

Σαλής Χρήστος

Βοηθητική Εργασία σε vhdl για την Αρχιτεκτονική Υπολογιστών

<http://arch.ict.e.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

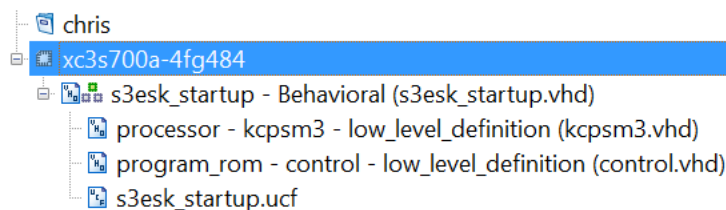
PASS 9 - Writing Verilog memory definition file
         control.v

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.

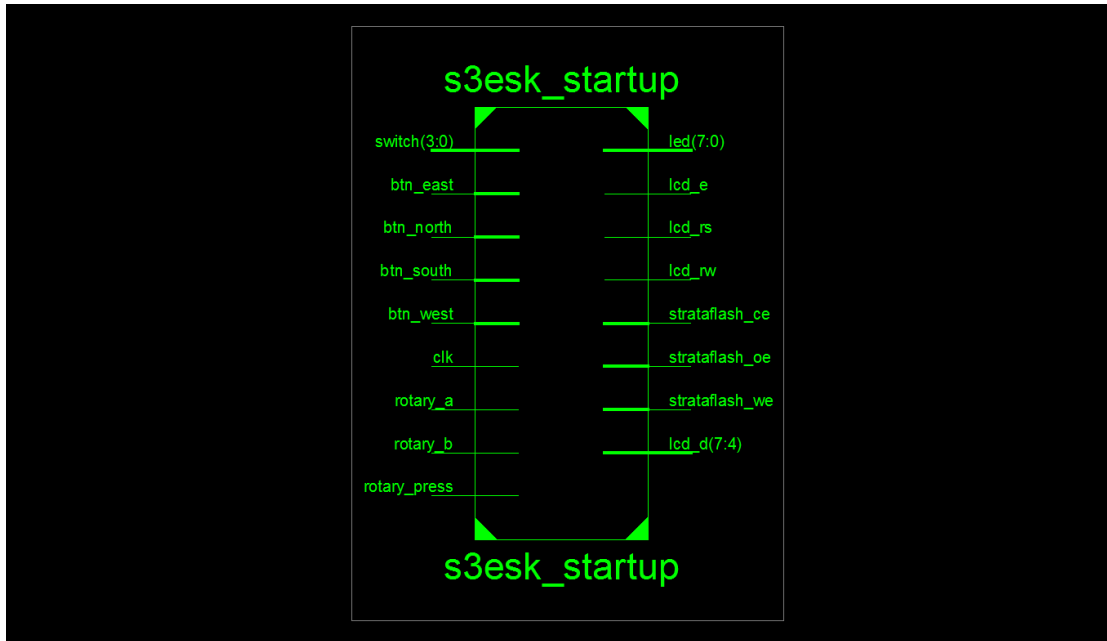
```

NET "clk" PERIOD = 20.0ns HIGH 50%;
#
# soldered 50MHz Clock.
#
NET "clk" LOC = "E12" | IOSTANDARD = LVTTTL;
#
#
# Simple LEDs
# Require only 3.5mA.
#
NET "led<0>" LOC = "R20" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<1>" LOC = "T19" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<2>" LOC = "U20" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<3>" LOC = "U19" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<4>" LOC = "V19" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<5>" LOC = "V20" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<6>" LOC = "Y22" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<7>" LOC = "W21" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
#
#
# LCD display
# Very slow so can use lowest drive strength.
#
NET "lcd_rs" LOC = "Y14" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_rw" LOC = "W13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
