



Πανεπιστημίο Πατρών

Μηχανικών Η/Υ και Πληροφορικής

---

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

---

*Author:*

Χαράλαμπος Καλάργαρης

*Supervisor:*

Θεμιστοκλής Χανιωτάκης

31 Μαρτίου 2012

## Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>2</b>
<b>2</b>	<b>OpenRISC Family</b>	<b>2</b>
2.1	OpenRISC 1200 . . . . .	2
<b>3</b>	<b>OR1200 Simulation</b>	<b>5</b>
3.1	Simulation Enviroment . . . . .	5
3.2	Package Structure . . . . .	6
3.3	Simulation commands . . . . .	7
3.3.1	Η βασική διαδικασία . . . . .	7
3.3.2	Εκτέλεση ενός συγκεκριμένου test . . . . .	8
3.3.3	Εκτέλεση συγκεκριμένων tests μαζί . . . . .	8
3.3.4	Παρέχοντας μια προσαρμοσμένη VMEM εικόνα (image) . . . . .	8
3.3.5	Παρέχοντας ένα "precompiled" εκτελέσιμο .ELF αρχείο . . . . .	8
3.3.6	Κυματομορφές . . . . .	8
3.3.7	Επιπρόσθετα επιλογές στις εντολες . . . . .	9

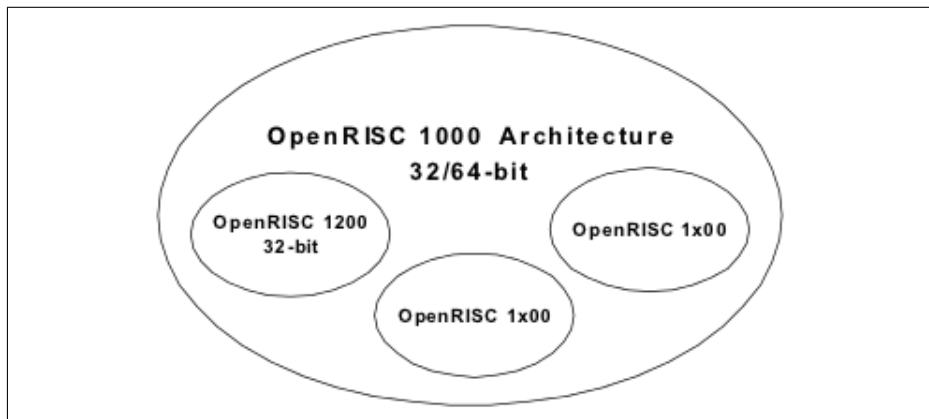
**Κατάλογος σχημάτων**

1	Σχηματική αναπαράσταση OpenRISC αρχιτεκτονικής. . . . .	2
2	Core's Architecture . . . . .	3
3	OR1200 Overview . . . . .	4
4	Συνολική ροή εξομοίωσης. . . . .	6

## 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/απόδοση και το είδος της εφαρμογής. Είναι μια 32/64-bit φορτώσης 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 αρχιτεκτονική στοχεύει σε μεσαία και υψηλή απόδοση δικτύωσης (networking), σε ενσωματωμένα, αυτοκινητοβιομηχανικά και φορητά υπολογιστικά περιβάλλοντα.



**Σχήμα 1:** Σχηματική αναπαράσταση OpenRISC αρχιτεκτονικής.

Όλες οι OpenRISC υλοποιήσεις που το πρώτο ψηφίο στον αριθμό ταυτότητας (identification number) είναι '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 υλοποίηση ήταν παραμετροποιημένη πριν χρησιμοποιηθεί σε πραγματική εφαρμογή.

