Projet ANR MORE : Documentation développeurs

Jonathan Ponroy, Olivier Zendra

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

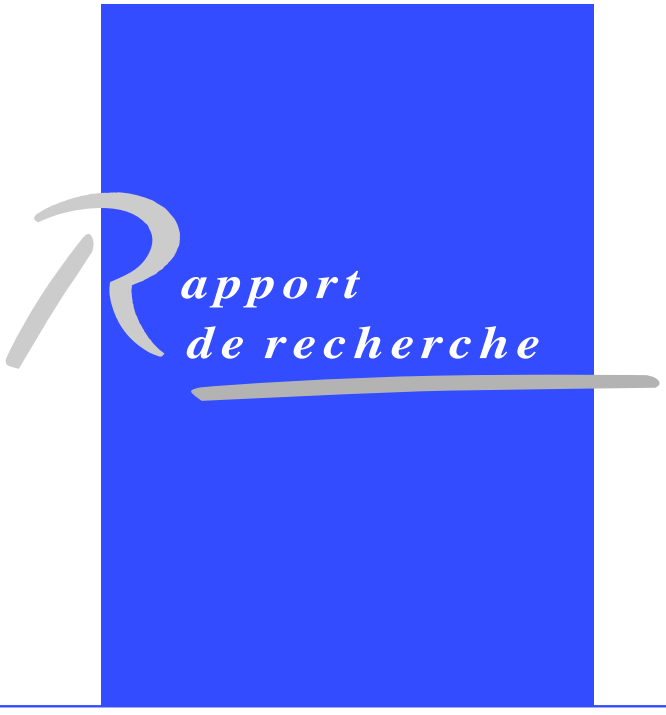
Projet ANR MORE : Documentation développeurs

Jonathan Ponroy, Olivier Zendra

N° 0397

Octobre 2010

Embedded System Code Optimization



ISSN 0249-6399

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche NANCY - GRAND EST

Projet ANR MORE : Technical documentation

Jonathan Ponroy[1], Olivier Zendra[2]

Thème Optimisations Multi Critère de code
Projet MORE

Rapport de recherche n° 0397 – Octobre 2010 - 31 pages

Abstract: The MORE (Multicriteria Optimizations for Real-time Embedded systems) project is aimed at developing a platform allowing iterative code optimization according to three criteria : code size, energy consumption caused by memory accesses and the worst cased execution time. This project is based on Otawa software platform from IRIT laboratory. We intend to describe first this project with commonly point of view then oriented developer.

Keywords: optimization, embedded systems, multi-criteria, technical documentation

[1] Inria Nancy Grand Est – jonathan.ponroy@inria.fr
[2] Inria Nancy Grand Est – olivier.zendra@inria.fr

Centre de recherche INRIA Nancy - Grand Est
615, rue du Jardin Botanique
B.P. 101
54602 Villers-lès-Nancy Cedex (France)
Téléphone : +33 03 83 59 30 00 — Télécopie : +33 03 83 27 83 19

Projet ANR MORE : Technical documentation

Résumé: Le projet MORE (Multicriteria Optimizations for Real-time Embedded systems) est un projet visant à proposer une plateforme permettant d'optimiser du code de façon itérative selon 3 critères et de façon conjointe : la taille du code, la consommation énergétique liée aux accès à la mémoire et le temps d'exécution pire cas. Ce projet repose sur la plateforme logicielle Otawa.de l'IRIT et que nous proposons dans ce document de décrire, tout d'abord de manière général puis orienté développeur.

Mots clés: optimisation, système embarqués, multi-critères, documentation technique

INRIA

Table des matières

1 Introduction

Ce document a pour but d'aider un développeur à acquérir les connaissances requises ainsi que les informations sur les logiciels utilisés et les développements réalisés au sein de l'INRIA Nan-

cy Grand Est. Celui-ci sera complété au fur et à mesure des prises de connaissances et des développements réalisés.

2 Information générale

Le projet MORE (Multicriteria Optimizations for Real-time Embedded systems) (http://www.irit.fr/recherches/ARCHI/MARCH/MORE/doku.php) est un projet visant à proposer une plateforme permettant d'optimiser du code de façon itérative selon 3 critères et de façon conjointe.
Ces trois critères sont :

- La taille du code
- la consommation énergétique liée aux accès à la mémoire
- Le temps d'exécution pire cas WCET (Worst-case execution time)

Dans le cadre de ce projet, 3 équipes se sont associées en apportant leurs connaissances :

- L'IRIT et l'équipe TRACES (http://www.irit.fr/recherches/ARCHI/MARCH/) qui est spécialisée dans l'étude du WCET. Cette étude est réalisée sur leur plateforme logiciel Otawa (www.otawa.fr) et sur laquelle les deux autres équipes vont insérer leur module. Hugues cassé (casse@irit.fr) et Christine Rochange (rochange@irit.fr) sont les développeurs d'Otawa dans l'équipe TRACE. Il gère le dépot ainsi que le site du projet MORE. Pour toutes questions techniques et création de comptes, c'est à eux qu'il faudra s'adresser.
- Le LIP6 et l'équipe ALSOC vont apporter leurs connaissances dans la compression de code. Karine Heydemann (karine.heydemann@lip6.fr) et Haluk Ozaktash développent le module de compression de code dans Otawa. Haluk Ozaktash a une bonne connaissance de Otawa et peut être interrogé pour obtenir des informations supplémentaires
- Le LORIA et l'équipe TRIO qui va apporter leurs connaissances dans l'étude de la consommation énergétique. L'étude et le développement sont menés par Olivier Zendra (Olivier.Zendra@inria.fr), Maha Idrissi Aouad (maha.idrissiaouad@inria.fr) et Jonathan Ponroy (jonathan.ponroy@inria.fr).

Comme vous l'aurez compris, la pierre angulaire du projet est la plateforme logicielle Otawa que nous allons maintenant décrire dans ce document d'abord de façon général puis plus orienté développeur.

3 La plateforme Otawa

Nous allons décrire de manière générale ce qu'est Otawa. Pour les personnes qui voudraient en savoir plus, vous pouvez visiter le site www.otawa.fr ou contacter les développeurs de l'équipe TRACES indiqués ci-dessus.
Otawa est une plateforme logiciel visant à simuler des plateformes matériels afin de mieux pouvoir étudier leurs comportements. A l'origine, elle a été développé pour étudier le WCET mais peut être étendu pour étudier d'autres aspects comme la consommation énergétique ou la compression de code.