#
#
NET "lcd_rw" LOC = "W13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_e" LOC = "AB4" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<4>" LOC = "AA12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<5>" LOC = "Y16" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<6>" LOC = "AB16" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "lcd_d<7>" LOC = "Y15" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
#
# Strata Flash (need to disable to use LCD display)
#
NET "strataflash_oe" LOC = "C18" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "strataflash_ce" LOC = "D16" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
NET "strataflash_we" LOC = "D17" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 2;
#
#
# Simple switches
# Pull UP resistors used to stop floating condition during switching.
#
NET "switch<0>" LOC = "V8" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<1>" LOC = "U10" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<2>" LOC = "U8" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<3>" LOC = "T9" | IOSTANDARD = LVTTTL | PULLUP;
#
#
# Press buttons
# Must have pull DOWN resistors to provide Low when not pressed.
#
NET "btn_north" LOC = "T14" | IOSTANDARD = LVTTTL | PULLDOWN;
NET "btn_east" LOC = "T16" | IOSTANDARD = LVTTTL | PULLDOWN;
NET "btn_south" LOC = "T15" | IOSTANDARD = LVTTTL | PULLDOWN;
NET "btn_west" LOC = "U15" | IOSTANDARD = LVTTTL | 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 = LVTTTL | PULLUP;
NET "rotary_b" LOC = "R14" | IOSTANDARD = LVTTTL | PULLUP;
NET "rotary_press" LOC = "R13" | IOSTANDARD = LVTTTL | PULLDOWN;

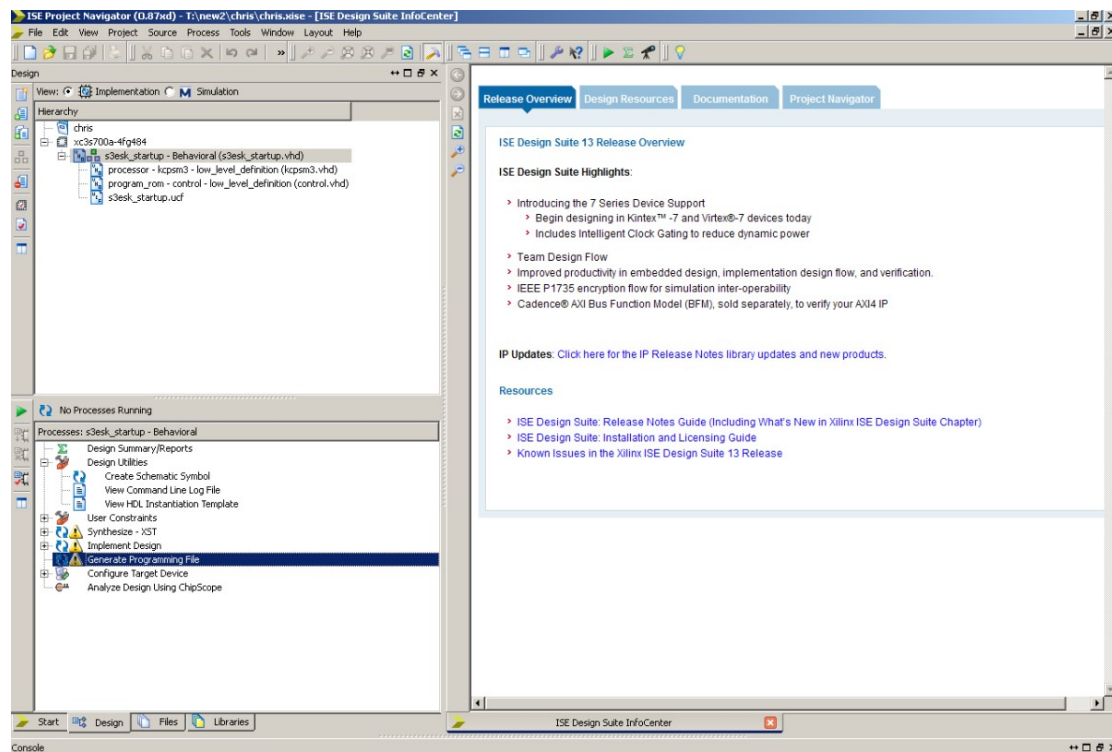
```

