Πανεπιστημίο Πατρών

Μηχανικών Η/Υ και Πληροφορικής

Σχεδιασμός System-on-Chip για επεξεργασία εικόνας και υλοποίηση με FPGA.

Author: Χαράλαμπος Καλάργαρης

Supervisor: Θεμιστοκλής Χανιωτάκης

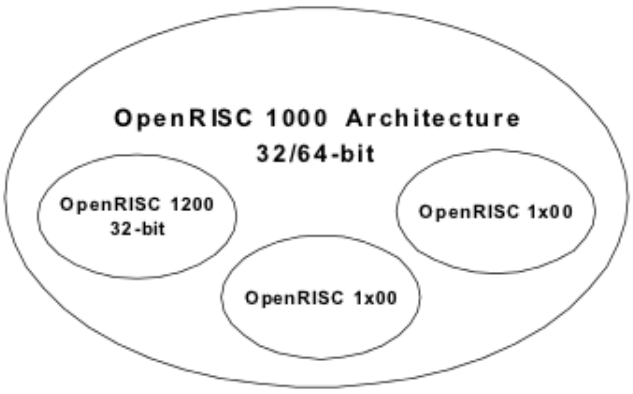
23 Μαρτίου 2012

Περιεχόμενα

1 Εισαγωγή

2 OpenRISC Family

Η OpenRISC 1000 οικογένεια επεξεργαστών αναφέρεται σε μια ελεύθερη, ανοιχτού λογισμικού RISC αρχιτεκτονική κεντρικών μονάδων επεξεργασίας.Σχετικά με την αρχιτεκτονική, η OpenRISC 1000 οικογένεια στοχεύει σε ένα φάσμα υλοποιήσεων που ποικίλουν ως προς την τιμή/αποδοση και το είδος της εφαρμογής.Είναι μια 32/64-bit φορτωσης και αποθήκευσης (load and store) RISC αρχιτεκτονική που σχεδιάστηκε με έμφαση στην απόδοση, στην απλότητα, στην χαμηλή ενεργειακή κατανάλωση, στην επεκτασιμότητα και στην ευελιξία.Η OpenRISC αρχιτεκτονίκη στοχεύει σε μεσσαία και υψηλή απόδοση δικτύωσης (networking), σε ενσωματομένα, αυτοκινητοβιομηχανικά και φορητά υπολογιστηκά περιβάλλοντα.



Σχήμα 1: Σχηματική αναπαράσταση OpenRISC αρχιτεκτονικής.

Όλες οι OpenRISC υλοποιήσεις που το πρώτο ψηφίο στον αριθμό ταυτότητας (identification number) ειναι '1' ανήκουν στην OpenRISC 1000 οικογένεια.Το δεύτερο ψηφίο ορίζει ποια χαρακτηριστικά της OpenRISC 1000 αρχιτεκτονικής είναι υλοποιημένα και με ποιο τρόπο είναι υλοποιημένα.Τα δύο τελευταία ψηφία αναφέρονται στο πως μια υλοποιήση ηταν παραμετροποιημένη πριν χρησιμοποιηθει σε πραγματική εφαρμογή.

2.1 OpenRISC 1200

Ο OR1200 είναι ενας 32-bit βαθμωτός RISC επεξεργαστής με Harvard μικροαρχιτεκτονική, 5 stage integer pipeline, υποστήριξη εικονικής μνήμης (MMU) και βασικές δυνατότητες DSP.

Οι προκαθορισμένες κρυφες μνήμες ειναι:

- 1-way direct-mapped 8KB κρυφή μνήμη δεδομένων.
- 1-way direct-mapped 8KB κρυφή μνήμη εντόλων.

- Κάθε κρυφή μνήμη έχει γραμμή μεγέθους 16-byte.
- Και οι δύο κρύφές μνήμες ειναι physically tagged(todo).

