Σαλής Χρήστος

Βοηθητική Εργασία σε vhdl για την Αρχιτεκτονική Υπολογιστών

http://arch.icte.uowm.gr

Στην εργασία αυτή, κατεβάζουμε τον επεξεργαστή picoblaze σε πλακέτα fpga και εκτυπώνουμε το μήνυμα hello world.

Θα πρέπει πρώτα να χρησιμοποιήσουμε σαν top module το αρχείο s3esk_startup.vhd, στο οποίο θα περιέχονται τα αρχεία control.vhd, kcpsm3.vhd και s3esk_startup.ucf. Στην αρχή θα πρέπει να δημιουργηθεί το αρχείο control.vhd, που θα περιέχει και την εκτύπωση μηνύματος. Θα πρέπει να αλλάξουμε τον κώδικα στο control.psm, για να εκτυπώνει το μήνυμα που θέλουμε.

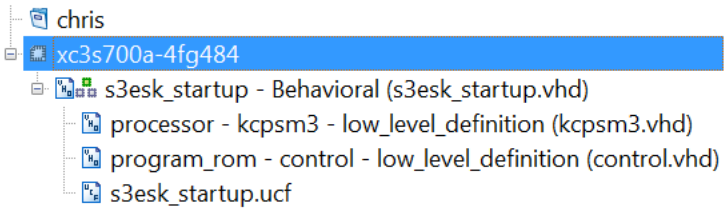
```
disp_SPARTAN: LOAD s5, character_H
              CALL LCD_write_data
              LOAD s5, character_e
              CALL LCD_write_data
              LOAD s5, character_l
              CALL LCD_write_data
              LOAD s5, character_l
              CALL LCD_write_data
              LOAD S5, character_o
              CALL disp_space
              LOAD s5, character_w
              CALL LCD_write_data
              LOAD s5, character_o
              CALL LCD_write_data
              LOAD s5, character_r
              CALL LCD_write_data
              LOAD s5, character_l
              CALL LCD_write_data
              LOAD s5, character_d
              CALL disp_space
              RETURN
```

Μετά στο τερματικό τρέχουμε την εντολή KCPSM3 control.psm

```
C:\Users\8A70~1\DOCUME~1\fpga2\ASSEMB~1>KCPSM3 control.psm
```

```
PASS 7 - Writing coefficient file
          control.coe

PASS 8 - Writing VHDL memory definition file
          control.vhd

PASS 9 - Writing Verilog memory definition file
          control.v

PASS 10 - Writing System Generator memory definition file
          control.m

PASS 11 - Writing memory definition files
          control.hex
          control.dec
          control.mem


KCPSM3 successful.

KCPSM3 complete.
```

Έτσι αφού παραχθεί το control.vhd, το προσθέτουμε στο s3esk_startup.vhd μαζί με τα άλλα αρχεία.



Επίσης, θα πρέπει να κάνουμε αλλαγή στον κώδικα του s3esk_startup.vhd. Πρέπει να αφαιρέσουμε την έξοδο proc_reset, που έχει να κάνει σχέση με τον JTAG loader, (αυτό συμβαίνει επειδή κάναμε reassemble το αρχείο psm). Έτσι, στο σήμα kcpsm3_reset, δίνουμε 0.

```
port map(      address => address,
           instruction => instruction,
           -- proc_reset => kcpsm3_reset,            --JTAG Loader
                 clk => clk);
              kcpsm3_reset<= '0';
```

Αφού κάνουμε και τις σωστές αλλαγές και στο αρχείο ucf, για να λειτουργεί στη πλακέτα Spartan 3A.

