Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2018-2019

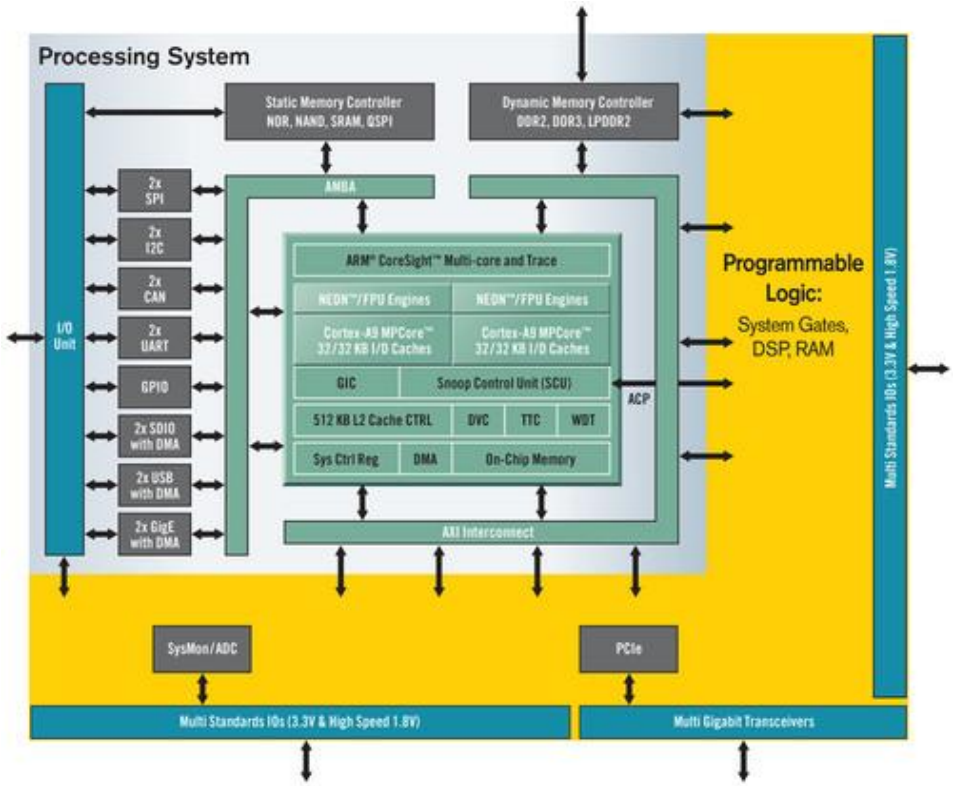
ΑΘΗΝΑ 19 Δεκεμβρίου 2018

8η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Εργ. Άσκ. στην εκπαιδευτική πλακέτα ZYBO
Εξέταση – Επίδειξη: Τετάρτη 9/1/2019