### 2.1 OpenRISC 1200

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

Οι προκαθορισμένες κρυφές μνήμες είναι:

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

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

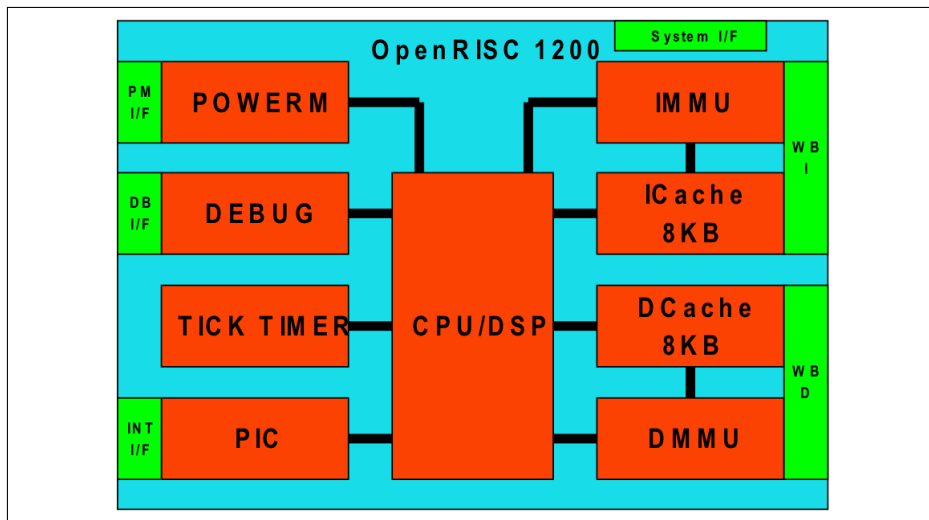
Η προκαθορισμένη MMU αποτελείται από:

- 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 είναι η μονάδα απασφαλμάτωσης πραγματικού χρόνου (real-time debug unit), υψηλής ανάλυσης χρονιστή, προγραμματιζόμενο ελεγκτή διακοπών (programmable interrupt controller) και μονάδα ρύθμισης των ενεργειακών απαιτήσεων.

Ο OR1200 ουσιαστικά προορίζεται για εφαρμογές σε ενσωματωμένα, 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.

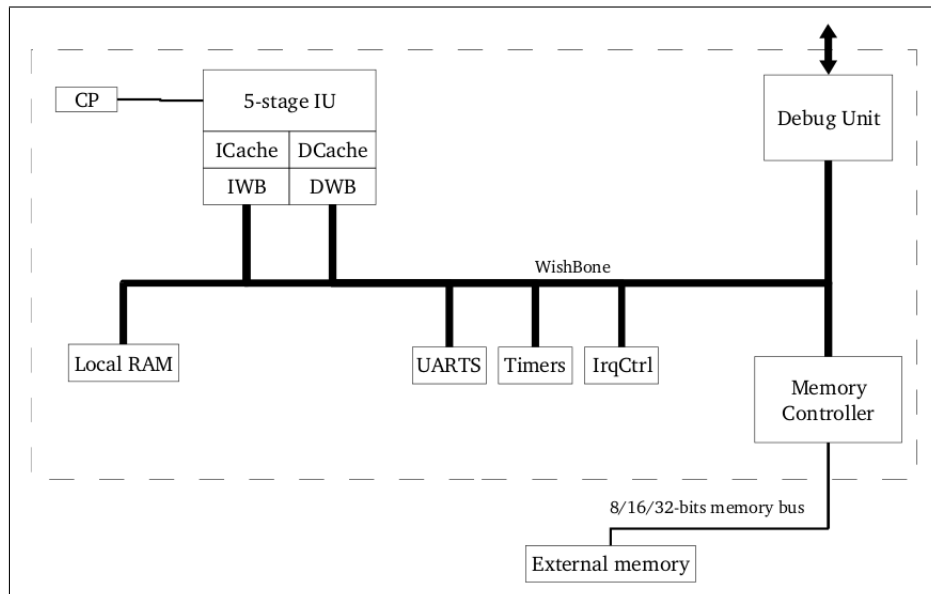
Παρακάτω βρίσκεται η σχηματική αναπαράσταση του OR1200 επεξεργαστή με τις υπομονάδες που τον αποτελούν.