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 kcp3m3 is
78     Port (
79         address : out std_logic_vector(9 downto 0);
80         instruction : in std_logic_vector(17 downto 0);
81         port_id : out std_logic_vector(7 downto 0);
82         write_strobe : out std_logic;
83         out_port : out std_logic_vector(7 downto 0);
84         read_strobe : out std_logic;
85         in_port : in std_logic_vector(7 downto 0);
86         interrupt : in std_logic;
87         interrupt_ack : out std_logic;
88         reset : in std_logic;
89         clk : in std_logic);
90 end kcp3m3;
91 -----
92 --
93 -- Start of Main Architecture for KCPSM3
94 --
95 architecture low_level_definition of kcp3m3 is
96 --
97 -----
98 --
99 -- Signals used in KCPSM3
100 --
101 -----
102 --
103 -- Fundamental control and decode signals
104 --
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;
111 -----
112 -----
113 -----
114 -----
115 -----
116 -----
117 -----
118 -----
119 -----
120 -----
121 -----
122 -----
123 -----
124 -----
125 -----
126 -----
127 -----
128 -----
129 -----
130 -----
131 -----
132 -----
133 -----
134 -----
135 -----
136 -----
137 -----
138 -----
139 -----
140 -----
141 -----
142 -----
143 -----
144 -----
145 -----
146 -----
147 -----
148 -----
149 -----
150 -----
151 -----
152 -----
153 -----
154 -----
155 -----
156 -----
157 -----
158 -----
159 -----
160 -----
161 -----
162 -----
163 -----
164 -----
165 -----
166 -----
167 -----
168 -----
169 -----
170 -----
171 -----
172 -----
173 -----
174 -----
175 -----
176 -----
177 -----
178 -----
179 -----
180 -----
181 -----
182 -----
183 -----
184 -----
185 -----
186 -----
187 -----
188 -----
189 -----
190 -----
191 -----
192 -----
193 -----
194 -----
195 -----
196 -----
197 -----
198 -----
199 -----
200 -----
201 -----
202 -----
203 -----
204 -----
205 -----
206 -----
207 -----
208 -----
209 -----
210 -----
211 -----
212 -----
213 -----
214 -----
215 -----
216 -----
217 -----
218 -----
219 -----
220 -----
221 -----
222 -----
223 -----
224 -----
225 -----
226 -----
227 -----
228 -----
229 -----
230 -----
231 -----
232 -----
233 -----
234 -----
235 -----
236 -----
237 -----
238 -----
239 -----
240 -----
241 -----
242 -----
243 -----
244 -----
245 -----
246 -----
247 -----
248 -----
249 -----
250 -----
251 -----
252 -----
253 -----
254 -----
255 -----
256 -----
257 -----
258 -----
259 -----
260 -----
261 -----
262 -----
263 -----
264 -----
265 -----
266 -----
267 -----
268 -----
269 -----
270 -----
271 -----
272 -----
273 -----
274 -----
275 -----
276 -----
277 -----
278 -----
279 -----
280 -----
281 -----
282 -----
283 -----
284 -----
285 -----
286 -----
287 -----
288 -----
289 -----
290 -----
291 -----
292 -----
293 -----
294 -----
295 -----
296 -----
297 -----
298 -----
299 -----
300 -----
301 -----
302 -----
303 -----
304 -----
305 -----
306 -----
307 -----
308 -----
309 -----
310 -----
311 -----
312 -----
313 -----
314 -----
315 -----
316 -----
317 -----
318 -----
319 -----
320 -----
321 -----
322 -----
323 -----
324 -----
325 -----
326 -----
327 -----
328 -----
329 -----
330 -----
331 -----
332 -----
333 -----
334 -----
335 -----
336 -----
337 -----
338 -----
339 -----
340 -----
341 -----
342 -----
343 -----
344 -----
345 -----
346 -----
347 -----
348 -----
349 -----
350 -----
351 -----
352 -----
353 -----
354 -----
355 -----
356 -----
357 -----
358 -----
359 -----
360 -----
361 -----
362 -----
363 -----
364 -----
365 -----
366 -----
367 -----
368 -----
369 -----
370 -----
371 -----
372 -----
373 -----
374 -----
375 -----
376 -----
377 -----
378 -----
379 -----
380 -----
381 -----
382 -----
383 -----
384 -----
385 -----
386 -----
387 -----
388 -----
389 -----
390 -----
391 -----
392 -----
393 -----
394 -----
395 -----
396 -----
397 -----
398 -----
399 -----
400 -----
401 -----
402 -----
403 -----
404 -----
405 -----
406 -----
407 -----
408 -----
409 -----
410 -----
411 -----
412 -----
413 -----
414 -----
415 -----
416 -----
417 -----
418 -----
419 -----
420 -----
421 -----
422 -----
423 -----
424 -----
425 -----
426 -----
427 -----
428 -----
429 -----
430 -----
431 -----
432 -----
433 -----
434 -----
435 -----
436 -----
437 -----
438 -----
439 -----
440 -----
441 -----
442 -----
443 -----
444 -----
445 -----
446 -----
447 -----
448 -----
449 -----
450 -----
451 -----
452 -----
453 -----
454 -----
455 -----
456 -----
457 -----
458 -----
459 -----
460 -----
461 -----
462 -----
463 -----
464 -----
465 -----
466 -----
467 -----
468 -----
469 -----
470 -----
471 -----
472 -----
473 -----
474 -----
475 -----
476 -----
477 -----
478 -----
479 -----
480 -----
481 -----
482 -----
483 -----
484 -----
485 -----
486 -----
487 -----
488 -----
489 -----
490 -----
491 -----
492 -----
493 -----
494 -----
495 -----
496 -----
497 -----
498 -----
499 -----
500 -----
501 -----
502 -----
503 -----
504 -----
505 -----
506 -----
507 -----
508 -----
509 -----
510 -----
511 -----