4 SystemC

Pour réaliser cette simulation, Otawa repose sur un framework écrit en C++ appelé SystemC mais également sur d'autres librairies. SystemC est une bibliothèque, qui fournit des classes C++ adaptées à la modélisation de matériel ainsi qu'un moteur de simulation événementiel rapide. Ainsi, on peut simuler une porte combinatoire AND à 2 entrées comme le montre la Figure 1.

```
#include "systemc.h"
SC_MODULE(and2)
{
        sc_in<bool> e1;
```

```
        sc_in<bool> e2;
        sc_out<bool> s;

        void compute_and()
        {
                s = e1 & e2;
        };

        SC_CTOR(and2)
        {
                SC_METHOD(compute_and);
                sensitive(e1);
                sensitive(e2);
        }
};
```

Figure 1 : code en SystemC d'une porte NAND

Comme précisé sur la Figure 1, Otawa utilise des librairies pour pouvoir simuler les instructions PowerPC notamment GLISS qui a permit d'écrire facilement un simulateur d'instruction ppc et sur lequel se repose Otawa. En effet, Otawa n'éxécute pas réellement les instructions mais à pour but de simuler le fonctionnement de l'environnement matériel (cache, mémoire,...) ainsi que l'aspect temporel de l'éxécution (WCET).

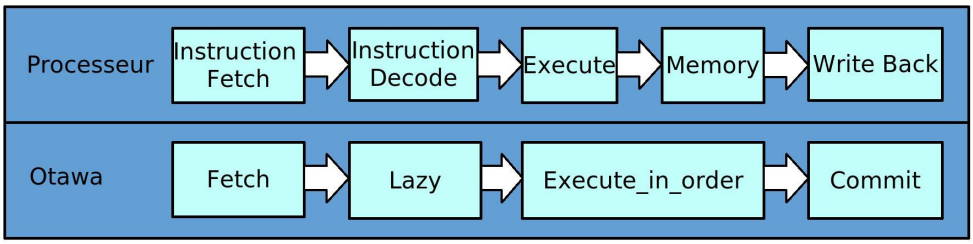


Figure 2. Pipeline d'un processeur et pipeline d'Otawa

Sur la Figure 2, vous pouvez voir la correspondance entre le pipeline d'instructions d'un processeur et le pipeline implémenté dans otawa. Ce pipeline est créé dans la fonction GenericProcessor::build se trouvant dans le fichier GenericProcessor.cpp. Cette fonction est chargée de créer les blocs ainsi que les liaisons entre eux en utilisant le framework SystemC.
La Figure 3 présente de façon plus précise les connections entre les blocs du pipeline selon les principes du SystemC.

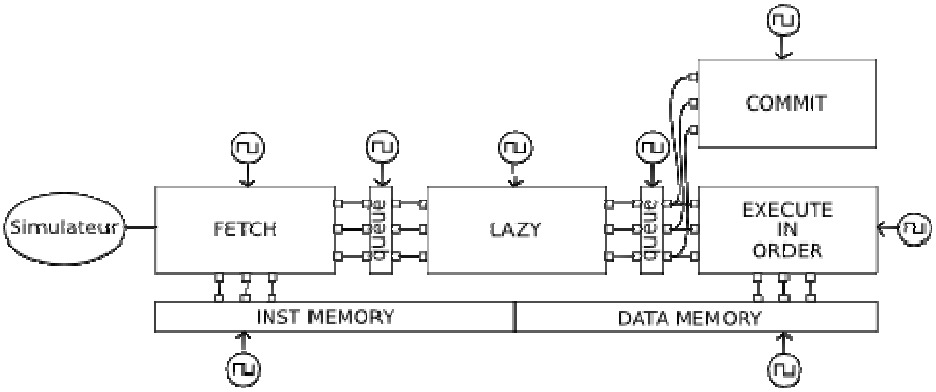


Figure 3 : Pipeline d'Otawa

Chacun des blocs est rythmé par une horloge SystemC comparable à une véritable horloge de processeur. A chaque front d'horloge, une instruction passe d'un bloc au suivant en passant par une queue. Le pipeline récupère les instructions du vrai simulateur (à gauche) qui éxécute réellement ces instructions.
L'implémentation de ces blocs se trouve dans le répertoire ogensim :

- FETCH dans Fetch.h/cpp
- LAZY dans LazyStage.h/cpp
- EXECUTE_IN_ORDER, EXECUTE_OUT_OF_ORDER et COMMIT dans Execute.h/cpp

Otawa permet donc au final de faire tourner du code powerPC par exemple sur une architecture donnée et de récupérer différentes informations via des features liés au WCET ou au accès de données en mémoire, etc. Une liste complète des features est disponible dans INSTALL_DIR/share/Otawa/autodoc.

5 Etude de l'énergie

Dans le cadre du projet MORE, le LORIA dois étudier la consommation d'énergie qui dépend d'une architecture mémoire et des règles qui la régisse. En général, une architecture mémoire se compose d'un processeur, d'un ou plusieurs niveaux de cache et d'une mémoire de masse comme le montre la Figure 4.

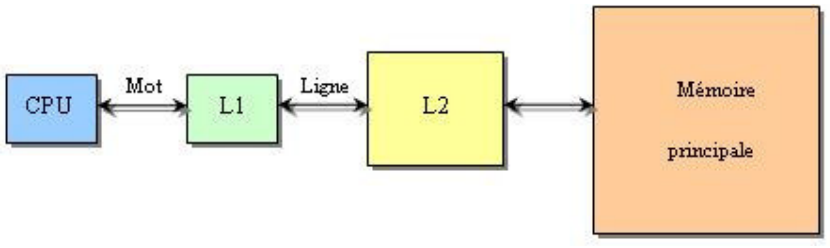


Figure 4 : Les niveaux de cache mémoire

La mémoire cache est petite mais rapide tandis que la mémoire principale est grande mais lente. Cette architecture permet d'accélérer en moyenne l'éxécution du code ainsi que l'accès aux données mais cette architecture à un coût énergétique. On va donc étudier la gestion de ces différentes mémoires et leurs coûts afin de les optimiser, le but étant de minimiser le coût énergétique global.

Le calcul du coût énergétique peut, dans sa version la plus simple, se résumer en la sommation de termes tels que nombre d'accès à une mémoire * coût énergétique d'accès à cette mémoire. Pour plus de précision sur les formules du calcul du coût, les différentes façon de gérer les différents niveau de mémoire, veuillez vous reporter au rapport du projet MORE disponible sur le site web MORE.