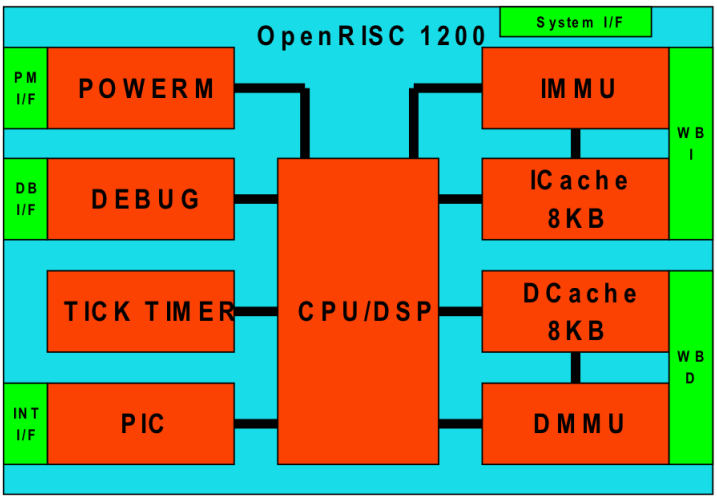
Η προκαθορισμένη MMU αποτελείται από:

- 64-entry hash based 1-way direct-mapped data TLB.
- 64-entry hash based 1-way direct-mapped instruction TLB.

Μερικές άλλες επιπρόσθετες λειτουργίες που παρέχει ο OpenRISC 1200 ειναι η μονάδα απασφαλμάτωσης πραγματικού χρόνου (real-time debug unit), υψηλής ανάλυσης χρονιστή, προγραμματιζόμενο ελεκτή διακοπών (programmable interrupt controller) και μονάδα ρύθμισης των ενεργειακών απαιτήσεων.

Ο OR1200 ουσιαστικά προορίζεται για εφαρμογες σε ενσωματομένα,φορητά και δικτύωσης συστήματα.Μπορεί να ανταγωνιστεί τους τελευταίους βαθμωτούς 32-bit RISC επεξεργαστές της κλάσης του και να υποστηρίξει αποδοτικά οποιοδήποτε μοντέρνο λειτουργικό σύστημα. Ανταγωνιστές του θεωρούνται οι ARM10, ARC και Tensilica RISC επεξεργαστές.

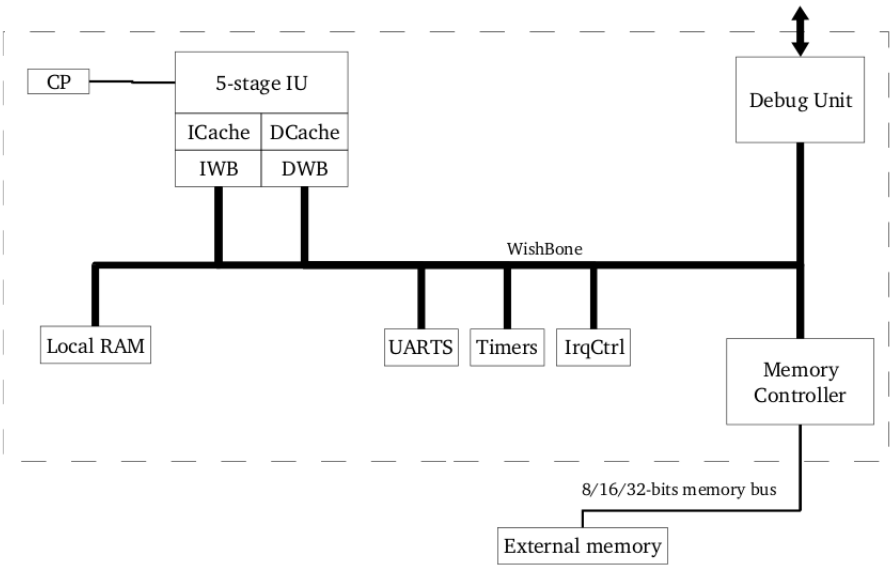
Παρακάτω βρίσκεται η σχηματική αναπαράσταση του OR1200 επεξεργαστή με τις υπομονάδες που τον αποτελούν.



Σχήμα 2: Core's Architecture

- CPU/FPU/DSP central block.
- Direct-mapped data cache.
- Direct-mapped instruction cache.
- Data MMU based on hash based DTLB.
- Instruction MMU based on hash based ITLB.
- Power management unit and power management interface.

- Tick timer.
- Debug unit and development interface.
- Interrupt controller and interrupt interface.
- Instruction and Data WISHBONE host interfaces.