512 -----
513 -----
514 -----
515 -----
516 -----
517 -----
518 -----
519 -----
520 -----
521 -----
522 -----
523 -----
524 -----
525 -----
526 -----
527 -----
528 -----
529 -----
530 -----
531 -----
532 -----
533 -----
534 -----
535 -----
536 -----
537 -----
538 -----
539 -----
540 -----
541 -----
542 -----
543 -----
544 -----
545 -----
546 -----
547 -----
548 -----
549 -----
550 -----
551 -----
552 -----
553 -----
554 -----
555 -----
556 -----
557 -----
558 -----
559 -----
560 -----
561 -----
562 -----
563 -----
564 -----
565 -----
566 -----
567 -----
568 -----
569 -----
570 -----
571 -----
572 -----
573 -----
574 -----
575 -----
576 -----
577 -----
578 -----
579 -----
580 -----
581 -----
582 -----
583 -----
584 -----
585 -----
586 -----
587 -----
588 -----
589 -----
590 -----
591 -----
592 -----
593 -----
594 -----
595 -----
596 -----
597 -----
598 -----
599 -----
600 -----
601 -----
602 -----
603 -----
604 -----
605 -----
606 -----
607 -----
608 -----
609 -----
610 -----
611 -----
612 -----
613 -----
614 -----
615 -----
616 -----
617 -----
618 -----
619 -----
620 -----
621 -----
622 -----
623 -----
624 -----
625 -----
626 -----
627 -----
628 -----
629 -----
630 -----
631 -----
632 -----
633 -----
634 -----
635 -----
636 -----
637 -----
638 -----
639 -----
640 -----
641 -----
642 -----
643 -----
644 -----
645 -----
646 -----
647 -----
648 -----
649 -----
650 -----
651 -----
652 -----
653 -----
654 -----
655 -----
656 -----
657 -----
658 -----
659 -----
660 -----
661 -----
662 -----
663 -----
664 -----
665 -----
666 -----
667 -----
668 -----
669 -----
670 -----
671 -----
672 -----
673 -----
674 -----
675 -----
676 -----
677 -----
678 -----
679 -----
680 -----
681 -----
682 -----
683 -----
684 -----
685 -----
686 -----
687 -----
688 -----
689 -----
690 -----
691 -----
692 -----
693 -----
694 -----
695 -----
696 -----
697 -----
698 -----
699 -----
700 -----
701 -----
702 -----
703 -----
704 -----
705 -----
706 -----
707 -----
708 -----
709 -----
710 -----
711 -----
712 -----
713 -----
714 -----
715 -----
716 -----
717 -----
718 -----
719 -----
720 -----
721 -----
722 -----
723 -----
724 -----
725 -----
726 -----
727 -----
728 -----
729 -----
730 -----
731 -----
732 -----
733 -----
734 -----
735 -----
736 -----
737 -----
738 -----
739 -----
740 -----
741 -----
742 -----
743 -----
744 -----
745 -----
746 -----
747 -----
748 -----
749 -----
750 -----
751 -----
752 -----
753 -----
754 -----
755 -----
756 -----
757 -----
758 -----
759 -----
760 -----
761 -----
762 -----
763 -----
764 -----
765 -----
766 -----
767 -----
768 -----
769 -----
770 -----
771 -----
772 -----
773 -----
774 -----
775 -----
776 -----
777 -----
778 -----
779 -----
780 -----
781 -----
782 -----
783 -----
784 -----
785 -----
786 -----
787 -----
788 -----
789 -----
790 -----
791 -----
792 -----
793 -----
794 -----
795 -----
796 -----
797 -----
798 -----
799 -----
800 -----
801 -----
802 -----
803 -----
804 -----
805 -----
806 -----
807 -----
808 -----
809 -----
810 -----
811 -----
812 -----
813 -----
814 -----
815 -----
816 -----
817 -----
818 -----
819 -----
820 -----
821 -----
822 -----
823 -----
824 -----
825 -----
826 -----
827 -----
828 -----
829 -----
830 -----
831 -----
832 -----
833 -----
834 -----
835 -----
836 -----
837 -----
838 -----
839 -----
840 -----
841 -----
842 -----
843 -----
844 -----
845 -----
846 -----
847 -----
848 -----
849 -----
850 -----
851 -----
852 -----
853 -----
854 -----
855 -----
856 -----
857 -----
858 -----
859 -----
860 -----
861 -----
862 -----
863 -----
864 -----
865 -----
866 -----
867 -----
868 -----
869 -----
870 -----
871 -----
872 -----
873 -----
874 -----
875 -----
876 -----
877 -----
878 -----
879 -----
880 -----
881 -----
882 -----
883 -----
884 -----
885 -----
886 -----
887 -----
888 -----
889 -----
890 -----
891 -----
892 -----
893 -----
894 -----
895 -----
896 -----
897 -----
898 -----
899 -----
900 -----
901 -----
902 -----
903 -----
904 -----
905 -----
906 -----
907 -----
908 -----
909 -----
910 -----
911 -----
912 -----
913 -----
914 -----
915 -----
916 -----
917 -----
918 -----
919 -----
920 -----
921 -----
922 -----
923 -----
924 -----
925 -----
926 -----
927 -----
928 -----
929 -----
930 -----
931 -----
932 -----
933 -----
934 -----
935 -----
936 -----
937 -----
938 -----
939 -----
940 -----
941 -----
942 -----
943 -----
944 -----
945 -----
946 -----
947 -----
948 -----
949 -----
950 -----
951 -----
952 -----
953 -----
954 -----
955 -----
956 -----
957 -----
958 -----
959 -----
960 -----
961 -----
962 -----
963 -----
964 -----
965 -----
966 -----
967 -----
968 -----
969 -----
970 -----
971 -----
972 -----
973 -----
974 -----
975 -----
976 -----
977 -----
978 -----
979 -----
980 -----
981 -----
982 -----
983 -----
984 -----
985 -----
986 -----
987 -----
988 -----
989 -----
990 -----
991 -----
992 -----
993 -----
994 -----
995 -----
996 -----
997 -----
998 -----
999 -----
1000 -----

```