6 CACTI

Les informations nécessaires pour calculer le coût énergétique sont :

- le nombre d'accès mémoires
- le coût d'accès énergétique

La première donnée est, ou sera, obtenu grâce à Otawa. La deuxième donnée est fourni quant à elle par un logiciel du nom de CACTI. En lui fournissant les caractéristiques d'une mémoire, il peut nous fournir, entre autre, des données sur le coût énergétique d'une écriture et d'un lecture dans cette mémoire. C'est un outil utilisable en ligne de commande. Vous trouverez à la fin de ce document, en annexe, la manière d'utiliser CACTI.
Maintenant que nous avons décris de façon général le projet MORE et les différents outils utilisés, nous allons maintenant rentrer plus dans le détail de ce qui intéresse le développeur. Dans la suite nous allons donc voir comment récupérer OTAWA et ce qui a été développé dans le cadre du projet MORE, l'archicture de Otawa, comment compiler, etc.

7 Benchmarks et cross compilateur

Pour pouvoir utiliser le simulateur Otawa, il nous faut avoir des binaires powerPC (format supporté par Otawa). Dans le cadre du projet MORE, plusieurs binaires PPC (aussi appelé benchmark) regroupés en famille ont été sélectionné. Certains benchmarks ne sont pas utilisés par toutes les équipes. Voici, la liste des benchs est leur répartition par équipe (Tableau 1).

Famille	Bench	Equipe		
		LORIA	IRIT	LIP6
SN-RT	Fir	OK		
	Jfdctint	OK		
MiBenchs	Bitcount			
	gsm	OK		
	sha	OK		OK
Mälardalen	adpcm	OK	OK	OK
	cnt	OK		
	crc		OK	
	matmul		OK	
	compress	OK	OK	OK
Papabench	autopilot	OK		OK
	fly_by_wire	OK		OK
WCET_bench	Nsischneu	OK	OK	OK
	Statemate	OK		OK

Mediabench	Djpeg	OK		OK
Spec 2000	gzip	OK		OK

Tableau 1 : Benchmarks par équipe

7.1 Génération du cross-compilateur

Pour pouvoir générer ces benchmarks, il nous faut un cross compilateur, c'est à dire un logiciel qui permet de généré un code machine pour une architecture B alors que la compilation se fera sur une machine d'architecture A. Pour cela, il existe un script se trouvant sur la page du projet MORE, rubrique compilateur : build.sh (que l'on trouvera également sous le nom de crosstool_build.sh).Celui-ci va compiler un cross-compilateur, par exemple un gcc mais qui genera du code pour powerPC et qui suffira d'utiliser comme tout compilateur gcc.
Vous avez deux manière de le lancer qui reviennent exactement au même:

- ./crosstool_build.sh
- ./crosstool_build.sh bin_utils gcc_bootstrap newlib gcc_final gdb

Il est possible que vous rencontriez des difficultés lors de la construction du crosscompilateur. Dans ce cas, reportez vous à la section Difficultés rencontrées juste en dessous de ce paragraphe.
La deuxième méthode sera utile si vous rencontrez une erreur de type « problème de résolution de nom » à une des 5 étapes. Si, cela arrive lors de l'étape newlib par exemple, il suffira de taper ./crosstool_build.sh newlib gcc_final gdb pour terminer l'ensemble du processus. A la fin, dans le répertoire caché .gcc-4.1.2-newlib-1.15.0/powerpc-eabi/bin qui se trouve au même niveau que le script crosstool_build.sh, vous trouverez le cross-compilateur.

Difficultés rencontrées :

- **problème de version de makeinfo ou texinfo** : il est possible que soit installé une version trop récente de l'application texinfo (4.11 sur Ubuntu Intrepid) pour le crosscompilateur qui n'accepte que des versions entre 4.4 et 4.9 inclus. Dans ce cas, il vous faudra downgrader l'application texinfo. Pour ubuntu, vous pourrez trouver des versions antèrieurs ici. Pensez à désinstaller la version actuelle avant d'installer la version antérieure puis à bloquer la version pour éviter de mettre à jour texinfo lors d'une mise à jour du système.
- **warnings being treated as errors** : Lors de la compilation du cross compilateur, vous pourriez tomber sur cette erreur. Dès lors, à chaque warning, la compilation s'arrête. Ce la est dû au fait que l'option -WERROR soit présente dans les options de compilation. Pour pouvoir enlever cette option, vous devez rajouter dans la variable OPTS du script crosstool_build.sh, l'option –disable-werror.
- **Error : no termcap library found** : installer le package libncurses5-dev

7.2 Génération des benchmarks avec le cross-compilateur

Comme indiquer ci-dessus, nous allons compiler les benchmarks comme nous compilons n'importe quelles autres codes C/C++. C'est simplement le gcc pour powerPC que l'on vient de créer qui va remplacer le gcc habituel. On utilise les mêmes outils tel que le makefile.
Tout les benchmarks se trouvent sur le serveur CVS ainsi que sur le wiki MORE. Pour récupérer ces benchmarks sur le serveur CVS, il vous faudra demander un compte à Hugues Cassé. Une fois, celui-ci obtenu, il vous suffira de taper la ligne suivante pour récupérer l'ensemble des benchmarks:

```
cvs -d :pserver:login@cvs.irit.fr:2401/usr/local/CVS_IRIT/CVS_OTAWA co benchs
```

Chaque benchmark contient un répertoire sauf pour fly_by_wire et autopilot qui se trouvent dans le répertoire papabench. Dans chaque répertoire, vous trouverez un makefile pour générer le(s) benchmark(s) correspondant généré soit dans le répertoire des sources pour les plus simples ou dans un répertoire bin. Certain contiendront dans leur nom cette chaîne de caractère « O1.powerpc-eabi.bin ». A partir de ces informations, vous devriez réussir à retrouver les binaires de ces benchmarks.
Les makefile fonctionneront à conditions que la variable PATH soit définie de la façon suivant dans le fichier ~/.bashrc :

export PATH=[répertoire d'éxécution de crosstool_build.sh]/.gcc-4.1.2-newlib-1.15.0/powerpc-eabi/bin:$PATH

8 Otawa et le projet MORE

8.1 Pré requis logiciel

8.1.1 SystemC

Vous pouvez télécharger SystemC sur http://www.systemc.org. Vous devrez créer un compte gratuitement pour pouvoir le télécharger. (Le téléchargement peut être long : 1kio/s constaté). Puis compilez le en suivant les instructions du fichier INSTALL (./configure, make, make install) . Même si le make install échoue, l'important est d'avoir le répertoire include pour qu'OTAWA puisse utiliser systemc.
ATTENTION !!! : erreur de compilation concernant getenv, rajouter dans le fichier src/sysc/utils/sc_utils_ids.cpp les lignes suivantes:

```
#include <stdlib.h>
#include <string.h>
#include "sysc/utils/sc_report.h"
using namespace std;
```

et supprimer « std:: » devant getenv vers la ligne 110 environ.

8.1.2 EFENC

Pour l'installer, il suffit de faire une recherche du package « electric-fence » dans votre gestionnaire de package.