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

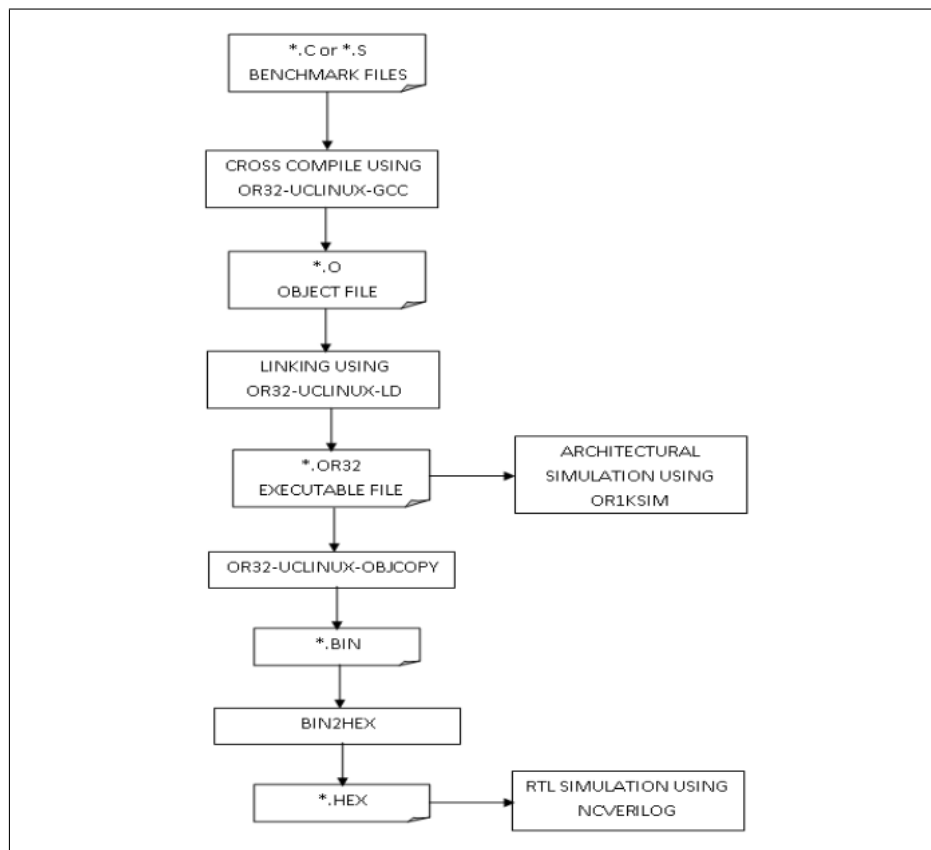
## 3 OR1200 Simulation

### 3.1 Simulation Enviroment

Υπάρχουν δύο περιβάλλοντα με τα οποία γίνεται η εξομοίωση του OR1200 επεξεργαστή. Το πρώτο χρησιμοποιεί τον OpenRISC αρχιτεκτονικό εξομοιωτή *or1ksim* και το δεύτερο χρησιμοποιεί τον *NC-Verilog* εξομοιωτή που κάνει εξομοίωση με βάση το υλικό (hardware based simulation). Στο πρώτο περιβάλλον γίνεται 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 αρχεία είτε .S αρχεία. Αύτα τα αρχεία, αρχικά γίνονται cross-compiled

