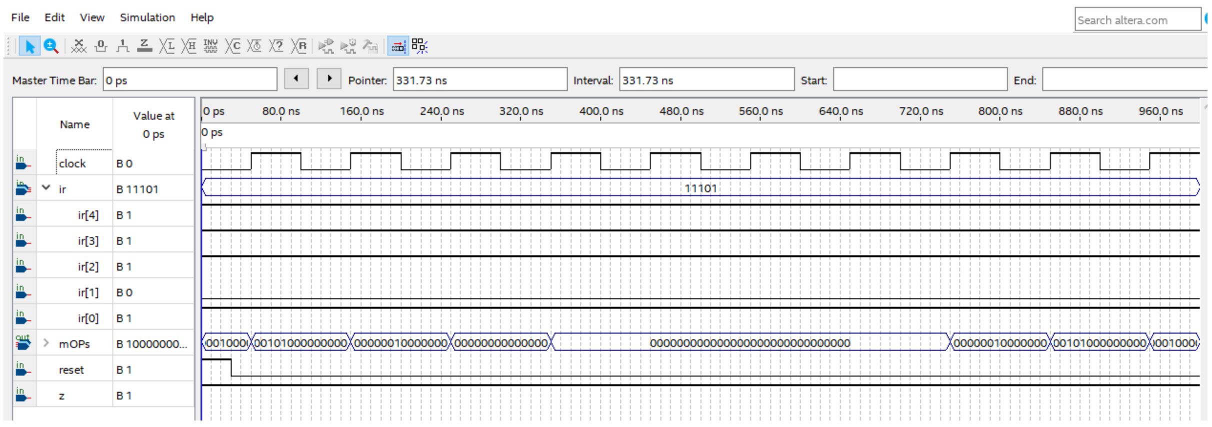
Simulation

- If ir = 11100



```
Simulation Waveform Editor - C:/Users/User/Downloads/Ergasia_final/modified_cpu - modified_cpu - [modified_cpu_20240120162159.sim.vwf (Read-Only)]

Master Time Bar: 0 ps    Pointer: 701.7 ns    Interval: 701.7 ns    Start:    End:

Name      Value at 0 ps
clock     B 0
ir        B 11100       11100
ir[4]     B 1
ir[3]     B 1
ir[2]     B 1
ir[1]     B 0
ir[0]     B 0
mOPs      B 10000000...  001000 | 00101000000000 | 00000010000000 | 00000000000000 | 00000000000000000000000000000 | 00000010000000 | 00101000000000 | 001000
reset     B 1
z         B 1
```

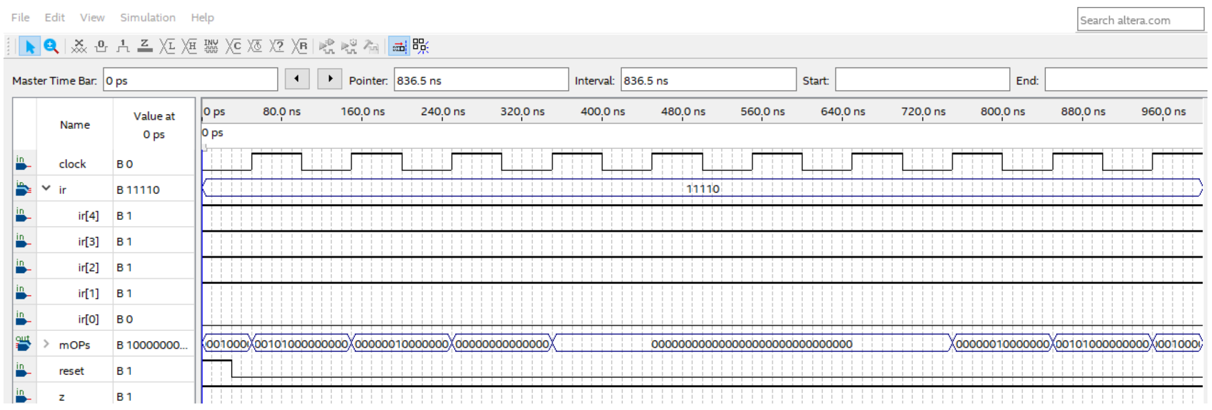
- If ir = 11101



Master Time Bar: 0 ps Pointer: 331.73 ns Interval: 331.73 ns Start: End:

Name	Value at 0 ps
clock	B 0
ir	B 11101
ir[4]	B 1
ir[3]	B 1
ir[2]	B 1
ir[1]	B 0
ir[0]	B 1
mOPs	B 10000000...
reset	B 1
z	B 1

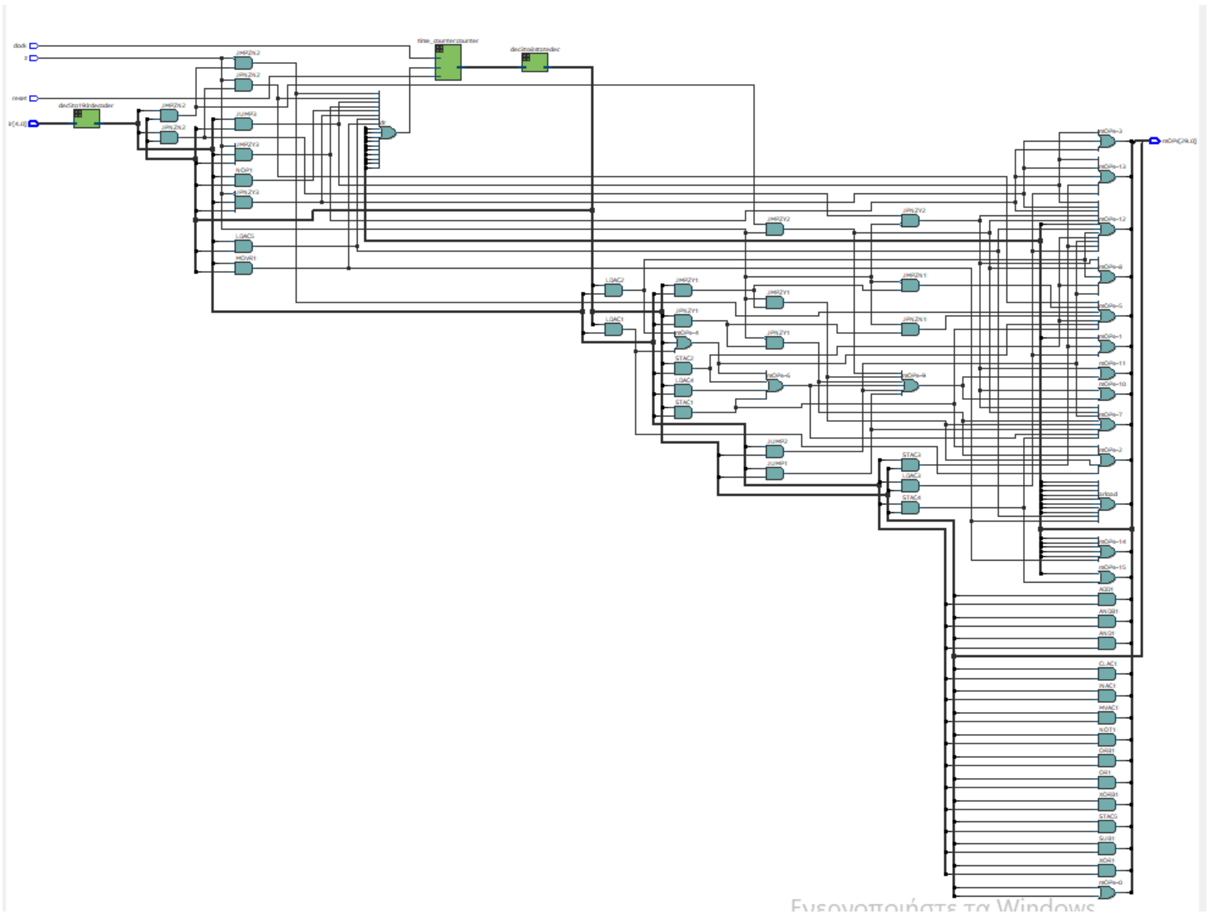
- If ir = 11110



Master Time Bar: 0 ps Pointer: 836.5 ns Interval: 836.5 ns Start: End:

Name	Value at 0 ps
clock	B 0
ir	B 11110
ir[4]	B 1
ir[3]	B 1
ir[2]	B 1
ir[1]	B 1
ir[0]	B 0
mOPs	B 10000000...
reset	B 1
z	B 1

RTL Diagram



clock
z
reset
ir[4..0]
time_counter:counter
mOPs[29..0]