8.1.3 Autres packages nécessaires

Vous devrez chercher et installer les packages suivants :
Automake, bison, flex, libxml2-dev, libtool, patch

8.1.4 Difficultés rencontrées

Dans ./build-more-XXX.sh, possible nécessité de mettre à jour les lignes concernant les versions de automake, autoconf et libtool défini dans une variable CHECK

8.2 Récupérer et compiler Otawa

Logiquement, vous trouverez toutes les informations nécessaires dans l'intranet du site web de MORE, dans le tableau des dernières nouvelles, « Script de construction Otawa ». Si vous n'arrivez pas à accéder à l'intranet, il faut contacter Hugues Cassé (casse@irit.fr) pour obtenir un compte.
Il n'est pas nécessaire d'avoir un compte CVS pour télécharger Otawa. Par contre, pour enregistrer ses modifications sur le CVS, il faut alors demander un compte à Hugues Cassé.
Une fois cela fait, voici la procédure à suivre:

- Créer un répertoire dans lequel va être installer Otawa, par exemple ~/MORE.

- Copier le build dans le répertoire, et taper tar xvfz build-more-XXX.tgz
- Définir la variable CVSROOT à pserver:login@cvs.irit.fr/usr/local/CVS_IRIT/CVS_OTAWA : soit en tapant la commande « export CVSROOT=:pserver:login@cvs.irit.fr/usr/local/CVS_IRIT/CVS_OTAWA » en remplaçant login par celui donné par Ugues Cassé. Soit, en copiant cette même commande dans votre ~/.bashrc et en relançant un terminal pour recharger le bashrc.
- taper cvs login
- puis ./build-more-XXX.sh --dev --with-systemc=CHEMIN_DE_SYSTEMC en absolue où CHEMIN_DE_SYSTEMC pourrait être /user/Home/MORE/systemc/ (**!!!** Attention de ne pas oublier le caractère « / » à la fin du chemin de systemc)
- ATTENTION : ce script ne marche que si le login avec lequel vous vous connectez à la machine est le même que le login utilisé pour vous connecté à CVS. Si ce n'est pas le cas , il faudra y rajouter juste avant cvs_user=$LOGNAME la ligne LOGNAME=votrelogin

Ensuite, il suffit d'attendre... En cas de problème, vous pouvez demander à Hugues Cassé.

8.2.1 Ajout de source dans Otawa

Otawa utilise plusieurs outils pour pouvoir avoir une gestion simplifiée de la compilation. Automake autoconf permettent ainsi de générer automatiquement des makefiles à partir de fichier Makefile.am.

8.2.2 Ajout d'un fichier source dans un répertoire existant

Pour pouvoir ajouter un fichier dans un répertoire déjà existant et que celui-ci soit pris en compte dans le processus de compilation, il faut ajouter dans le fichier Makefile.am du répertoire où vous voulez ajouter le fichier, le nom de votre source (ex : monCode.cpp) dans la variable xxx_SOURCES.

```
AM_CXXFLAGS = @OTAWA_CXXFLAGS@ @SYSTEMC_CXXFLAGS@ -Wall
–O2

Bin_PROGRAMS=simuLoria

simuLoria_LDADD = @LOADER_LIBS@ @OTAWA_LIBS@
@OGENSIM_LIBS@ ../../LIP6/cacti/libcacti.la ../energy/libenergy.la –lpthread –lm

simuLoria_DEPENDENCIES = @LOADER_DEPS@ @OTAWA_DEPS@
@OGENSIM_DEPS@ ../../LIP6/cacti/libcacti.la

simuLoria_SOURCES = \
                    simuLoria.cpp \
                    InfosCacti.cpp
```

Figure 5 : Exemple de makefile

Attention : prenez bien soin de choisir l'ordre, surtout si une source A dépend d'un autre source B alors il faudra placer la source B avant la source A dans la liste pour qu'elle soit compiler en premier.

Ainsi, lors d'un prochain make, le nouveau fichier sera pris en compte.

8.2.3 Ajout d'un fichier dans un répertoire à créer

Pour pouvoir ajouter un fichier dans un répertoire que vous devez créer préalablement, il faudra de la même manière faire connaître le répertoire dans des fichiers de configuration. Voici la procédure à suivre :

- créer le répertoire que vous désirez : par exemple, créer src/MORE/monRepertoire
- aller dans le répertoire src/MORE
- dans le fichier Makefile.am, ajouter votre répertoire monRepertoire dans la variable SUBDIRS
- dans le répertoire src/MORE/monRepertoire, ajouter un fichier source. ex : monCode.cpp
- récupérer un Makefile.am se trouvant au même niveau et mettez le dans src/MORE/monRepertoire
- Dans ce fichier Makefile.am, mettez votre source monCode.cpp dans la variable xxx_SOURCES (supprimer les sources précédentes qui ne vous intéresse plus)
- Dans le fichier configure.ac se trouvant à la racine de Otawa, ajouter le Makefile que vous voudriez que automake génère en ajoutant dans la variable AC_CONFIG_FILES([...]) ceci : src/MORE/monRepertoire/Makefile

Si vous avez bien suivi cette procédure, en faisant un make soit à la racine du projet ou au niveau supérieur ou même en compilant dans Eclipse (cf. rubrique suivante), vous devriez voir apparaître votre nouveau Makefile qui aura servi à compiler votre fichier monCode.cpp.

8.3 Otawa dans Eclipse

Eclipse est un IDE multi langage et il est possible de travailler sur Otawa dans cet éditeur. Grâce à lui, vous pourrez gérer la compilation, l'exécution et la gestion collaborative CVS.

Pour importer Otawa dans Eclipse, vous devez suivre les étapes suivantes :

- cliquer sur file->new->C++ Project
- donner un nom de projet, par exemple, OtawaProject
- décocher « default location »
- dans le champ location, indiquer le répertoire de Otawa, par exemple, <install_dir>/otawa/otawa
- dans project type, choisir Makefile project -> Empty project
- dans Toolchain, choisir Linux GCC
- valider

Eclipse va alors charger tout Otawa dans le menu à gauche, charger les numéros de version des fichiers et logiquement déjà faire une première compilation. Si ce n'est pas le cas, il vous suffira de lancer la compilation par vous même.

Pour compiler le projet, il suffira d'aller dans le menu « project » et cliquez sur « Build All ». La compilation peut prendre plusieurs minutes.

8.3.1 Exécution sur un benchmark dans Eclipse

Comme précisé plus haut dans cette documentation, Otawa est un simulateur et celui-ci nécessite des paramètres pour le lancer tel que les fichiers de configuration du processeur et de la mémoire cache. Il est possible de le faire directement dans Eclipse.
Voici la démarche à suivre pour créer un run pour le bench fir:

- Allez dans le menu run puis runs configuration. Une fenêtre s'ouvre.