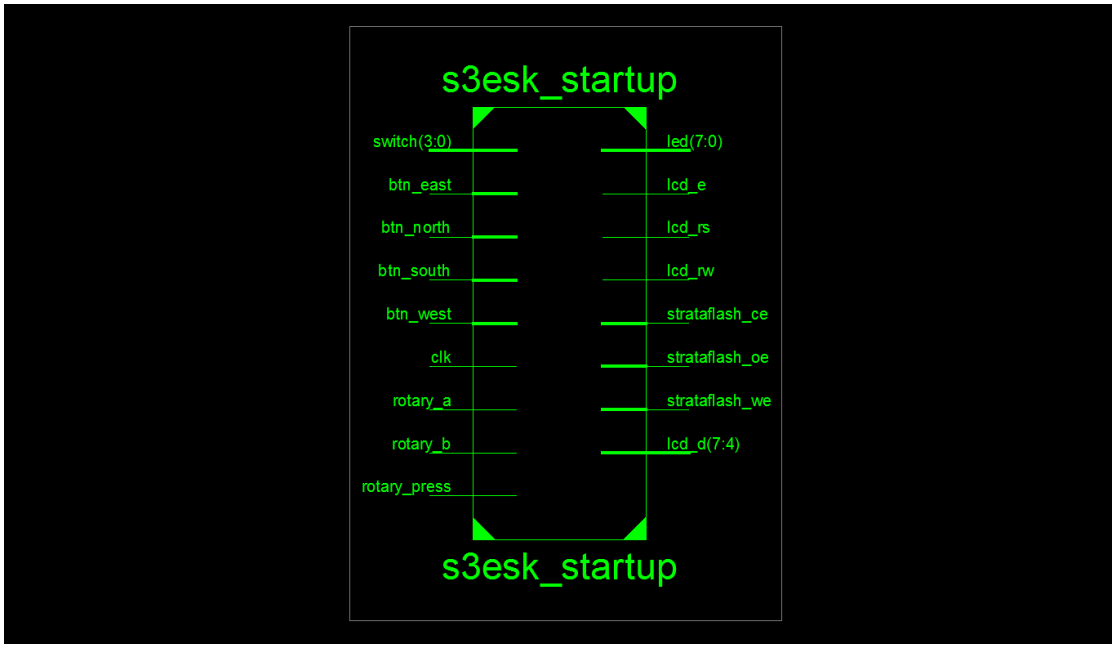
```
NET "clk" PERIOD = 20.0ns HIGH 50%;
#
# soldered 50MHz Clock.
#
NET "clk" LOC = "E12" | IOSTANDARD = LVTTL;
#
#
# Simple LEDs
# Require only 3.5mA.
#
NET "led<0>" LOC = "R20" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<1>" LOC = "T19" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<2>" LOC = "U20" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<3>" LOC = "U19" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<4>" LOC = "V19" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<5>" LOC = "V20" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<6>" LOC = "Y22"  | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
NET "led<7>" LOC = "W21"  | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 4;
#
#
# LCD display
# Very slow so can use lowest drive strength.
#
NET "lcd_rs"   LOC = "Y14" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_rw"   LOC = "W13" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
```

```
NET "lcd_rw"   LOC = "W13" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_e"    LOC = "AB4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<4>" LOC = "AA12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<5>" LOC = "Y16" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<6>" LOC = "AB16" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<7>" LOC = "Y15" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
#
# Strata Flash (need to disable to use LCD display)
#
NET "strataflash_oe" LOC = "C18" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "strataflash_ce" LOC = "D16" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
NET "strataflash_we" LOC = "D17" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 2;
#
#
# Simple switches
#   Pull UP resistors used to stop floating condition during switching.
#
NET "switch<0>" LOC = "V8" | IOSTANDARD = LVTTL | PULLUP;
NET "switch<1>" LOC = "U10" | IOSTANDARD = LVTTL | PULLUP;
NET "switch<2>" LOC = "U8" | IOSTANDARD = LVTTL | PULLUP;
NET "switch<3>" LOC = "T9" | IOSTANDARD = LVTTL | PULLUP;
#
#
# Press buttons
#   Must have pull DOWN resistors to provide Low when not pressed.
```

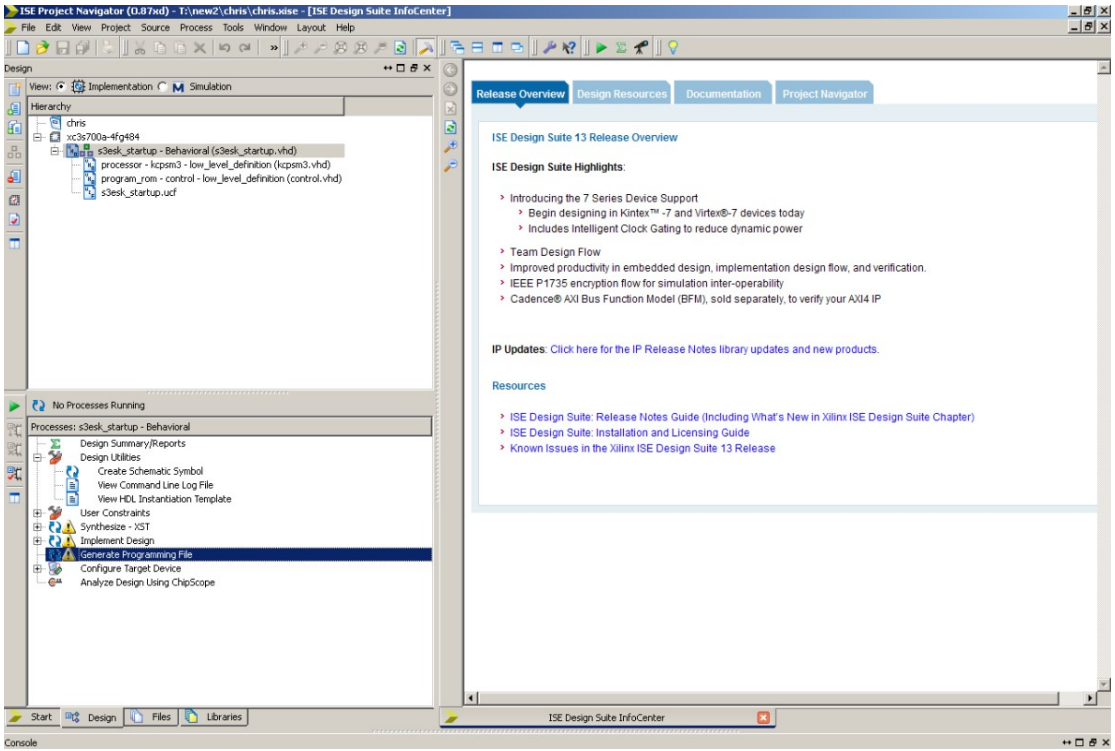
```
NET "btn_north" LOC = "T14"  | IOSTANDARD = LVTTL | PULLDOWN;
NET "btn_east"  LOC = "T16" | IOSTANDARD = LVTTL | PULLDOWN;
NET "btn_south" LOC = "T15" | IOSTANDARD = LVTTL | PULLDOWN;
NET "btn_west"  LOC = "U15" | IOSTANDARD = LVTTL | PULLDOWN;
#
# Rotary encoder.
#   Rotation contacts require pull UP resistors to provide High level.
#   Press contact requires pull DOWN resistor to provide Low when not pressed..
#
NET "rotary_a"     LOC = "T13" | IOSTANDARD = LVTTL | PULLUP;
NET "rotary_b"     LOC = "R14" | IOSTANDARD = LVTTL | PULLUP;
NET "rotary_press" LOC = "R13" | IOSTANDARD = LVTTL | PULLDOWN;
```