```

126 --
127 -- Signals used to interface to rotary encoder
128 --
129 signal rotary_a_in : std_logic;
130 signal rotary_b_in : std_logic;
131 signal rotary_press_in : std_logic;
132 signal rotary_in : std_logic_vector(1 downto 0);
133 signal rotary_q1 : std_logic;
134 signal rotary_q2 : std_logic;
135 signal delay_rotary_q1 : std_logic;
136 signal rotary_event : std_logic;
137 signal rotary_left : std_logic;
138 --
139 --
140 -----
141 --
142 -- Start of circuit description
143 --
144 begin
145 --
146 -----
147 -- Disable unused components
148 -----
149 --
150 --StrataFLASH must be disabled to prevent it conflicting with the LCD display
151 --
152 strataflash_oe <= '1';
153 strataflash_ce <= '1';
154 strataflash_we <= '1';
155 --
156 --
157 -----
158 -- KCPSM3 and the program memory
159 -----
160 --
161 --
162 processor: kcpam3
163   port map(
164     address => address,
165     instruction => instruction,
166     port_id => port_id,
167     write_strobe => write_strobe,
168     out_port => out_port,
169     read_strobe => read_strobe,
170     in_port => in_port,
171     interrupt => interrupt,
172     interrupt_ack => interrupt_ack,
173     reset => kcpam3_reset,
174     clk => clk);

```

-- 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;
signal invert_arith_carry : std_logic;

```

```

283 --
284 -- Fundamental Control
285 --
286 -- Definition of T-state and internal reset
287 --
288 -----
289
290 t_state_lut: LUT1
291 --synthesis translate_off
292 generic map (INIT => X"1")
293 --synthesis translate_on
294 port map( IO => t_state,
295           O => not_t_state );
296
297 toggle_flop: FDR
298 port map ( D => not_t_state,
299           Q => t_state,
300           R => internal_reset,
301           C => clk);
302
303 reset_flop1: FDS
304 port map ( D => '0',
305           Q => reset_delay,
306           S => reset,
307           C => clk);
308
309 reset_flop2: FDS
310 port map ( D => reset_delay,
311           Q => internal_reset,
312           S => reset,
313           C => clk);
314
315 -----
316
317 -- Interrupt input logic, Interrupt enable and shadow Flags.
318 --
319 -- Captures interrupt input and enables the shadow flags.
320 -- Decodes instructions which set and reset the interrupt enable flip-flop.
321 --
322 -----
323
324 -- Interrupt capture
325 --
326
327 int_capture_flop: FDR
328 port map ( D => interrupt,
329           Q => clean_int,
330           R => internal_reset