Έκθεση: Κυριακή 13/1/2019

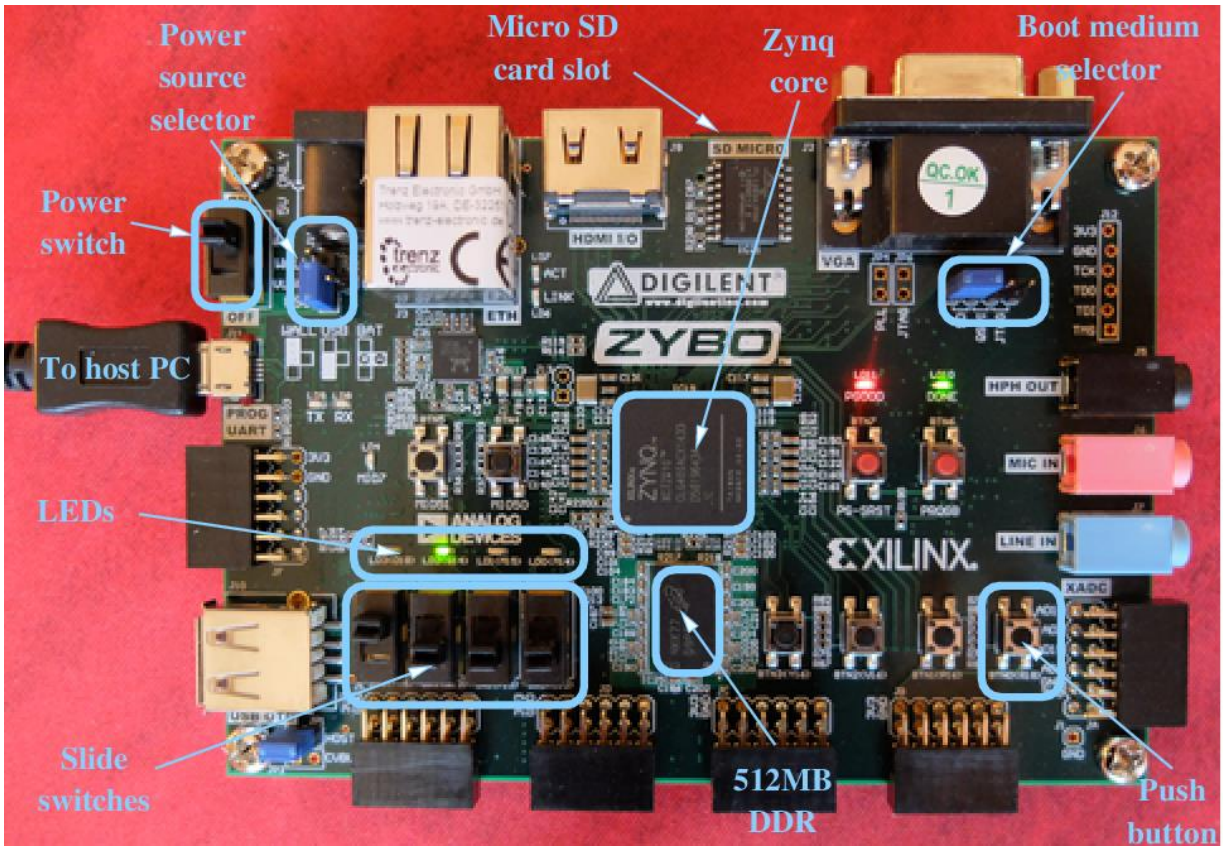
Τι είναι FPGA / Η εκπαιδευτική πλακέτα ZYBO

Τα FPGAs είναι συσκευές ημιαγωγών που αποτελούνται από επαναπρογραμματιζόμενο υλικό. Βασίζονται γύρω από μια μήτρα διαμορφώσιμων μπλοκ λογικής (CLBs), τα οποία συνδέονται μέσω προγραμματιζόμενων διασυνδέσεων. Αυτές οι διασυνδέσεις προγραμματίζονται χρησιμοποιώντας μια γλώσσα περιγραφής υλικού (HDL), όπως η Verilog και η VHDL, με σκοπό την υλοποίηση μιας εφαρμογής με συγκεκριμένη επιθυμητή λειτουργικότητα. Δεδομένου ότι κάθε μπλοκ λογικής μπορεί να είναι υπολογιστικά ανεξάρτητο από τα υπόλοιπα, υψηλά επίπεδα παραλληλίας καθίστανται ικανά, με αποτέλεσμα τη σημαντική επιτάχυνση του χρόνου εκτέλεσης ενός προγράμματος. Στο παρελθόν, τα FPGAs δεν ενσωματώνονταν στο ίδιο τσιπ με τον επεξεργαστή. Ως αποτέλεσμα, η off-chip επικοινωνία μεταξύ τους ήταν δύσκολη και αναποτελεσματική. Το 2010 η εταιρεία Xilinx εισήγαγε την οικογένεια SoC συσκευών Zynq-7000-All Programmable στην αγορά, η οποία ενσωματώνει προγραμματισμό λογισμικού ενός διπύρηνου ARM Cortex-A9 με τον προγραμματισμό υλικού των 28nm Artix-7 FPGA, προσφέροντας επιτάχυνση με χρήση υλικού και ενσωμάτωση CPU, DSP, μνήμης και πολλών περιφερειακών σε μία μόνο συσκευή (Εικόνα 1). Δεδομένου ότι ο ARM επεξεργαστής είναι ικανός να υποστηρίξει πλήρη λειτουργικά συστήματα, (το πιο συχνά χρησιμοποιούμενο από τα οποία είναι το Linux, δεδομένου ότι είναι ανοιχτού κώδικα και παρέχει μια εδραιωμένη υποστηρικτική κοινότητα) ο προγραμματιστής έχει τη δυνατότητα να αναπτύσσει την εφαρμογή χρησιμοποιώντας συνεργασία υλικού/λογισμικού, η οποία είναι ιδανική για σχεδίαση σε ενσωματωμένα συστήματα.