Έτσι, αφού κάνουμε generate το bitstream μπορούμε να το κατεβάσουμε με το impact στη πλακέτα και να εκτυπώσουμε το μήνυμα στην οθόνη lcd της πλακέτας.

Το schematic του κυκλώματος φαίνεται στη παρακάτω εικόνα



Εδώ μπορούμε να δούμε ότι δημιουργήθηκε το bitstream.



Ένα κομμάτι κώδικα σε vhdl του αρχείου kcpsm3.vhd

```
64  -- Standard IEEE libraries
65  --
66  library IEEE;
67  use IEEE.STD_LOGIC_1164.ALL;
68  use IEEE.STD_LOGIC_ARITH.ALL;
69  use IEEE.STD_LOGIC_UNSIGNED.ALL;
70  library unisim;
71  use unisim.vcomponents.all;
72  --
73  ----------------------------------------------------------------------------------
74  --
75  -- Main Entity for KCPSM3
76  --
77  entity kcpsm3 is
78      Port (      address : out std_logic_vector(9 downto 0);
79              instruction : in std_logic_vector(17 downto 0);
80                  port_id : out std_logic_vector(7 downto 0);
81             write_strobe : out std_logic;
82                 out_port : out std_logic_vector(7 downto 0);
83             read_strobe : out std_logic;
84                 in_port : in std_logic_vector(7 downto 0);
85               interrupt : in std_logic;
86           interrupt_ack : out std_logic;
87                   reset : in std_logic;
88                     clk : in std_logic);
89      end kcpsm3;
90  --
91  ----------------------------------------------------------------------------------
92  --
93  -- Start of Main Architecture for KCPSM3
94  --
95
96  architecture low_level_definition of kcpsm3 is
97  --
98  ----------------------------------------------------------------------------------
99  --
100 -- Signals used in KCPSM3
101 --
102 ----------------------------------------------------------------------------------
103 --
104 -- Fundamental control and decode signals
105 --
106 signal t_state                : std_logic;
107 signal not_t_state            : std_logic;
108 signal internal_reset         : std_logic;
109 signal reset_delay            : std_logic;
110 signal move_group             : std_logic;
```

```
70  --
71  architecture Behavioral of s3esk_startup is
72  --
73  ----------------------------------------------------------------------------------
74  --
75  -- declaration of KCPSM3
76  --
77    component kcpsm3
78      Port (      address : out std_logic_vector(9 downto 0);
79              instruction : in std_logic_vector(17 downto 0);
80                  port_id : out std_logic_vector(7 downto 0);
81             write_strobe : out std_logic;
82                 out_port : out std_logic_vector(7 downto 0);
83             read_strobe : out std_logic;
84                 in_port : in std_logic_vector(7 downto 0);
85               interrupt : in std_logic;
86           interrupt_ack : out std_logic;
87                   reset : in std_logic;
88                     clk : in std_logic);
89      end component;
90  --
91  -- declaration of program ROM
92  --
93    component control
94      Port (      address : in std_logic_vector(9 downto 0);
95              instruction : out std_logic_vector(17 downto 0);
96               --proc_reset : out std_logic;                        --JTAG Loader version
97                     clk : in std_logic);
98      end component;
99  --
100 ----------------------------------------------------------------------------------
101 --
102 -- Signals used to connect KCPSM3 to program ROM and I/O logic
103 --
104 signal address          : std_logic_vector(9 downto 0);
105 signal instruction      : std_logic_vector(17 downto 0);
106 signal port_id          : std_logic_vector(7 downto 0);
107 signal out_port         : std_logic_vector(7 downto 0);
108 signal in_port          : std_logic_vector(7 downto 0);
109 signal write_strobe     : std_logic;
110 signal read_strobe      : std_logic;
111 signal interrupt        : std_logic :='0';
112 signal interrupt_ack    : std_logic;
113 signal kcpsm3_reset     : std_logic;
114 --
115 --
116 -- Signals for LCD operation
```

```
--
-- Signals used to interface to rotary encoder
--
signal      rotary_a_in : std_logic;
signal      rotary_b_in : std_logic;
signal  rotary_press_in : std_logic;
signal        rotary_in : std_logic_vector(1 downto 0);
signal        rotary_q1 : std_logic;
signal        rotary_q2 : std_logic;
signal  delay_rotary_q1 : std_logic;
signal     rotary_event : std_logic;
signal      rotary_left : std_logic;
--
--
------------------------------------------------------------------------------------------------------------------------
--
-- Start of circuit description
--
begin
  --
  ----------------------------------------------------------------------------------------------------------------------
  -- Disable unused components
  ----------------------------------------------------------------------------------------------------------------------
  --
  --StrataFLASH must be disabled to prevent it conflicting with the LCD display
  --
  strataflash_oe <= '1';
  strataflash_ce <= '1';
  strataflash_we <= '1';
  --
  --
  ----------------------------------------------------------------------------------------------------------------------
  -- KCPSM3 and the program memory
  ----------------------------------------------------------------------------------------------------------------------
  --

  processor: kcpsm3
    port map(      address => address,
               instruction => instruction,
                   port_id => port_id,
              write_strobe => write_strobe,
                  out_port => out_port,
               read_strobe => read_strobe,
                   in_port => in_port,
                 interrupt => interrupt,
             interrupt_ack => interrupt_ack,
                     reset => kcpsm3_reset,
```

```
-- Data Register signals
--
signal sx                       : std_logic_vector(7 downto 0);
signal sy                       : std_logic_vector(7 downto 0);
signal register_type            : std_logic;