- Cliquez deux fois sur C/C++ Local Application. Vous verrez apparaître New_configuration
- A droite, dans le champ name et tapez « simuloria_fir »
- Dans le champ Project, cliquez sur le bouton browse et choisissez OtawaProject
- Dans le champ C/C++ application, cliquez sur le bouton Search project... et choisissez le simulateur simuLoria.
- Dans l'onglet arguments, dans le champ program arguments, taper les paramètres du simulateur. Par exemple, -p < chemin absolue du fichier config1.xml > -c <chemin absolue du fichier idcache.xml > < chemin absolue du fichier fir > Attention : ne pas mettre des caractères interprétés comme ~, car ceux ci ne sont pas interprété dans Eclipse

Maintenant, vous avez un run prêt à être lancé en un clic et vous pouvez en créer d'autres de la même manière.

9 Fonctionnement du pipeline processeur

9.1 Quelques précisions sur certains modules

9.1.1 Sérialisation/désérialisation

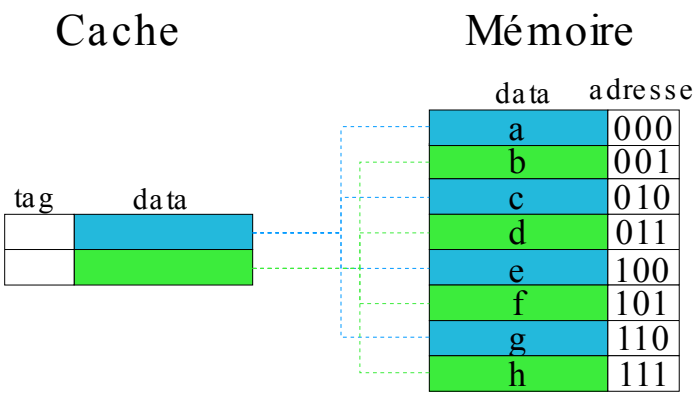
En plus de la documentation que l'on trouve à ce sujet dans Otawa, on peut ajouter que la sérialisation d'une classe nécessite que celle-ci possède un constructeur pas défaut ne prenant aucun paramètre ou des paramètres ayant une valeur par défaut.
Egalement, si la désérialisation xml->objet est possible, l'inverse ne l'est pas, tout du moins dans la branche MORE de ELM.

10 Fonctionnement de la mémoire cache

Dans ce paragraphe, nous allons expliquer comment fonctionne plus précisément la mémoire cache. On va donc supposer qu'il n'y qu'un cache, une mémoire de masse et un processeur. Il existe plusieurs types de cache et pour certain de ces caches, on peut appliquer certaines stratégies dite de remplacement.
Les 3 types de cache sont le Direct Mapped cache, le Full Associative cache et le N Way set associative cache qui est un mélange entre les deux premiers. Chaque comportement que l'on va indiquer dans la suite est intégralement « gravé » dans la puce avec des composants électronique (porte AND par exemple)

10.1 Direct Mapped Cache

Le direct mapped cache est un cache où chaque ligne du cache est associée à plusieurs lignes déterminées et fixe de la mémoire de masse. Ainsi, si j'ai une mémoire de 8 lignes et que je ne dispose que d'un cache de 2 lignes, j'obtiendrai une association comme sur la Figure 6.



Il y a 8=2^3 lignes dans cette mémoire donc l'adressage de la mémoire se fera sur 3 bits. Le processeur veut accéder à la donnée se trouvant à l'adresse 1O1. Le cache étant vide, il y a défaut de cache dit obligatoire étant donnée qu'il s'agit de la première demande. Le contrôleur de cache va donc faire une copie de la donnée en mémoire et d'une partie de son adresse dans le cache comme sur la Figure 7.

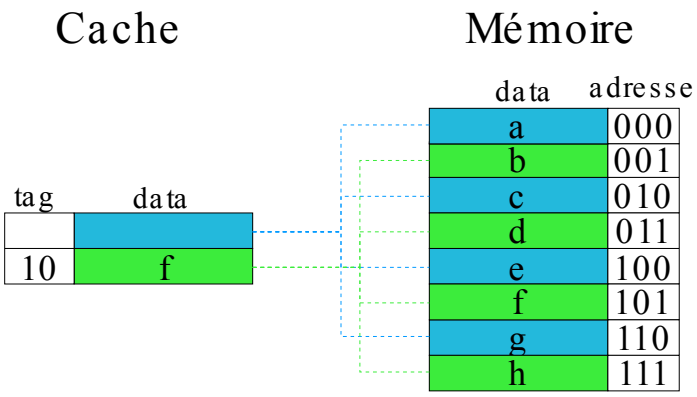


Figure 7 : Etat du cache après une demande de donné

Maintenant, imaginons que le processeur redemande la donnée se trouvant à l'adresse 101. La première chose que le contrôleur de cache va faire c'est de vérifier que la donnée ne se trouve pas déjà dans le cache. Pour cela, il va découper l'adresse en 2, le tag et l'index. L'index va lui permettre de retrouver la ligne du cache où la donnée pourrait éventuellement être. Il y a 2=2^1 ligne dans la cache donc 1 bit lui suffit pour connaître l'index de la ligne : le bit de poids le plus faible. Il s'agit de la ligne de cache d'adresse 1. Si on avait eu un cache de 4=2^2 lignes, il aurai fallu utiliser les 2 premiers bits de poids le plus faible pour connaître l'index de la ligne.
Une fois la ligne trouver, il suffit de comparer le reste de l'adresse appelé tag avec le tag se trouvant à la ligne 1 du cache. Si c'est le même, la donnée est déjà présente dans le cache sinon il y a défaut de cache.
En réalité, l'adresse ne sera pas découpée en 2 (tag et index) mais en 3 (tag, index et offset) car chaque ligne de la mémoire de masse contient en fait plusieurs octets et l'offset permet de récupérer un octet particulier.
Ici, il n'y a pas de stratégie de remplacement contrairement aux 2 autres types de cache.

10.2 Full Associative cache

Contrairement au Direct Mapped cache, dans le cas du Full associative cache, une ligne de la mémoire de masse peut se placer n'importe où dans le cache. Il n'y a pas de place prédéfinie. Donc lorsque l'on cherche à savoir si une donnée est disponible dans le cache, on va comparer le tag de l'adresse à toutes les entrées du cache et ceci, en parallèle. Ici, l'adresse n'est plus découpée qu'en 2 morceaux : le tag et l'offset.
Les défauts de cache sont corrigés en remplissant au fur et à mesure le cache et lorsque celui-ci est plein, il faut alors choisir d'écraser une donnée déjà présente dans le cache. C'est là qu'interviennent les stratégies de remplacement qui sont au nombre de 3 :

- La stratégie de choix aléatoire : la ligne est choisie de manière aléatoire parmi les lignes possible.
- La stratégie LRU (Least Recently Used) : La ligne choisie est celle qui a été la plus anciennement référencée.