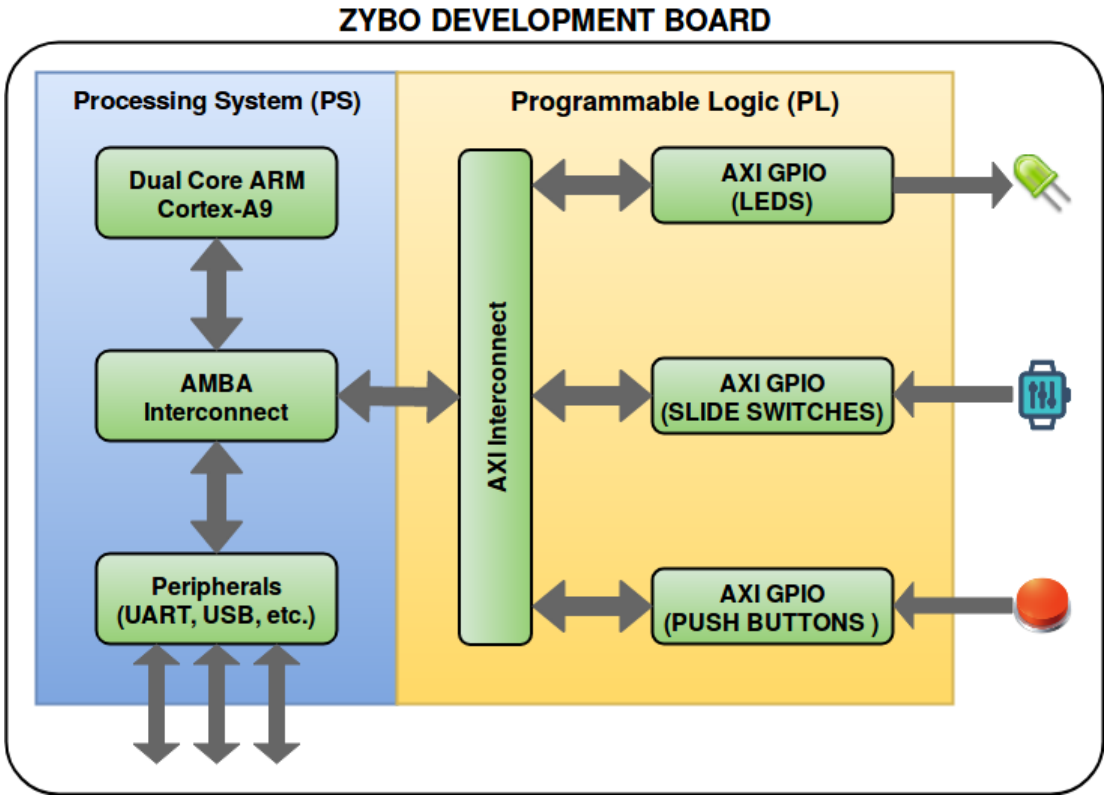


Σχήμα 8.1 Η αρχιτεκτονική των συσκευών Zynq-7000

Το SoC FPGA που θα χρησιμοποιηθεί στην παρούσα εργασία είναι το Zybo Development Board, που κατασκευάζεται από την Digilent χρησιμοποιώντας το μικρότερο μέλος της οικογένειας Xilinx Zynq-7000, το Z-7010. Το Z-7010 βασίζεται στην αρχιτεκτονική Xilinx System-on-Chip (SoC), που ενσωματώνει έναν διπύρηνο επεξεργαστή ARM Cortex-A9 με ένα Xilinx 7-series FPGA στο ίδιο ολοκληρωμένο.



Σχήμα 8.2 Η εκπαιδευτική πλακέτα ZYBO



Σχήμα 8.3 Αρχιτεκτονική συστήματος της άσκησης

Παράδειγμα:
Στο Σχήμα 8.3 δίνεται η αρχιτεκτονική του συστήματος το οποίο θα χρησιμοποιηθεί για την υλοποίηση της εργαστηριακής άσκησης, ενώ στην συνέχεια δίνεται ένα ενδεικτικό παράδειγμα προγράμματος το οποίο εμφανίζει την τιμή μιας μεταβλητής που αυξάνεται κατά ένα στα leds, ενώ παράλληλα διαβάζει ως εισόδους τις τιμές από τα push buttons και τα switches και τις εκτυπώνει.