```

```

377           I1 => instruction(15),
378           I2 => instruction(16),
379           I3 => instruction(17),
380           O => int_update_enable );
381
382 int_value_lut: LUT3
383 --synthesis translate_off
384 generic map (INIT => X"04")
385 --synthesis translate_on
386 port map( IO => active_interrupt,
387           I1 => instruction(0),
388           I2 => interrupt_ack_internal,
389           O => int_enable_value );
390
391 int_enable_flop: FDRE
392 port map ( D => int_enable_value,
393           Q => int_enable,
394           CE => int_update_enable,
395           R => internal_reset,
396           C => clk);
397
398 -----
399
400 -- Decodes for the control of the program counter and CALL/RETURN stack
401 --
402 -----
403
404 move_group_lut: LUT4
405 --synthesis translate_off
406 generic map (INIT => X"7400")
407 --synthesis translate_on
408 port map( IO => instruction(14),
409           I1 => instruction(15),
410           I2 => instruction(16),
411           I3 => instruction(17),
412           O => move_group );
413
414 condition_met_lut: LUT4
415 --synthesis translate_off
416 generic map (INIT => X"5A3C")
417 --synthesis translate_on
418 port map( IO => carry_flag,
419           I1 => zero_flag,
420           I2 => instruction(10),
421           I3 => instruction(11),
422           O => condition_met );
423
424 normal_count_lut: LUT3

```

```

432
433 call_type_lut: LUT4
434 --synthesis translate_off
435 generic map (INIT => X"1000")
436 --synthesis translate_on
437 port map( I0 => instruction(14),
438           I1 => instruction(15),
439           I2 => instruction(16),
440           I3 => instruction(17),
441           O => call_type );
442
443 push_pop_lut: LUT4
444 --synthesis translate_off
445 generic map (INIT => X"5400")
446 --synthesis translate_on
447 port map( I0 => instruction(14),
448           I1 => instruction(15),
449           I2 => instruction(16),
450           I3 => instruction(17),
451           O => push_or_pop_type );
452
453 valid_move_lut: LUT2
454 --synthesis translate_off
455 generic map (INIT => X"D")
456 --synthesis translate_on
457 port map( I0 => instruction(12),
458           I1 => condition_met,
459           O => valid_to_move );
460
461 -----
462
463 -- The ZERO and CARRY Flags
464 --
465 -----
466
467 -- Enable for flags
468
469 flag_type_lut: LUT4
470 --synthesis translate_off
471 generic map (INIT => X"41FC")
472 --synthesis translate_on
473 port map( I0 => instruction(14),
474           I1 => instruction(15),
475           I2 => instruction(16),
476           I3 => instruction(17),
477           O => flag_type );
478
479 flag_extra_flag: FN
480
481
482
483
484 Q => flag_write,
485 C => clk);
486
487 flag_enable_lut: LUT2
488 --synthesis translate_off
489 generic map (INIT => X"8")
490 --synthesis translate_on
491 port map( I0 => t_state,
492           I1 => flag_write,
493           O => flag_enable );
494
495 -- Zero Flag
496
497 low_zero_lut: LUT4
498 --synthesis translate_off
499 generic map (INIT => X"0001")
500 --synthesis translate_on
501 port map( I0 => alu_result(0),
502           I1 => alu_result(1),
503           I2 => alu_result(2),
504           I3 => alu_result(3),
505           O => low_zero );
506
507 high_zero_lut: LUT4
508 --synthesis translate_off
509 generic map (INIT => X"0001")
510 --synthesis translate_on
511 port map( I0 => alu_result(4),
512           I1 => alu_result(5),
513           I2 => alu_result(6),
514           I3 => alu_result(7),
515           O => high_zero );
516
517 low_zero_muxcy: MUXCY
518 port map( DI => '0',
519           CI => '1',
520           S => low_zero,
521           O => low_zero_carry );
522
523 high_zero_cymux: MUXCY
524 port map( DI => '0',
525           CI => low_zero_carry,
526           S => high_zero,
527           O => high_zero_carry );
528
529 sel_shadow_zero_lut: LUT3
530 --synthesis translate_off
531 generic map (INIT => X"V000")

```



```

528     generic map (INIT => X"3F")
529     --synthesis translate_off
530     port map( I0 => shadow_zero,
531              I1 => instruction(16),
532              I2 => instruction(17),
533              O => sel_shadow_zero );
534
535     zero_cjmux: MUXCY
536     port map( DI => shadow_zero,
537              CI => high_zero_carry,
538              S => sel_shadow_zero,
539              O => zero_carry );
540
541     zero_xor: XORCY
542     port map( LI => '0',
543              CI => zero_carry,
544              O => zero_fast_route);
545
546     zero_flag_flop: FDRE
547     port map ( D => zero_fast_route,
548               Q => zero_flag,
549               CE => flag_enable,
550               R => internal_reset,
551               C => clk);
552
553     -- Parity detection
554
555     low_parity_lut: LUT4
556     --synthesis translate_off
557     generic map (INIT => X"6996")
558     --synthesis translate_on
559     port map( I0 => logical_result(0),
560              I1 => logical_result(1),
561              I2 => logical_result(2),
562              I3 => logical_result(3),
563              O => low_parity );
564
565     high_parity_lut: LUT4
566     --synthesis translate_off
567     generic map (INIT => X"6996")
568     --synthesis translate_on
569     port map( I0 => logical_result(4),
570              I1 => logical_result(5),
571              I2 => logical_result(6),
572              I3 => logical_result(7),
573              O => high_parity );
574
575     parity_muxcy: MUXCY