signal register_write           : std_logic;
signal register_enable          : std_logic;
signal second_operand           : std_logic_vector(7 downto 0);
--
-- Scratch Pad Memory signals
--
signal memory_data              : std_logic_vector(7 downto 0);
signal store_data               : std_logic_vector(7 downto 0);
signal memory_type              : std_logic;
signal memory_write             : std_logic;
signal memory_enable            : std_logic;
--
-- Stack signals
--
signal stack_pop_data           : std_logic_vector(9 downto 0);
signal stack_ram_data           : std_logic_vector(9 downto 0);
signal stack_address            : std_logic_vector(4 downto 0);
signal half_stack_address       : std_logic_vector(4 downto 0);
signal stack_address_carry      : std_logic_vector(3 downto 0);
signal next_stack_address       : std_logic_vector(4 downto 0);
signal stack_write_enable       : std_logic;
signal not_active_interrupt     : std_logic;
--
-- ALU signals
--
signal logical_result           : std_logic_vector(7 downto 0);
signal logical_value            : std_logic_vector(7 downto 0);
signal sel_logical              : std_logic;
signal shift_result             : std_logic_vector(7 downto 0);
signal shift_value              : std_logic_vector(7 downto 0);
signal sel_shift                : std_logic;
signal high_shift_in            : std_logic;
signal low_shift_in             : std_logic;
signal shift_in                 : std_logic;
signal shift_carry              : std_logic;
signal shift_carry_value        : std_logic;
signal arith_result             : std_logic_vector(7 downto 0);
signal arith_value              : std_logic_vector(7 downto 0);
signal half_arith               : std_logic_vector(7 downto 0);
signal arith_internal_carry     : std_logic_vector(7 downto 0);
signal sel_arith_carry_in       : std_logic;
signal arith_carry_in           : std_logic;
```

```
--
-- Fundamental Control
--
-- Definition of T-state and internal reset
--
-------------------------------------------------------------------------------
--
  t_state_lut: LUT1
  --synthesis translate_off
    generic map (INIT => X"1")
  --synthesis translate_on
  port map( I0 => t_state,
            O => not_t_state );

  toggle_flop: FDR
  port map ( D => not_t_state,
             Q => t_state,
             R => internal_reset,
             C => clk);

  reset_flop1: FDS
  port map ( D => '0',
             Q => reset_delay,
             S => reset,
             C => clk);

  reset_flop2: FDS
  port map ( D => reset_delay,
             Q => internal_reset,
             S => reset,
             C => clk);
--
-------------------------------------------------------------------------------
--
-- Interrupt input logic, Interrupt enable and shadow Flags.
--
-- Captures interrupt input and enables the shadow flags.
-- Decodes instructions which set and reset the interrupt enable flip-flop.
--
-------------------------------------------------------------------------------
--

  -- Interrupt capture

  int_capture_flop: FDR
  port map ( D => interrupt,
             Q => clean_int,
             R => internal_reset
```

```
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => int_update_enable );

  int_value_lut: LUT3
  --synthesis translate_off
    generic map (INIT => X"04")
  --synthesis translate_on
  port map( I0 => active_interrupt,
            I1 => instruction(0),
            I2 => interrupt_ack_internal,
             O => int_enable_value );

  int_enable_flop: FDRE
  port map ( D => int_enable_value,
             Q => int_enable,
            CE => int_update_enable,
             R => internal_reset,
             C => clk);
--
-------------------------------------------------------------------------------
--
-- Decodes for the control of the program counter and CALL/RETURN stack
--
-------------------------------------------------------------------------------
--
  move_group_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"7400")
  --synthesis translate_on
  port map( I0 => instruction(14),
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => move_group );

  condition_met_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"5A3C")
  --synthesis translate_on
  port map( I0 => carry_flag,
            I1 => zero_flag,
            I2 => instruction(10),
            I3 => instruction(11),
             O => condition_met );

  normal_count_lut: LUT3
```

```
  call_type_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"1000")
  --synthesis translate_on
  port map( I0 => instruction(14),
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => call_type );

  push_pop_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"5400")
  --synthesis translate_on
  port map( I0 => instruction(14),
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => push_or_pop_type );

  valid_move_lut: LUT2
  --synthesis translate_off
    generic map (INIT => X"D")
  --synthesis translate_on
  port map( I0 => instruction(12),
            I1 => condition_met,
             O => valid_to_move );
--
-------------------------------------------------------------------------------
--
-- The ZERO and CARRY Flags
--
-------------------------------------------------------------------------------
--
  -- Enable for flags

  flag_type_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"41FC")
  --synthesis translate_on
  port map( I0 => instruction(14),
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => flag_type );

  flag_write_flop: FD
```

```
             Q => flag_write,
             C => clk);

  flag_enable_lut: LUT2
  --synthesis translate_off
    generic map (INIT => X"8")
  --synthesis translate_on
  port map( I0 => t_state,
            I1 => flag_write,
             O => flag_enable );

  -- Zero Flag

  low_zero_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"0001")
  --synthesis translate_on
  port map( I0 => alu_result(0),
            I1 => alu_result(1),
