Dual Core Application on ZYNQ-Zybo

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ,ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΚΑΙ ΠΟΛΥΕΠΕΞΕΡΓΑΣΤΙΚΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

2019

ΣΚΟΠΟΣ ΕΡΓΑΣΤΗΡΙΟΥ

Δημιουργία εφαρμογής που κάνει χρήση δύο πυρήνων σε πλατφόρμα FPGA(ZYNQ-Zybo)

Δημιουργία Design

- Δημιουργία Project στο Vivado
  - File->Project->New
  - Στο επόμενο παράθυρο next
  - Επιλογή ονόματος project (“Dual_core_zybo”), στη συνέχεια next
  - Επιλογή RTL project και next
  - Επιλογή πλακέτας και next
  - Τέλος πατάμε finish

Block Design

- Στο flow navigator που βρίσκεται στα αριστερά του προγράμματος επιλέγουμε “Create Block Design”
- Δίνουμε όνομα “dual_core_design” και πατάμε OK
- Στο παράθυρο που δημιουργήθηκε πατάμε στο + για να προσθέσουμε IP
- Από τον κατάλογο επιλέγουμε στο ZYNQ Processing System
- Αφού δημιουργηθεί, συνδέουμε το FCLK_CLK0 με το M_AXI_GP0_ACLK
- Μετά πατάμε “Run Block Automation” και OK στο παράθυρο που θα εμφανιστεί
- Έπειτα, στο Design Sources πατάμε δεξί κλικ και επιλέγουμε “Create HDL wrapper” και OK

Export Design

- Όταν ολοκληρωθεί το παραπάνω βήμα κάνουμε synthesis, implementation και generate Bitstream
- Κλείνουμε το παράθυρο που θα εμφανιστεί μετά τη δημιουργία bitstream
- Από το μενού, επιλέγουμε File->Export->Export Hardware
- Επιλέγουμε “include Bitstream” και πατάμε OK
- Τέλος, από το μενού επιλέγουμε File->Launch SDK και OK στη συνέχεια

Δημιουργία Application στο SDK(1)

- Κατά το άνοιγμα του SDK επιλέγουμε το path του workspace, αν μας ζητηθεί
- Στο Project explorer πρέπει να εμφανίζεται το wrapper αρχείο που δημιουργήθηκε από το Vivado
- Από το μενού επιλέγουμε File->New->Application Project
- Σ αυτό το σημείο θα δημιουργήσουμε το αρχείο που θα τρέξει στον πρώτο επεξεργαστή
- Επιλέγουμε project name :“CPU0” , Processor :ps7_cortex9_0 ,Board Support Package : new με όνομα CPU0_bsp
- Στη συνέχεια πατάμε next και επιλέγουμε project Hello world
- Πατάμε Finish

Δημιουργία Application στο SDK(2)

- Ακολουθούμε την ίδια διαδικασία για το δεύτερο application που θα τρέξει στον δεύτερο επεξεργαστή.
- File->New->Application Project
- Επιλέγουμε project name :“CPU1” , Processor :ps7_cortex9_1 ,Board Support Package : new με όνομα CPU1_bsp
- Στη συνέχεια πατάμε next και επιλέγουμε project Hello world
- Πατάμε Finish

CPU0

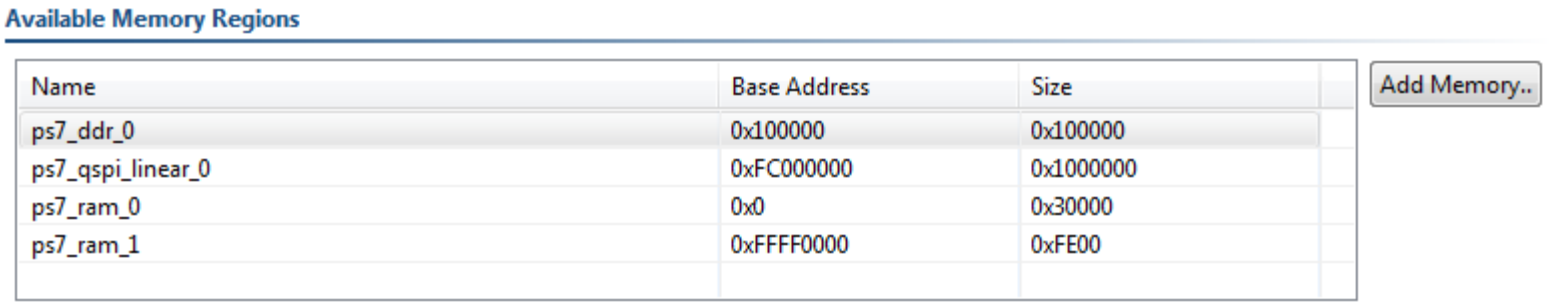
- Τροποποιούμε τον κώδικα του αρχείου helloworld στο project CPU0 σύμφωνα με τον παρακάτω κώδικα