```

```

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"3FFF")
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"FB")
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"3F")

```

```

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"FFFF")
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"FF3")
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")
618 --synthesis translate_on
619 port map( I0 => sel_arith_carry,
620          I1 => sel_shift_carry,
621          O => sel_carry(2) );
622
623 sel_parity_muxcy: MUXCY
624 port map( DI => parity,
625          CI => sel_carry(2),
626          S => sel_parity,
627          O => sel_carry(3) );
628
629 carry_xor: XORCY
630 port map( LI => '0',
631          CI => sel_carry(3),
632          O => carry_fast_route);
633
634 carry_flag_flop: FDRE
635 port map ( D => carry_fast_route,
636          Q => carry_flag,
637          CE => flag_enable,
638          R => internal_reset,
639          C => clk);
640
641 -----
642 --
643 -- The Program Counter
644 --
645 -- Definition of a 10-bit counter which can be loaded from two sources
646 --
647 -----
648 --
649 invert_enable: INV -- Inverter should be implemented in the CE to flip flops
650 port map( I => t_state,
651          O => pc_enable);
652
653 pc_loop: for i in 0 to 9 generate
654 --
655 -- Attribute to define LUT contents during implementation
656 -- The information is repeated in the generic map for functional simulation
657 --
658 attribute INIT : string;
659 attribute INIT of vector_select_mux : label is "E4";
660 attribute INIT of value_select_mux : label is "E4";
661 --
662 begin
663     vector_select_mux: LUT3
664     --synthesis translate_off
665     generic map (INIT => X"333")
666     --synthesis translate_on

```

```

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( IO => instruction(i5),
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( IO => 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(i3),
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(i3),
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( I1 => 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"00")
--synthesis translate_on
port map( I0 => e_state,
          I1 => register_write,
          O => register_enable );

-- 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),
        DPR0 => instruction(4),
        DPR1 => instruction(5),
        DPR2 => instruction(6),
        DPR3 => instruction(7),
        SPO => sx(i),
        DPO => sy(i));

    operand_select_mux: LUT3
    --synthesis translate_off
    generic map (INIT => X"4")
    --synthesis translate_on
    port map( I0 => instruction(12),
              I1 => instruction(1),
              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

```

```

870 -- Forming decode signal
871
872 memory_type_lut: LUT4
873 --synthesis translate_off
874 generic map (INIT => X"0400")
875 --synthesis translate_on
876 port map( I0 => active_interrupt,
877           I1 => instruction(15),
878           I2 => instruction(16),
879           I3 => instruction(17),
880           O => memory_type );
881
882 memory_write_flops: FD
883 port map ( D => memory_type,
884           Q => memory_write,
885           C => clk);
886
887 memory_enable_lut: LUT4
888 --synthesis translate_off
889 generic map (INIT => X"8000")
890 --synthesis translate_on
891 port map( I0 => t_state,
892           I1 => instruction(13),
893           I2 => instruction(14),
894           I3 => memory_write,
895           O => memory_enable );
896
897 store_loop: for i in 0 to 7 generate
898 --
899 -- Attribute to define RAM contents during implementation
900 -- The information is repeated in the generic map for functional simulation
901 --
902 attribute INIT : string;
903 attribute INIT of memory_bit : label is "0000000000000000";
904 --
905 begin
906
907 memory_bit: RAM64X1S
908 --synthesis translate_off
909 generic map (INIT => X"0000000000000000")
910 --synthesis translate_on
911 port map ( D => sx(i),
912           WE => memory_enable,
913           WCLK => clk,
914           A0 => second_operand(0),
915           A1 => second_operand(1),
916           A2 => second_operand(2),
917           A3 => second_operand(3)