χρησιμοποιώντας την εντολή `or32-uclinux-gcc` και παράγουν ένα `.O` object αρχείο. Το object αρχείο μετά μετατρέπεται σε ένα `.OR32` εκτελέσιμο αρχείο χρησιμοποιώντας την συνδυαστική εντολή `or32-uclinux-ld`. Αυτό το εκτελέσιμο αρχείο χρησιμοποιείται από τον `or1k` αρχιτεκτονικό εξομοιωτή. Περαιτέρω το αρχείο `.OR32` μετατρέπεται σε δυαδικό (binary) αρχείο χρησιμοποιώντας την εντολή `or32-uclinux-objcopy`. Στο τέλος δημιουργείται ένα `.HEX` αρχείο χρησιμοποιώντας τον `binary to hex` μετατροπέα `bin2hex`. Το παραγόμενο αρχείο `.HEX` φορτώνεται στην flash μνήμη του RTL κώδικα του OR1200 επεξεργαστή και μετά γίνεται 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 περιεχόμενα 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 πλακέτες 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 λειτουργία του επεξεργαστή.
  2. *sim/run* : Εδώ εκτελούνται όλες οι εντολές που σχετίζονται με την εξομοίωση του επεξεργαστή.
  3. *sim/out* : Εδώ τοποθετούνται όλα τα αρχεία που παράγονται μετά την εξομοίωση. Να σημειωθεί ότι τα waveforms τοποθετούνται στο *sim/run*.
- **Sw** : Εδώ βρίσκονται τα βασικότερα αρχεία που είναι απαραίτητα για το cross-compilation και την σωστή λειτουργία του OpenRISC επεξεργαστή.
  1. *sw/drivers* : Εδώ βρίσκονται οι drivers και τα εργαλεία για τροποποίηση του hardware.
  2. *sw/lib* : Εδώ βρίσκεται μια απλή βιβλιοθήκη που σε συνδυασμο με τους drivers κατά την διάρκεια του compile δημιουργούν την βιβλιοθήκη *liborpsoc* που τοποθετείται στο *sw/lib*.
  3. *sw/lib/include* : Εδώ βρίσκεται το αρχείο *cpu-utils.h* που περιέχει όλες τις συναρτήσεις σχετικές με την CPU του OpenRISC.
  4. *sw/tests* : Εδώ βρίσκεται το λογισμικό που χρησιμοποιείται (.C και .S αρχεία) για να δοκιμαστεί η σωστή λειτουργία του επεξεργαστή (testing) σε υπομονάδες όπως *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 υπάρχουν τα tests που εκτελούνται κατά την εξομοίωση του επεξεργαστή σε ένα PC και στον φάκελο *board* υπάρχουν τα tests που εκτελούνται κατά την λειτουργία του επεξεργαστή.
- **Doc** : Εδώ βρίσκεται documentation που χρειαζόμαστε για να καταλάβουμε την φύση των testbenches και οι οδηγίες για το simulation σύμφωνα με το Makefile που δημιουργήσαμε για τους σκοπούς του εργαστηρίου.

### 3.3 Simulation commands

#### 3.3.1 Η βασική διαδικασία

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

1. Το Makefile που ελέγχει το simulation βρίσκετε στο */sim/bin/* και είναι προσπελάσιμο και από το */sim/run/*.
2. Στον φάκελο */sim/run/* εκτελώντας την εντολή *make rtl-tests* κάνει 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/*. Αυτά είναι (στα αγγλικά 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 ενός συγκεκριμένου test

Η εξομοίωση ενός συγκεκριμένου test γίνεται με την εντολή *make rtl-test TEST=test-name*<sup>1</sup>. Πρέπει το αρχείο *test-name.c* (ή *test-name.s*) να είναι τοποθετημένο στο *sw/tests/module/sim/* όπου *module* είναι η υπομονάδα που θέλουμε να ελέξουμε με κάποιο από 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 γίνεται με την εντολή *make rtl-test TEST=" test-name1 test-name2 ..."* (πχ *make rtl-tests TESTS="sdram-rows uart-simple or1200-mmio 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 από 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 μετονομαστεί 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 από την αρχή test το software. Χρησιμοποιώντας την μεταβλητή *USER\_ELF* μπορούμε να καθορίσουμε το μονοπάτι στο οποίο βρίσκεται αυτό το αρχείο. Για παράδειγμα *make rtl-test USER\_ELF=/path/to/myapp.elf*. Το ELF αρχείο θα μετατραπεί σε δυαδική μορφή και μετά σε VMEM και θα φορτωθεί στο μοντέλο για να εκτελεστεί.

### 3.3.6 Κυματομορφές

Παράλληλα με το simulation ενός ή περισσότερων 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*.

<sup>1</sup> test-name: Είναι η ονομασία του test που θέλουμε να εκτελέσουμε.

### 3.3.7 Επιπρόσθετα επιλογές στις εντολές

Παρακάτω παρουσιάζονται μερικές 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 κατά την εκτέλεση μιας εξομοίωσης .Αυτό βοηθάει στην επιτάχυνση της εξομοίωσης αφού απαιτείται λιγότερος χρόνος στην εγγραφή αρχείων και αποτρέπει την δημιουργία πολύ μεγάλων αρχείων σε χρονοβόρες εξομοιώσεις.
- *SIMULATOR* :Επιλέγουμε τον εξομοιωτή υλικού που θέλουμε να χρησιμοποιήσουμε. Προκαθορισμένος εξομοιωτής είναι ο NC-Verilog.Άλλη επιλογή είναι το ICARUS.