            I2 => alu_result(2),
            I3 => alu_result(3),
             O => low_zero );

  high_zero_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"0001")
  --synthesis translate_on
  port map( I0 => alu_result(4),
            I1 => alu_result(5),
            I2 => alu_result(6),
            I3 => alu_result(7),
             O => high_zero );

  low_zero_muxcy: MUXCY
  port map( DI => '0',
            CI => '1',
             S => low_zero,
             O => low_zero_carry );

  high_zero_cymux: MUXCY
  port map( DI => '0',
            CI => low_zero_carry,
             S => high_zero,
             O => high_zero_carry );

  sel_shadow_zero_lut: LUT3
  --synthesis translate_off
    generic map (INIT => X"3F")
```

```
    generic map (INIT => X"3F")
  --synthesis translate_on
  port map( I0 => shadow_zero,
            I1 => instruction(16),
            I2 => instruction(17),
             O => sel_shadow_zero );

  zero_cymux: MUXCY
  port map( DI => shadow_zero,
            CI => high_zero_carry,
             S => sel_shadow_zero,
             O => zero_carry );

  zero_xor: XORCY
  port map( LI => '0',
            CI => zero_carry,
             O => zero_fast_route);

  zero_flag_flop: FDRE
  port map ( D => zero_fast_route,
             Q => zero_flag,
            CE => flag_enable,
             R => internal_reset,
             C => clk);

  -- Parity detection

  low_parity_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"6996")
  --synthesis translate_on
  port map( I0 => logical_result(0),
            I1 => logical_result(1),
            I2 => logical_result(2),
            I3 => logical_result(3),
             O => low_parity );

  high_parity_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"6996")
  --synthesis translate_on
  port map( I0 => logical_result(4),
            I1 => logical_result(5),
            I2 => logical_result(6),
            I3 => logical_result(7),
             O => high_parity );

  parity_muxcy: MUXCY
```

```
            I1 => logical_result(5),
            I2 => logical_result(6),
            I3 => logical_result(7),
             O => high_parity );

  parity_muxcy: MUXCY
  port map( DI => '0',
            CI => '1',
             S => low_parity,
             O => parity_carry );

  parity_xor: XORCY
  port map( LI => high_parity,
            CI => parity_carry,
             O => parity);

  -- CARRY flag selection

  sel_parity_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"F3FF")
  --synthesis translate_on
  port map( I0 => parity,
            I1 => instruction(13),
            I2 => instruction(15),
            I3 => instruction(16),
             O => sel_parity );

  sel_arith_carry_lut: LUT3
  --synthesis translate_off
    generic map (INIT => X"F3")
  --synthesis translate_on
  port map( I0 => arith_carry,
            I1 => instruction(16),
            I2 => instruction(17),
             O => sel_arith_carry );

  sel_shift_carry_lut: LUT2
  --synthesis translate_off
    generic map (INIT => X"C")
  --synthesis translate_on
  port map( I0 => shift_carry,
            I1 => instruction(15),
             O => sel_shift_carry );

  sel_shadow_carry_lut: LUT2
  --synthesis translate_off
```

```
570             I1 => logical_result(5),
571             I2 => logical_result(6),
572             I3 => logical_result(7),
573              O => high_parity );
574
575   parity_muxcy: MUXCY
576   port map( DI => '0',
577             CI => '1',
578              S => low_parity,
579              O => parity_carry );
580
581   parity_xor: XORCY
582   port map( LI => high_parity,
583             CI => parity_carry,
584              O => parity);
585
586   -- CARRY flag selection
587
588   sel_parity_lut: LUT4
589   --synthesis translate_off
590     generic map (INIT => X"F3FF")
591   --synthesis translate_on
592   port map( I0 => parity,
593             I1 => instruction(13),
594             I2 => instruction(15),
595             I3 => instruction(16),
596              O => sel_parity );
597
598   sel_arith_carry_lut: LUT3
599   --synthesis translate_off
600     generic map (INIT => X"F3")
601   --synthesis translate_on
602   port map( I0 => arith_carry,
603             I1 => instruction(16),
604             I2 => instruction(17),
605              O => sel_arith_carry );
606
607   sel_shift_carry_lut: LUT2
608   --synthesis translate_off
609     generic map (INIT => X"C")
610   --synthesis translate_on
611   port map( I0 => shift_carry,
612             I1 => instruction(15),
613              O => sel_shift_carry );
614
615   sel_shadow_carry_lut: LUT2
616   --synthesis translate_off
617     generic map (INIT => X"3")
```

```
638              S => sel_arith_carry,
639              O => sel_carry(2) );
640
641   sel_parity_muxcy: MUXCY
642   port map( DI => parity,
643             CI => sel_carry(2),
644              S => sel_parity,
645              O => sel_carry(3) );
646
647   carry_xor: XORCY
648   port map( LI => '0',
649             CI => sel_carry(3),
650              O => carry_fast_route);
651
652   carry_flag_flop: FDRE
653   port map ( D => carry_fast_route,
654              Q => carry_flag,
655             CE => flag_enable,
656              R => internal_reset,
657              C => clk);
658 --
659 ------------------------------------------------------------------------------
660 --
661 -- The Program Counter
662 --
663 -- Definition of a 10-bit counter which can be loaded from two sources
664 --
665 ------------------------------------------------------------------------------
666 --
667
668   invert_enable: INV   -- Inverter should be implemented in the CE to flip flops
669   port map(  I => t_state,
670              O => pc_enable);
671
672   pc_loop: for i in 0 to 9 generate
673   --
674   -- Attribute to define LUT contents during implementation
675   -- The information is repeated in the generic map for functional simulation
676   --
677   attribute INIT : string;