Σχήμα 3: OR1200 Overview

Συνοπτικά παρουσιάζονται παρακάτω τα χαρακτηριστικά του OR1200:

	OR 1200
License	*GNU LGPL*
Platform	*FPGA, ASIC*
Distributed file format	*Verilog*
General	
Architecture	*32-bit RISC*
Byte Ordering	*Big endian*
Pipeline depth	*5*
Issue type	*Single*
Register file	
Organization	*Flat*
# of global registers	*32*
Total # of GPRs	*32*
ISA	
Type	*ORBIS32*
Addressing modes	*Immediate, displacement, pcrelative*
MAC	*32x32-bit, 48-bit Acc*
Custom instructions	*Yes*
Custom coprocessor	*Yes*
Software floating-point support	*IEEE-754 Single and double precision*
Cache	
Hierarchy	*Harvard*
Instruction cache size	*512 byte-8 Kbyte*
Data cache size	*4-8 Kbyte*
Line size	*8-16 byte*
Placement scheme	*Direct-mapped*
Valid bits	*One per cache line*
Line-locking	*Set basis*
System Interface	*Wishbone SoC rev. B32-bit*
Power Management	*Slow and idle mode, sleep mode, doze mode*
Memory	
On-chip RAM	*Configurable*
Operating system support	*Linux, uClinux, OAR RTEMS RTOS*

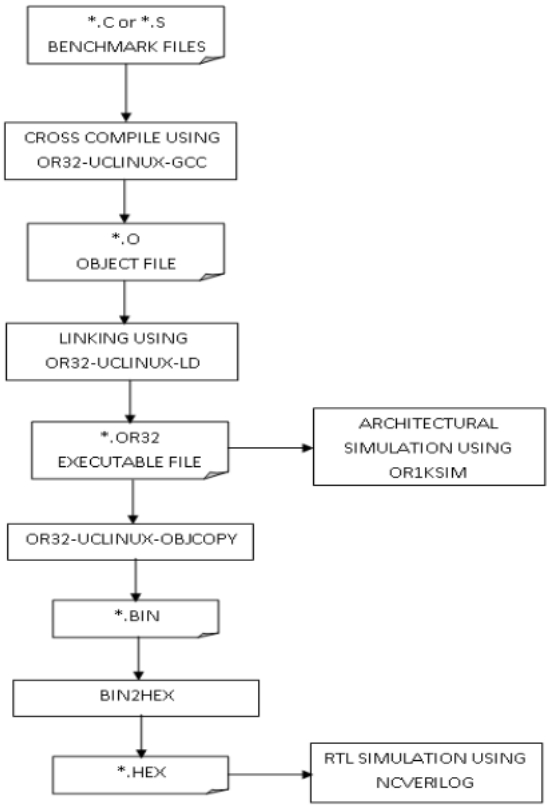
3 OR1200 Simulation

3.1 Simulation Enviroment

Υπάρχουν δύο περιβάλλοντα με τα οποία γίνετε η εξομοίωση του OR1200 επεξεργαστή. Το πρώτο χρησιμοποιεί τον OpenRISC αρχιτεκτονικό εξομοιωτή *or1ksim* και το δεύτερο χρησιμοποιέι τον *NC-Verilog* εξομοιωτή που κάνει εξομοίωση με βάση το υλικό (hardware based simulation).Στο πρώτο περιβάλλον γίνετε η επαλήθευση της λειτουργικότητας των benchmarks και στο δεύτερο προσομοιώνεται η ουσιαστική λειτουργία του OR1200 σε επίπεδο υλικού με βάση το benchmark που εκτελέσαμε.

Η συνολική ροή της εξομοίωσης παρουσιάζεται στο Σχήμα 4.Τα benchmarks είναι είτε .C αρχέια είτε .S αρχεία.Αύτα τα αρχεία, αρχίκα γίνονται cross-compiled

χρησιμοποιώντας την εντολή or32-uclinux-gcc και παράγουν ένα .O object αρχείο. Το object αρχείο μετά μετατρέπεται σε ενα .OR32 εκτελέσιμο αρχείο χρησιμοποιώντας την συνδετική εντολή or32-uclinux-ld. Αυτό το εκτελέσιμο αρχείο χρησιμοποιείται απο τον or1k αρχιτεκτονικό εξομοιωτή.Περαιτέρω το αρχείο .OR32 μετατρέπεται σε δυαδικό (binary) αρχείο χρησιμοποιώντας την εντολή or32-uclinux-objcopy.Στο τέλος δημιουργείται ενα .HEX αρχείο χρησιμοποιώντας τον binary to hex μετατροπέα bin2hex. Το παραγώμενο αρχείο .HEX φορτώνεται στην flash μνημη του RTL κώδικα του OR1200 επεξεργαστή και μετά γινετε η εξομοίωση με βάση το υλικό.



Σχήμα 4: Συνολική ροή εξομοίωσης.