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

#include <sleep.h>
#define COMM_VAL  (*(volatile unsigned long *)(0xFFFF0000))
int main()
{
    init_platform();
    //Disable cache on OCM
    Xil_SetTlbAttributes(0xFFFF0000,0x14de2);           // S=b1 TEX=b100 AP=b11, Domain=b1111, C=b0, B=b0
    COMM_VAL = 0;
    while(1)
    {
        print("Hello CPU0\n\r");
        sleep(1);
        COMM_VAL = 1;
        while(COMM_VAL == 1);
    }
    cleanup_platform();
    return 0;
}
```

CPU0 Linker Script

- Στο αρχείο lscript.ld δηλώνονται οι θέσεις μνημών που θα χρησιμοποιήσει η εφαρμόγή
- Αλλάζουμε το Size της ps7_ddr_0 σε 0x100000
- Αυτό το βήμα είναι υποχρεωτικό, ώστε η κάθε εφαρμογή να έχει δικές τις θέσεις μνήμης



CPU1

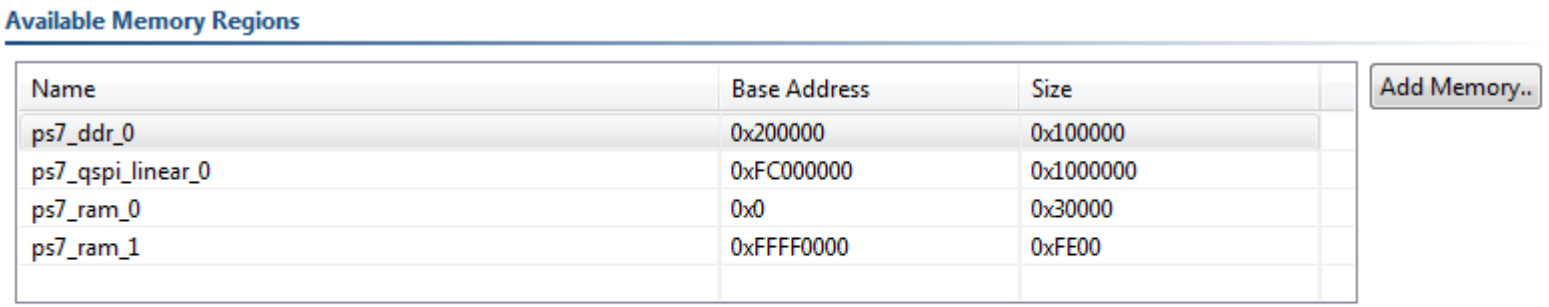
- Τροποποιούμε τον κώδικα του αρχείου helloworld στο project CPU1 σύμφωνα με τον παρακάτω κώδικα

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include <sleep.h>
#include "xil_cache.h"
#include "xil_exception.h"
#define COMM_VAL    (*(volatile unsigned long *)(0xFFFF0000))
int main()
{

    init_platform();
    //Disable cache on OCM
    Xil_SetTlbAttributes(0xFFFF0000,0x14de2);           // S=b1 TEX=b100 AP=b11, Domain=b1111, C=b0, B=b0
    print("CPU1: init_platform\n\r");
    while(1)
    {
    while(COMM_VAL == 0){};
    print("Hello CPU1\n\r");
    sleep(1);
    COMM_VAL = 0;
    }
    cleanup_platform();
    return 0;
}
```

CPU1 Linker Script

- Στο αρχείο lscript.ld του CPU1 project δηλώνουμε με αντίστοιχο τρόπο τις θέσεις μνήμης, όπως κάναμε για τη CPU0.
- Αλλάζουμε το Base address της ps7_ddr_0 σε 0x200000
- Αλλάζουμε το Size της ps7_ddr_0 σε 0x100000



Run Configurations

- Μετά τη δημιουργία των εφαρμογών από το μενού επιλέγουμε Run->Runs Configurations->Xilinx C/C++ application,System Debugger (Διπλό κλικ)
- Στο target Setup μενού στη ρύθμιση Debug Type επιλέγουμε “Standalone Application Debug”
- Στο Application μενού ακριβώς δίπλα κλικάρουμε “ps7_cortex9_0” και “ps7_cortex9_1”
- Ελέγχουμε ότι στη στήλη Project εμφανίστηκαν τα project CPU0 και CPU1 αντίστοιχα.
- Ξεκλικάρουμε την επιλογή Reset Processor
- Πατάμε run, έχοντας πρώτα συνδέσει-ενεργοποιήσει το FPGA
- Στο SDK terminal δημιουργούμε μια UART συνδέσει και παρατηρούμε το αποτέλεσμα της εφαρμογής μας

Αποτέλεσμα εφαρμογής

Connected to: Serial (COM12, 115200, 0, 8)

Hello CPU1

Hello CPU0

Hello CPU1

Hello CPU0

Hello CPU1

Hello CPU0

Hello CPU1