678   attribute INIT of vector_select_mux : label is "E4";
679   attribute INIT of value_select_mux  : label is "E4";
680   --
681   begin
682
683     vector_select_mux: LUT3
684     --synthesis translate_off
685       generic map (INIT => X"E4")
```

```
677   attribute INIT : string;
678   attribute INIT of vector_select_mux : label is "E4";
679   attribute INIT of value_select_mux  : label is "E4";
680   --
681   begin
682
683     vector_select_mux: LUT3
684     --synthesis translate_off
685       generic map (INIT => X"E4")
686     --synthesis translate_on
687     port map( I0 => instruction(15),
688               I1 => instruction(i),
689               I2 => stack_pop_data(i),
690                O => pc_vector(i) );
691
692     value_select_mux: LUT3
693     --synthesis translate_off
694       generic map (INIT => X"E4")
695     --synthesis translate_on
696     port map( I0 => normal_count,
697               I1 => inc_pc_vector(i),
698               I2 => pc(i),
699                O => pc_value(i) );
700
701      register_bit: FDRSE
702      port map ( D => inc_pc_value(i),
703                 Q => pc(i),
704                 R => internal_reset,
705                 S => active_interrupt,
706                CE => pc_enable,
707                 C => clk);
708
709      pc_lsb_carry: if i=0 generate
710        begin
711
712          pc_vector_muxcy: MUXCY
713          port map( DI => '0',
714                    CI => instruction(13),
715                     S => pc_vector(i),
716                     O => pc_vector_carry(i));
717
718          pc_vector_xor: XORCY
719          port map( LI => pc_vector(i),
720                    CI => instruction(13),
721                     O => inc_pc_vector(i));
722
723          pc_value_muxcy: MUXCY
724          port map( DI => '0',
725                    CI => normal_count,
726                     S => pc_value(i),
727                     O => pc_value_carry(i));
728
729          pc_value_xor: XORCY
730          port map( LI => pc_value(i),
731                    CI => normal_count,
732                     O => inc_pc_value(i));
733
734        end generate pc_lsb_carry;
735
736      pc_mid_carry: if i>0 and i<9 generate
737     begin
738
739          pc_vector_muxcy: MUXCY
740          port map( DI => '0',
741                    CI => pc_vector_carry(i-1),
742                     S => pc_vector(i),
743                     O => pc_vector_carry(i));
744
745          pc_vector_xor: XORCY
746          port map( LI => pc_vector(i),
747                    CI => pc_vector_carry(i-1),
748                     O => inc_pc_vector(i));
749
750          pc_value_muxcy: MUXCY
751          port map( DI => '0',
752                    CI => pc_value_carry(i-1),
753                     S => pc_value(i),
754                     O => pc_value_carry(i));
755
756          pc_value_xor: XORCY
757          port map( LI => pc_value(i),
758                    CI => pc_value_carry(i-1),
759                     O => inc_pc_value(i));
760
761        end generate pc_mid_carry;
762
763      pc_msb_carry: if i=9 generate
764        begin
765
766          pc_vector_xor: XORCY
767          port map( LI => pc_vector(i),
768                    CI => pc_vector_carry(i-1),
769                     O => inc_pc_vector(i));
770
771           pc_value_xor: XORCY
```

```
          pc_value_xor: XORCY
         port map( LI => pc_value(i),
                   CI => pc_value_carry(i-1),
                    O => inc_pc_value(i));

       end generate pc_msb_carry;

  end generate pc_loop;

  address <= pc;
--
------------------------------------------------------------------------------------
--
-- Register Bank and second operand selection.
--
-- Definition of an 8-bit dual port RAM with 16 locations
-- including write enable decode.
--
-- Outputs are assigned to PORT_ID and OUT_PORT.
--
------------------------------------------------------------------------------------
--
  -- Forming decode signal

  register_type_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"0145")
  --synthesis translate_on
  port map( I0 => active_interrupt,
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => register_type );

  register_write_flop: FD
  port map ( D => register_type,
             Q => register_write,
             C => clk);

  register_enable_lut: LUT2
  --synthesis translate_off
    generic map (INIT => X"8")
  --synthesis translate_on
  port map( I0 => t_state,
            I1 => register_write,
             O => register_enable );

  reg_loop: for i in 0 to 7 generate



  --
  begin

    register_bit: RAM16X1D
    --synthesis translate_off
    generic map(INIT => X"0000")
    --synthesis translate_on
    port map (       D => alu_result(i),
                    WE => register_enable,
                  WCLK => clk,
                    A0 => instruction(8),
                    A1 => instruction(9),
                    A2 => instruction(10),
                    A3 => instruction(11),
                 DPRA0 => instruction(4),
                 DPRA1 => instruction(5),
                 DPRA2 => instruction(6),
                 DPRA3 => instruction(7),
                   SPO => sx(i),
                   DPO => sy(i));

    operand_select_mux: LUT3
    --synthesis translate_off
      generic map (INIT => X"E4")
    --synthesis translate_on
    port map( I0 => instruction(12),
              I1 => instruction(i),
              I2 => sy(i),
               O => second_operand(i) );

  end generate reg_loop;

  out_port <= sx;
  port_id <= second_operand;
--
------------------------------------------------------------------------------------
--
-- Store Memory
--
-- Definition of an 8-bit single port RAM with 64 locations
-- including write enable decode.
--
------------------------------------------------------------------------------------
--
  -- Forming decode signal

  memory_type_lut: LUT4
  --synthesis translate_off
```