- La stratégie FIFO (First In First Out) : les lignes de ma mémoires cache sont éffacées dans l'ordre où elles sont arrivées dans la mémoire cache

10.3 N-way Set Associative cache

Il s'agit d'un compromis entre les deux premières solutions. En fait, tout se passe comme si il y avait plusieurs Direct Mapped cache en parallèle. Prenons par exemple, un 2-way set associative cache. L'index va permettre de séléctionner un « ensemble » ou « set » du cache comme pour le Direct Mapped cache. Dans chaque ensemble par contre, le tag de l'adresse sera comparé aux adresses présentent dans cet ensemble à la manière d'un Full Associative cache. Bien entendu, la stratégie de remplacement est également présente dans ce type de cache.

10.4 Politiques d'écriture

Les accès en lecture sur le cache sont les plus nombreux (lecture d'opérandes, accès aux instructions) et ne posent pas de réels problèmes, alors que les accès en écriture, moins nombreux, nécessitent une gestion rigoureuse de la cohérence des données au travers de la hiérarchie mémoire. Là encore, diverses politiques sont mises en oeuvre lors de la modification d'une donnée :

- **Écriture simultanée** (write-through) : lorsqu'une ligne du cache est modifiée, cette ligne est écrite tout de suite dans un niveau mémoire inférieur de la hiérarchie. Cette politique garantit la cohérence entre les niveaux mais augmente le trafic sur le bus de données. Généralement, pour éviter que les écritures entravent le fonctionnement de l'UC, elles sont placées dans un tampon d'écriture ou write-buffer et effectuées pendant que l'UC se consacre à d'autres tâches.
- **Recopie** ou write-back : la ligne est marquée modifiée et elle sera recopiée en mémoire lorsqu'elle sera la cible d'un remplacement.

De plus, lorsque l'UC veut écrire une donnée dans une ligne absente du cache, l'une des deux options suivantes est généralement utilisée :

- **Écriture attribuée** ou write-allocate : la ligne est chargée dans le cache avant d'être modifiée.
- **Écriture non attribuée** ou no write allocate : la ligne est modifiée directement dans le niveau inférieur de la hiérarchie et non chargée dans le cache.

L'option write-allocate est généralement associée à la politique d'écriture write-back, et l'option no write-allocate à la politique d'écriture write-through (ce qui est cohérent dans la mesure où une écriture nécessitera sa mise à jour directement dans la mémoire, on évite ainsi des transferts supplémentaires).

10.5 Diagramme de classe du cache modélisé

Maintenant que nous avons décrit le fonctionnement du cache dans la réalité, nous allons décrire comment le cache est finalement modélisé. Ce diagramme de classe décrit le fonctionnement de la mémoire cache simulée.

Les classes de ce diagramme sont réparties dans des fichiers de la façon suivante :

- memory.h/cpp : définition et implémentation des classes **MemorySystem** et **MemoryStats**
- GenericState.h/cpp : définition et implémentation de la classe **GenericState**
- Cache.h : définition et implémentation de la classe **Cache**
- CacheDriver.h : définition de la classe **CacheDriver**
- AbstractCacheDriver.h : définition des classes **AbstractCacheDriver**, **LRUCacheDriver**, **FIFOCacheDriver** et **DirectMappedCacheDriver**
- sim_AbstracCacheDriver.cpp : implémentation des classes **AbstractCacheDriver**, **LRUCacheDriver**, **FIFOCacheDriver** et **DirectMappedCacheDriver**

La classe hard::Cache contient les caractéristiques d'un cache (capacité, nombre de voies, taille d'un block, etc). Les comportements, qui seront déduis des caractéristiques du cache, sont définis dans les classes xxxDriver.
Dans Otawa, seules les adresses mémoires sont utilisés pour simuler les accès aux caches. Ainsi, on trouvera dans la classe AbstractCacheDriver, un pointeur lines sur un tableau de tag de taille nombre de voie * nombre de lignes par voie.
Ensuite, dans les classes LRUCacheDriver, FIFOCaceDriver et DirectMappedCacheDriver sont définis les fonctions touch et replace qui seront chargées de réorganiser le tableau pour chaque type de cache. Ces fonctions sont implémentées pour simuler la stratégie FIFO et LRU. Par contre, on verra que touch n'est pas implémenté dans la classe DirectMappedCacheDriver car la stratégie de remplacement est intrinsèque pour ce type de cache.