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xgpio.h"

#define LEDS_DEV          XPAR_LEDS_DEVICE_ID
#define BUTTONS_DEV   XPAR_BUTTONS_DEVICE_ID
#define SWITCHES_DEV  XPAR_SWITCHES_DEVICE_ID

#define LED_DELAY     10000000*5

XGpio leds_inst;        // leds gpio driver instance
XGpio buttons_inst;    // buttons gpio driver instance
XGpio switches_inst;  // switches gpio driver instance

int main()
{
   int statusCodes = 0;
   uint32_t led_value = 0;
   uint32_t buttons_value = 0;
   uint32_t switches_value = 0;
   uint32_t delay = 0;

   init_platform();

   /* Initialize the GPIO driver for the leds */
   statusCodes = XGpio_Initialize(&leds_inst, LEDS_DEV);
   if (statusCodes != XST_SUCCESS) {
      xil_printf("ERROR: failed to init LEDS. Aborting\r\n");
      return XST_FAILURE;
   }

   /* Initialize the GPIO driver for the buttons */
   statusCodes = XGpio_Initialize(&buttons_inst, BUTTONS_DEV);
   if (statusCodes != XST_SUCCESS) {
      xil_printf("ERROR: failed to init BUTTONS. Aborting\r\n");
      return XST_FAILURE;
   }

   /* Initialize the GPIO driver for the switches */
   statusCodes = XGpio_Initialize(&switches_inst, SWITCHES_DEV);
   if (statusCodes != XST_SUCCESS) {
      xil_printf("ERROR: failed to init SWITCHES. Aborting\r\n");
      return XST_FAILURE;
   }

   /* Set the direction for all led signals as outputs */
   XGpio_SetDataDirection(&leds_inst, 1, 0);

   /* Set the direction for all buttons signals as inputs */
   XGpio_SetDataDirection(&buttons_inst, 1, 1);
```

```
    /* Set the direction for all switches signals as inputs */
    XGpio_SetDataDirection(&switches_inst, 1, 1);

    while(1) {
        /* Set the LED value */
        XGpio_DiscreteWrite(&leds_inst, 1, led_value = led_value);
        led_value = led_value + 1;
        /* Wait a small amount of time so the LED is visible */
        for (delay = 0; delay < LED_DELAY; delay++);

        buttons_value = XGpio_DiscreteRead(&buttons_inst, 1);
        xil_printf("buttons value: %d\r\n", buttons_value);
        for (delay = 0; delay < LED_DELAY; delay++);

        switches_value = XGpio_DiscreteRead(&switches_inst, 1);
        xil_printf("switches value: %d\r\n", switches_value);
        for (delay = 0; delay < LED_DELAY; delay++);
    }
    cleanup_platform();
    return 0;
}
```

Τα ζητούμενα της 8ης εργαστηριακής άσκησης (ZYNQ)

Ζήτημα 8.1 Να προγραμματίσετε σε C και να επιδείξετε στην εκπαιδευτική πλακέτα ZYBO πρόγραμμα που να απεικονίζει ένα αναμμένο LED. Το LED να κινείται συνεχώς ξεκινώντας από το LSB (LD0) προς τα MSB (LD3) και αντίστροφα όταν φτάνει στο άλλο άκρο. Η κίνηση του LED να ελέγχεται από το push button BTN0. Όταν αυτό είναι πατημένο η κίνηση να σταματάει, ενώ διαφορετικά να συνεχίζεται.

Ζήτημα 8.2 Να δοθεί (στην εκπαιδευτική πλακέτα ZYBO) πρόγραμμα σε C που να υλοποιεί τις λογικές συναρτήσεις:

F0 = (AB + BC + CD + DA)', F1 = ABCD + D'A', F2 = F0 + F1

όπου A = SW0, B = SW1, C = SW2, D = SW3, F0 = LD0, F1 = LD1, F2 = LD2

Ζήτημα 8.3 Να γραφεί (στην εκπαιδευτική πλακέτα ZYBO) πρόγραμμα σε C το οποίο αρχικά να ανάβει το LD0. Στη συνέχεια με το πάτημα των push buttons BTN0-BTN3 να συμβαίνουν τα εξής:

α) BTN0 ολίσθηση-περιστροφή μία θέση αριστερά (κυκλικά)

β) BTN1 ολίσθηση-περιστροφή μία θέση δεξιά (κυκλικά)

γ) BTN2 μετακίνηση του αναμμένου led στην θέση (MSB-LD3)

d) BTN3 μετακίνηση του αναμμένου led στην αρχική του θέση (LSB-LD0)

Όλες οι αλλαγές να γίνονται αφήνοντας (επανερχόμενα) τα push buttons BTNx.

ΣΗΜΕΙΩΣΗ: Για την υλοποίηση των προγραμμάτων του κάθε ζητήματος μπορείτε να χρησιμοποιήσετε τον κώδικα του παραδείγματος αλλάζοντας μόνο την λογική που υλοποιείται μέσα στην δομή while(1) {}