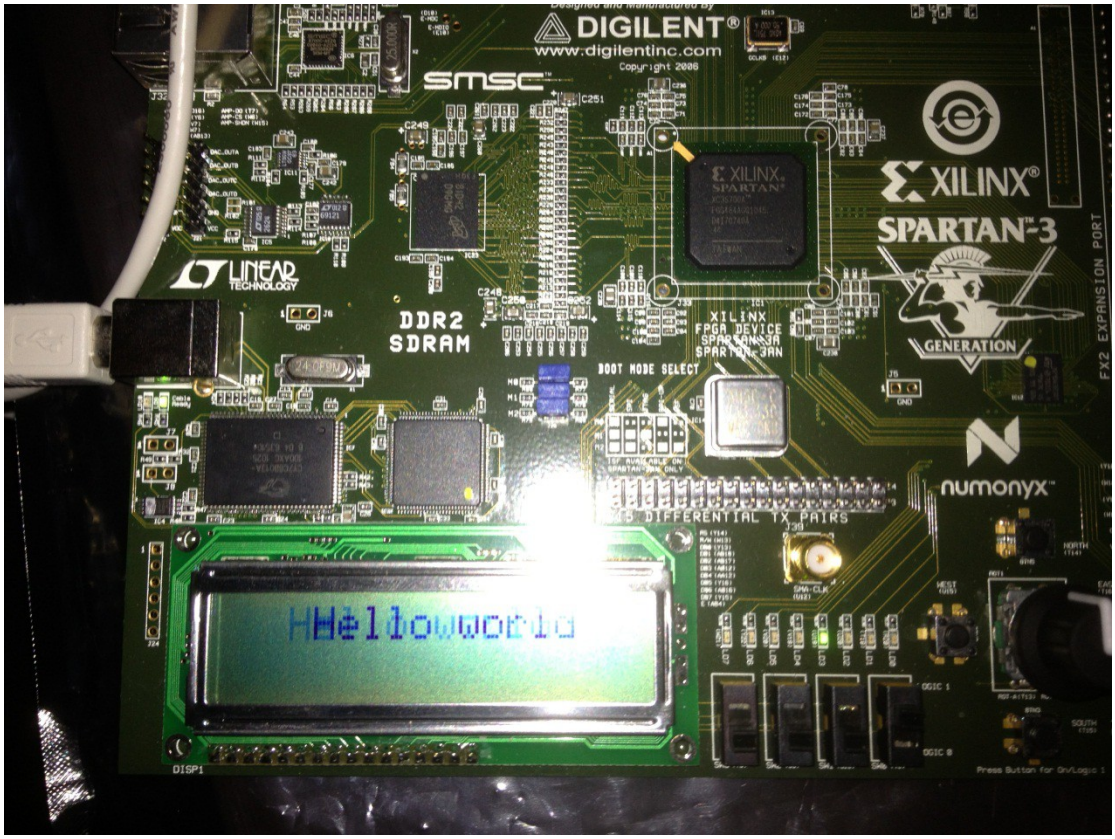
```
  -- Forming decode signal

  memory_type_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"0400")
  --synthesis translate_on
  port map( I0 => active_interrupt,
            I1 => instruction(15),
            I2 => instruction(16),
            I3 => instruction(17),
             O => memory_type );

  memory_write_flop: FD
  port map ( D => memory_type,
             Q => memory_write,
             C => clk);

  memory_enable_lut: LUT4
  --synthesis translate_off
    generic map (INIT => X"8000")
  --synthesis translate_on
  port map( I0 => t_state,
            I1 => instruction(13),
            I2 => instruction(14),
            I3 => memory_write,
             O => memory_enable );

  store_loop: for i in 0 to 7 generate
  --
  -- Attribute to define RAM contents during implementation
  -- The information is repeated in the generic map for functional simulation
  --
  attribute INIT : string;
  attribute INIT of memory_bit : label is "0000000000000000";
  --
  begin

    memory_bit: RAM64X1S
    --synthesis translate_off
    generic map(INIT => X"0000000000000000")
    --synthesis translate_on
    port map (       D => sx(i),
                    WE => memory_enable,
                  WCLK => clk,
                    A0 => second_operand(0),
                    A1 => second_operand(1),
                    A2 => second_operand(2),
                    A3 => second_operand(3)
```