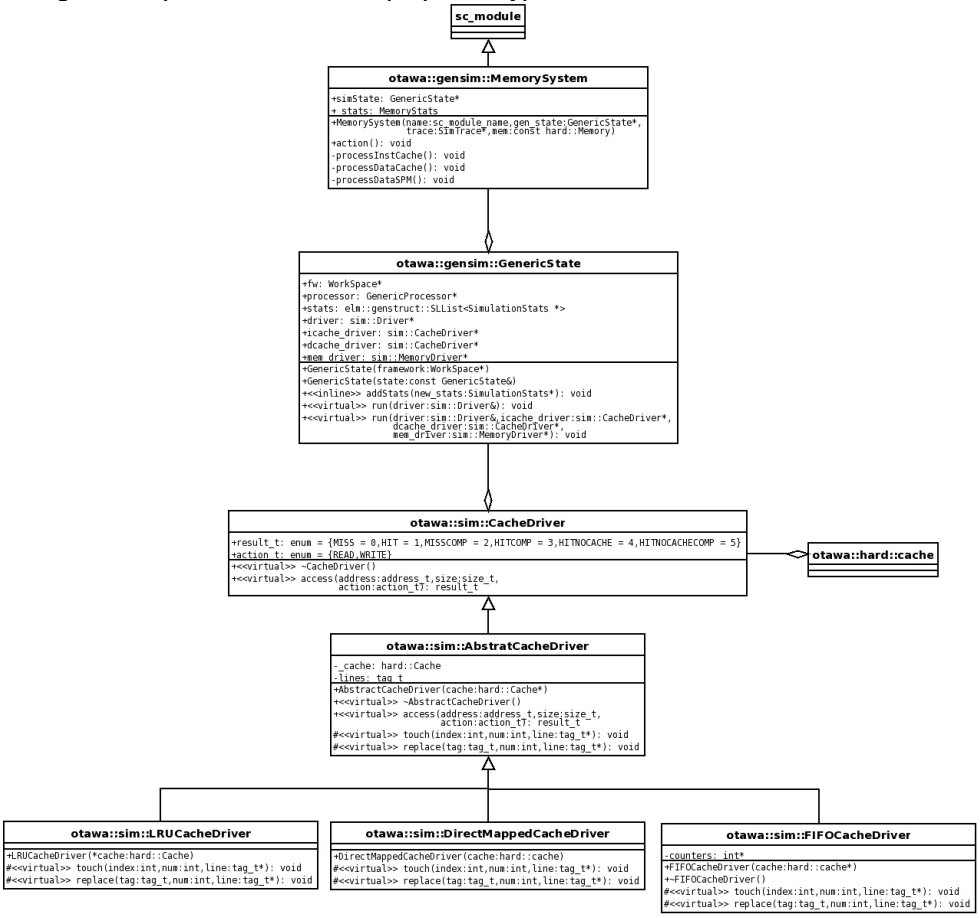


Figure 8 : Diagramme de classe de la gestion des caches mémoires

10.6 GenericState et MemorySystem

Quelques mots sur ces 2 classes. GenericState est une classe qui fournie des services pour contrôler la simulation en général. Elle contient le processeur, les différents drivers et une liste de classe de statistique (cf SimulationStats) qui vont évoluer durant la simulation. C'est également grâce à cette classe que l'on va pouvoir contrôler la simulation.

11 Implémentation du calcul de l'énergie

11.1 Implémentation de la hiérarchie mémoire

11.1.1 Fichier de configuration

Le fichier de configuration permet de définir la hiérarchie mémoire afin de pouvoir simuler celle-ci.
Vous pouvez voir la grammaire du fichier de configuration de la hiérarchie mémoire à cette adresse. Si vous rencontrez un problème de connexion, vous pouvez contacter Hugues Cassé.
Ci-dessous, vous pouvez voir un exemple de fichier de configuration respectant cette grammaire.

```
<?xml version="1.0" encoding="UTF-8"?>
<memory>
  <banks>
     <bank>
        <name>DRAM</name>
        <address>
           <space>0x00000000</space>
           <offset>0x00000000</offset>
        </address>
        <size>0x8FFFFFFF</size>
        <type>DRAM</type>
        <latency>12</latency>
        <power>10</power>
        <modes>
           <mode id="eveille">
              <name>eveille</name>
              <latency>11</latency>
              <static_power>11</static_power>
              <dynamic_power>11</dynamic_power>
              <transitions>
                 <transition>
                    <latency>5</latency>
                    <power>5</power>
                    <dest ref="endormi"/>
                 </transition>
              </transitions>
           </mode>
           <mode id="endormi">
              <name>endormi</name>
              <latency>15</latency>
               <static_power>3</static_power>
               <dynamic_power>4</dynamic_power>
               <transitions>
                 <transition>
                    <latency>5</latency>
                    <power>5</power>
                    <dest ref="eveille"/>
                 </transition>
              </transitions>
           </mode>
        </modes>
        <block_bits>4</block_bits>       //puissance de 2
        <cached>true</cached>
        <on_chip>true</on_chip>
        <writable>true</writable>
        <port>
           <number>1</number>
```

```
            <bus ref="Bus1"/>
         </port>
      </bank>
      <bank>
         <name>SCRATCH PAD MEMORY</name>
         <address>
            <space>0x00000000</space>
            <offset>0xA0000000</offset>
         </address>
         <size>0x0FFFFFFF</size>
         <type>SPM</type>
         <latency>10</latency>
         <power>10</power>
         <block bits>4</block bits>
         <cached>false</cached>
         <on_chip>true</on_chip>
         <writable>true</writable>
         <port>
            <number>1</number>
            <bus ref="Bus1"/>
         </port>
      </bank>
      <bank>
         <name>non-volatile memory</name>
         <address>
            <space>0x00000000</space>
            <offset>0xB0000000</offset>
         </address>
         <size>0x00000100</size>
          <type>ROM</type>
          <latency>10</latency>
         <power>10</power>
          <block_bits>4</block_bits>    //puissance de 2
         <cached>false</cached>
         <on_chip>true</on_chip>
         <writable>false</writable>
         <port>
            <number>1</number>
            <bus ref="Bus1"/>
         </port>
      </bank>
   </banks>
   <buses>
      <bus id="Bus1">
         <name>Bus1</name>
         <type>LOCAL</type>
      </bus>
   </buses>
</memory>
```

Figure 9 : Fichier de configuration de la hiérarchie mémoire

Ce fichier doit respecter également d'autres règles. Par exemple, deux bancs de même type de mémoire doivent avoir les mêmes caractéristiques (cached, on_chip, etc). Pour deux bancs de type DRAM, ceux-ci doivent être juxtaposés dans l'espace d'adressage et donc ne pas être séparés par un banc de type différent. Ces règles sont les conséquences d'un choix techniques qui aurait pu être tout autre.

11.1.2 Diagramme de classe

Pour simuler la hiérarchie mémoire en terme de structure, on a ajouté à la classe **GenericState**, qui contient déjà 2 CacheDriver pour le cache d'instructions et le cache de données, 2 classes **SPMDriver** et **DRAMDriver** correspondant à la SPM et à la DRAM. Ces 2 classes héritent de

MemoryDriver. On pourra ajouter par la suite deux classes supplémentaires comme IODriver et ROMDriver si on le désire. Comme vous pouvez le remarquer, **l'AbstractCacheDriver** a connaissance du **DRAMDriver** afin de pouvoir correctement simuler le comportement du cache vis à vis de la mémoire de niveau supérieur lors d'un miss.