```

Κομμάτι κώδικα από το control.vhd

```

126 --
127 -- Signals used to interface to rotary encoder
128 --
129 signal rotary_a_in : std_logic;
130 signal rotary_b_in : std_logic;
131 signal rotary_press_in : std_logic;
132 signal rotary_in : std_logic_vector(1 downto 0);
133 signal rotary_q1 : std_logic;
134 signal rotary_q2 : std_logic;
135 signal delay_rotary_q1 : std_logic;
136 signal rotary_event : std_logic;
137 signal rotary_left : std_logic;
138 --
139 --
140 -----
141 --
142 -- Start of circuit description
143 --
144 begin
145 --
146 -----
147 -- Disable unused components
148 -----
149 --
150 --StrataFLASH must be disabled to prevent it conflicting with the LCD display
151 --
152 strataflash_oe <= '1';
153 strataflash_ce <= '1';
154 strataflash_we <= '1';
155 --
156 -----
157 --
158 -- KCPSM3 and the program memory
159 -----
160 --
161
162 processor: kcp3m3
163 port map(
164     address => address,
165     instruction => instruction,
166     port_id => port_id,
167     write_strobe => write_strobe,
168     out_port => out_port,
169     read_strobe => read_strobe,
170     in_port => in_port,
171     interrupt => interrupt,
172     interrupt_ack => interrupt_ack,
173     reset => kcp3m3_reset,
174     clk => clk

```

```

--
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"4D006D035419200240010EF40F01ED030DFE0020008003700DD0510C00100C7",
INIT_01 => X"00E75C22F000E010075ED03EDF40900E75C0BF00CE01007510294DF1026",
INIT_02 => X"2001E000A07F503420806006A02A00CC280400400B541B200240010EF40F01",
INIT_03 => X"056F009D056C009D056C009D0565009D0548A00CA80EA020A0C40340A025033",
INIT_04 => X"009D0573009D056BA00000690564009D056C009D0572009D056F009D05770069",
INIT_05 => X"009D056D009D0561009D0569009D0564009D052D009D052D009D056F009D0569",
INIT_06 => X"A000546DC001000EA000009D0520A000009D0564009D0574009D056E009D0561",
INIT_07 => X"0432A000547BC30100750314A0005476C20100700219A0005471C101006C012B",
INIT_08 => X"A4F01430A0000084C40404F8A00C440E011006CC440E011A0005480C401007A",
INIT_09 => X"C400C4F01450A000C44004F00070008A04060406040604071450006C0084C408",
INIT_0A => X"C440040EA000C44004F000700084C4400406040604071450006C0084C440",
INIT_0B => X"000E000EA5F0C440E401400206CC440E4011006CC440E401450206CC440E401",
INIT_0C => X"04200070008A0075008A007A008A0430007A0000070C4400404D500000E000E",
INIT_0D => X"A60F05E32510A0000075007500E0501008E050C008E0506008E05280070008A",
INIT_0E => X"C080400150F14000003E001A000008E0518A00008E5C0A50FA00008E5C580",
INIT_0F => X"000000000000000000000000000000000000000000000000000000000000",
INIT_10 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_11 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_12 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_13 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_14 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_15 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_16 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_17 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_18 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_19 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1A => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1B => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1C => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1D => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1E => X"000000000000000000000000000000000000000000000000000000000000",
INIT_1F => X"000000000000000000000000000000000000000000000000000000000000",
INIT_20 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_21 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_22 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_23 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_24 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_25 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_26 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_27 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_28 => X"000000000000000000000000000000000000000000000000000000000000",

INIT_2C => X"000000000000000000000000000000000000000000000000000000000000",
INIT_2D => X"000000000000000000000000000000000000000000000000000000000000",
INIT_2E => X"000000000000000000000000000000000000000000000000000000000000",
INIT_2F => X"000000000000000000000000000000000000000000000000000000000000",
INIT_30 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_31 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_32 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_33 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_34 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_35 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_36 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_37 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_38 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_39 => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3A => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3B => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3C => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3D => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3E => X"000000000000000000000000000000000000000000000000000000000000",
INIT_3F => X"40EA00000000000000000000000000000000000000000000000000000000",
INITE_00 => X"2DCB72DCB4B2CCCCCCCCCCCC33333332ABB63428F40F5E3F5F74D0223CF",
INITE_01 => X"000000320D2B2C2C36FC0CC3FCE0A20E38388A3EA3E028FA3C0B8A38B7",
INITE_02 => X"000000000000000000000000000000000000000000000000000000000000",
INITE_03 => X"000000000000000000000000000000000000000000000000000000000000",
INITE_04 => X"000000000000000000000000000000000000000000000000000000000000",
INITE_05 => X"000000000000000000000000000000000000000000000000000000000000",
INITE_06 => X"000000000000000000000000000000000000000000000000000000000000",
INITE_07 => X"C00000000000000000000000000000000000000000000000000000000000",
--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;
--
-----
270
271
272
273
274
275

```

Η εμφάνιση του Hello world στην οθόνη