3.2 Package Structure

Σε αυτό το μέρος θα παρουσιάσουμε την δομή και τα περιεχομενα του φακέλου που αποτελεί τον επεξεργαστή OR1200.

Το συγκεκριμένο πακέτο περιέχει:

- **Rtl** : Εδώ βρίσκεται ο κώδικας verilog που περιγράφει σε υλικό τον επεξεργαστή OR 1200.
- **Boards** : Περιέχει κατάλληλα αρχεία για να περάσεις τον OR1200 σε συγκεκριμένες πλακετες FPGA.

- **Sim** : Σε αυτό τον φάκελο δημιουργούνται τα αποτελέσματα της εξομοίωσης σύμφωνα με το Makefile που δημιουργήσαμε για τους σκοπούς του εργαστηρίου.

    1. *sim/bin* : Εδώ βρίσκεται το Makefile που δημιουργήσαμε για τους σκοπούς του εργαστηρίου και χειρίζεται την λειτούργια του επεξεργαστή.
    2. *sim/run* :Εδώ εκτελούνται όλες οι εντολές που σχετίζονται με την εξομοίωση του επεξεργαστη.
    3. *sim/out* :Εδώ τοποθετούνται όλα τα αρχεία που παράγονται μετά την εξομοίωση. Να σημειωθεί οτι τα waveforms τοποθετούνται στο sim/run.

- **Sw** : Εδω βρίσκονται τα βασικότερα αρχεία που είναι απαραίτητα για το cross-compilation και την σωστή λειτουργία του OpenRISC επεξεργαστή.

    1. *sw/drivers* :Εδω βρίσκονται οι drivers και τα εργαλεία για τροποποίηση του hardware.
    2. *sw/lib* :Εδω βρίσκεται μια απλή βιβλιοθήκη που σε συνδυασμο με τους drivers κατά την διάρκεια του compile δημηιουργούν την βιβλιοθήκη liborpsoc που τοποθετείται στο *sw/lib*.
    3. *sw/lib/include* :Εδώ βρίσκεται το αρχείο cpu-utils.h που περιέχει όλες τις συναρτήσεις σχετίκες με την CPU του OpenRISC.
    4. *sw/tests* :Εδώ βρίσκεται το λογισμικό που χρησιμοποιείται (.C και .S αρχέια) για να δοκιμαστεί η σωστή λειτουργία του επεξεργαστή (testing) σε υπομονάδες οπως ethmac, or1200,sdram, spi και uart.Στον φάκελο κάθε υπομονάδας (πχ sw/test/sdram) υπάρχουν δύο υποφάκελοι board και sim (πχ sw/test/sdram/board και sw/test/sdram/sim).Στον sim φάκελο υπαρχουν τα tests που εκτελούνται κατα την εξομοίωση του επεξεργαστη σε ενα PC και στον φάκελο board υπάρχουν τα tests που εκτελούνται κατά την λειτουργία του επεξεργαστή.

- **Doc** : Εδω βρισκεται documentation που χρειαζόμαστε για να καταλαβουμε την φύση των testbenches και οι οδηγιες για το simulation συμφωνα με το Makefile που δημιουργησαμε για τους σκοπους του εργαστηρίου.

3.3 Simulation commands

3.3.1 Η βασική διαδικασία

Η διαδικασία με την οποία γίνετε η εξομοίωση του OR1200 είναι η εξής:

1. Το Makefile που ελέγχει το simulation βρίσκετε στο */sim/bin/* και είναι προσπελάσιμο και απο το */sim/run/* .

2. Στον φάκελο */sim/run/* εκτελώντας την εντολή *make rtl-tests* κανει compile τον rtl (verilog) κωδικα του OR1200 και εκτελεί όλα τα testbenches (assembly) που βρίσκονται στο *sw/tests/or1200*.

3. Τα αποτελέσματα του παραπάνω βήματος τοποθετούνται στο */sim/out/*. Αυτα ειναι (στα αγγλίκα για καλύτερη κατανόηση):

    - *test-name-executed.log* : A trace of the processor after each executed instruction

- *test-name-sprs.log* : A list of processor special purpose registers (SPR) accesses is created
- *test-name-lookup.log* : A list of when each instruction was executed is generated
- *test-name-general.log* : The use of the processor's report mechanism is commonplace in the test software, as it allows for the checking of intermediate values after simulation.