Κομμάτι κώδικα από το control.vhd

```
--
-- Signals used to interface to rotary encoder
--
signal      rotary_a_in : std_logic;
signal      rotary_b_in : std_logic;
signal  rotary_press_in : std_logic;
signal        rotary_in : std_logic_vector(1 downto 0);
signal        rotary_q1 : std_logic;
signal        rotary_q2 : std_logic;
signal  delay_rotary_q1 : std_logic;
signal     rotary_event : std_logic;
signal      rotary_left : std_logic;
--
--
------------------------------------------------------------------------------------------------------------------------
--
-- Start of circuit description
--
begin
  --
  ----------------------------------------------------------------------------------------------------------------------
  -- Disable unused components
  ----------------------------------------------------------------------------------------------------------------------
  --
  --StrataFLASH must be disabled to prevent it conflicting with the LCD display
  --
  strataflash_oe <= '1';
  strataflash_ce <= '1';
  strataflash_we <= '1';
  --
  --
  ----------------------------------------------------------------------------------------------------------------------
  -- KCPSM3 and the program memory
  ----------------------------------------------------------------------------------------------------------------------
  --

  processor: kcpsm3
    port map(      address => address,
               instruction => instruction,
                   port_id => port_id,
              write_strobe => write_strobe,
                  out_port => out_port,
               read_strobe => read_strobe,
                   in_port => in_port,
                 interrupt => interrupt,
             interrupt_ack => interrupt_ack,
                     reset => kcpsm3_reset,
```

```
--
begin
--
  --Instantiate the Xilinx primitive for a block RAM
  ram_1024_x_18: RAMB16_S18
  --synthesis translate_off
  --INIT values repeated to define contents for functional simulation
  generic map ( INIT_00 => X"4D006D035419200240010EF40F01ED030DFFE00200080037 00DD0510C00100C7",
                INIT_01 => X"00E75C22EF00CE010075ED03EDFF400900E75C0BEF00CE010075102 94DFF1026",
                INIT_02 => X"2001E000A07F5034208060006A02A000C0804000400B541B200240010EF40F01",
                INIT_03 => X"056F009D056C009D056C009D0565009D0548A000CA80EA020A0C40340A025033",
                INIT_04 => X"009D0573009D056BA000006905640 09D056C009D0572009D056F009D05770069",
                INIT_05 => X"009D056D009D0561009D0569009D0564009D052D009D0570009D056F009D0569",
                INIT_06 => X"A000546DC001000BA000009D0520A000009D0564009D0574009D056E009D0561",
                INIT_07 => X"0432A000547BC30100750314A0005476C201007002 19A0005471C101006C0128",
                INIT_08 => X"A4F01450A0000084C440A4F8A000C440E401006CC440E401A0005480C401007A",
                INIT_09 => X"C40CA4F01450A000C44004F00070008A0406040604060407 1450006C008AC408",
                INIT_0A => X"C440040EA000C44004F000700084C4400406040604070407 1450006C0084C440",
                INIT_0B => X"000E000EA5F0C440E4014002006CC440E401006CC440E4014502006CC440E401",
                INIT_0C => X"04200070008A0075008A007A008A0430007AA0000070C4400404D500000E000E",
                INIT_0D => X"A50F50E32510A0000075007 5008E0501008E050C008E0506008E0528007 0008A",
                INIT_0E => X"C0804001 50F140006003E001A000008E0518A000008EC5C0A50FA000008EC580",
                INIT_0F => X"0000000000000000000000000000000000000000000000000000000000080016001E000",
                INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000000000000",
                INIT_3F => X"40EA0000000000000000000000000000000000000000000000000000000000000000000000",
               INITP_00 => X"2DCB72DCB4B2CCCCCCCCCCCCCCB3333333332ABB63428F40F5E3F5F74D0223CF",
               INITP_01 => X"000000320D2B2C2C36FCCCCF9FFCEE0AA20E38388A3EAA3E028FAA9C0B8A38B7",
               INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
               INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
               INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
               INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
               INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
               INITP_07 => X"C000000000000000000000000000000000000000000000000000000000000000")
    --synthesis translate_on
    port map(    DI => "0000000000000000",
                DIP => "00",
                 EN => '1',
                 WE => '0',
                SSR => '0',
                CLK => clk,
               ADDR => address,
                 DO => instruction(15 downto 0),
                DOP => instruction(17 downto 16));
  --
  end low_level_definition;
  --
  ------------------------------------------------------------------------------
  --
  -- END OF FILE control.vhd
  --
  ------------------------------------------------------------------------------
```

Η εμφάνιση του Hello world στην οθόνη


DIGILENT®
www.digilentinc.com
Copyright 2006
XILINX®
SPARTAN™-3
GENERATION
LINEAR TECHNOLOGY
DDR2 SDRAM
BOOT MODE SELECT
numonyx™
FX2 EXPANSION PORT
DISP1
HHello world