3.3.2 Εκτέλεση ενος συγκεκριμένου test

Η εξομοίωση ενός συγκεκριμένου test γίνετε με την εντολή *make rtl-test TEST=test-name*[1] .Πρέπει το αρχείο *test-name.c* (ή *test-name.s*)να είναι τοποθετημένο στο *sw/tests/module /sim/* όπου module ειναι η υπομονάδα που θέλουμε να ελέξουμε με κάποιο απο τα tests που μας παρέχει (πχ sw/tests/sdram/sim/sdram-rows.c).

3.3.3 Εκτέλεση συγκεκριμένων tests μαζί

Η εξομοίωση πολλών συγκεκριμένων tests γίνετε με την εντολή *make rtl-test TEST=" test-name1 test-name2 ..."* (πχ make rtl-tests TESTS="sdram-rows uart-simple or1200-mmu or1200-fp")

3.3.4 Παρέχοντας μια προσαρμοσμένη VMEM εικόνα (image)

Είναι δυνατό να καθορίσουμε το μονοπάτι μιας ήδη υπάρχουσας VMEM εικόνας που θα την χρησιμοποιήσουμε αντί να κάνουμε πάλι απο την αρχή test το software.Χρησιμοποιώντας την μεταβλητή *USER_VMEM* μπορούμε να καθορίσουμε το μονοπάτι της VMEM εικόνας που θέλουμε να τρέξουμε. Για παράδειγμα *make rtl-test USER_VMEM=/path/to/myapp.vmem* Αυτή η εικόνα θα αντιγραφεί στο φάκελο στον οποίο εργαζόμαστε και θα μετονομαστέι σύμφωνα με το τι η μνήμη στην εξομοίωση απαιτεί.

3.3.5 Παρέχοντας ένα "precompiled" εκτελέσιμο .ELF αρχείο

Είναι δυνατό να καθορίσουμε το μονοπάτι ενός OR32 ELF εκτελέσιμου αρχείου που θα το χρησιμοποιήσουμε αντί να κάνουμε πάλι απο την αρχή test το software.Χρησιμοποιώντας την μεταβλητή *USER_ELF* μπορούμε να καθορίσουμε το μονοπάτι στο οποίο βρίσκετε αυτό το αρχείο. Για παράδειγμα *make rtl-test USER_ELF=/path/to/myapp.elf* Το ELF αρχείο θα μετατραπεί σε δυαδίκη μορφή και μετά σε VMEM και θα φορτωθεί στο μοντέλο για να εκτελεστέι.

3.3.6 Κυματομορφές

Παράλληλα με το simulation ενός ή περισσοτέρων testbench παράγονται και οι κυματομορφές των εξομοιώσεων που μας βοηθάνε στην καλύτερη κατανόηση τους.Τα αρχεία που παράγονται βρίσκονται σε φακέλους μέσα στο */sim/run/* που έχουν την ονομασία *test-name.shm* .Για να εμφανίσεις τις κυματομορφές αυτές πρέπει μέσω κονσόλας να οδηγηθείς στο /sim/run/ και μετά να εκτελέσεις την εντολή *simvision test-name.shm*.

[1] test-name: Είναι η ονομασία του test που θέλουμε να εκτελέσουμε.

3.3.7 Επιπρόσθετα επιλογές στις εντολες

Παρακάτω παρουσιάζονται μερίκες μεταβλητές που μας βοηθούν να επιλέξουμε καποιές συγκεκριμένες λειτουργίες.

- *END_TIME :*Αναγκάζει την εξομοίωση να τερματίσει ($finish).Πχ *make rtl-test TEST="or1200-mul" END_TIME=100* όπου είναι ίδιο με το *#100 $finish* σε αρχείο verilog.
- *DISABLE_PROCESSOR_LOGS :* Απενεργοποιεί την οθόνη παρακολούθησης του επεξεργαστή που συλλέγει πληροφορίες κατα την εκτέλεση μιας εξομοίωσης .Αυτό βοηθάει στην επιτάχυνση της εξομοίωσης αφού απαιτείται λιγότερος χρόνος στην εγραφή αρχείων και αποτρέπει την δημιουργία πολύ μεγάλων αρχείων σε χρονοβόρες εξομοιώσεις.
- *SIMULATOR :*Επιλέγουμε τον εξομοιωτή υλικου που θέλουμε να χρησιμοποιήσουμε. Προκαθορισμένος εξομοιωτής ειναι ο NC-Verilog.Άλλη επιλόγη ειναι το ICARUS.