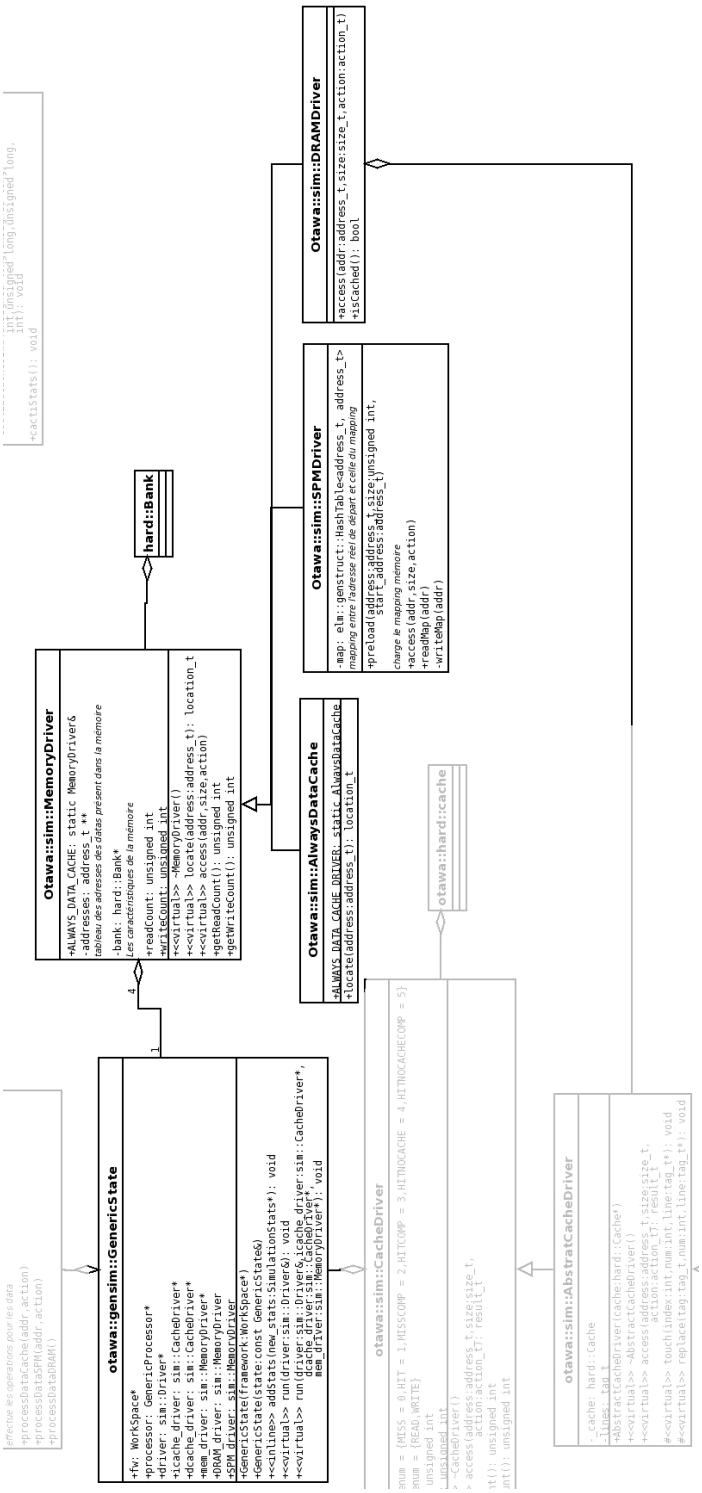


Figure 10 : diagramme de classes de gestion de la mémoire de masse

Enfin, chaque entité mémoire (cache, SPM, DRAM) possède deux compteurs, **readCount** et **writeCount** qui sont incrémentés lors de l'appel à la méthode access. Ces compteurs sont accessibles par des accesseurs afin de pouvoir calculer l'énergie consommée.

11.2 SPM et statistique des données

Dans la SPM existe, pour simplifier, deux conteneurs. Un conteneur contient les adresses des données qui sont disponible dans la SPM, et une map qui fait la correspondance entre les adresses d'entrées qui n'auront pas pu être mis dans la SPM et leurs adresses de placement dans la DRAM.
Avant de pouvoir charger la SPM, la première étape est de pouvoir enregistrer l'ensemble des adresses des accès aux données (détaillé dans le paragraphe Guide d'utilisation de l'outil de calcul de l'énergie), des tailles d'accès et des types d'action. Une fois que celles-ci sont enregistré, elles pourront être chargés à travers l'objet DataAccessStatistic et à partir du quelle, on pourra obtenir une liste d'adresses triées selon différents critères. Les critères sont :

- firstUsed : les adresses sont placées par ordre d'apparitions
- frequency : les adresses sont placées par ordre décroissant du nombre d'apparitions
- size : les adresses sont placées par ordre croissant de la taille d'accès
- numberonsize : les adresses sont placées par ordre décroissant de la division du nombre d'apparitions sur la taille de l'accès

Une fois obtenu cette liste triée, on peut demander à pré charger la SPM grâce à la fonction preload() de la classe **SPMDriver**.

11.3 Calcul de l'énergie

Pour calculer l'énergie consommée lors de l'utilisation d'un bench, nous utilisons une application, Cacti dont les sources ont été intégrées à Otawa par le LIP6 et qui a réalisé une interface d'utilisation. Cette interface nous permet d'obtenir le coût unitaire énergétique en lecture et en écriture de chaque entité mémoire.
Le calcul de l'énergie consommée est réalisé grâce à la classe **EnergyEstimator** chargée de récupérer les coûts unitaires énergétiques ainsi que des compteurs de lecture/écriture de chaque entité. Le coût total revient alors à une somme de :

cout unitaire lecture/écriture * compteurs de lecture/écriture

12 Création des diagrammes à partir des résultats sous Open Office

Ce qui suit est un résumé non exhaustif qui permet de créer un diagramme à partir des données grâce au langage de macro Open Office Basic (ressemblant énormément à VB).
Voici un exemple de données brutes à traiter

AL1:BC65536 | Energy Decomp

	A	B	AK
1	Bench	Parameters	Energy Tota
2	/home/more/Bureau/MORE/benchs/bitcount/src/bitcnts-O1.powerpc-eabi.bin	100	32722.
3	/home/more/Bureau/MORE/benchs/bitcount/src/bitcnts-O1.powerpc-eabi.bin	300	36946.
4	/home/more/Bureau/MORE/benchs/bitcount/src/bitcnts-O1.powerpc-eabi.bin	500	41073.
5	/home/more/Bureau/MORE/benchs/bitcount/src/bitcnts-O1.powerpc-eabi.bin	700	45236.
6	/home/more/Bureau/MORE/benchs/bitcount/src/bitcnts-O1.powerpc-eabi.bin	900	49428.
7	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/80x60xtype1.jpg.ppm ./data_djpeg/80x60xtype1.jpg	79853
8	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/160x120xtype1.jpg.ppm ./data_djpeg/160x120xtype1.jpg	2.72E+00
9	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/320x240xtype1.jpg.ppm ./data_djpeg/320x240xtype1.jpg	1.05E+00
10	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/640x480xtype1.jpg.ppm ./data_djpeg/640x480xtype1.jpg	4.13E+00
11	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/80x60xtype2.jpg.ppm ./data_djpeg/80x60xtype2.jpg	36894
12	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/160x120xtype2.jpg.ppm ./data_djpeg/160x120xtype2.jpg	1.19E+00
13	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/320x240xtype2.jpg.ppm ./data_djpeg/320x240xtype2.jpg	4.15E+00
14	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/640x480xtype2.jpg.ppm ./data_djpeg/640x480xtype2.jpg	1.68E+00
15	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/80x60xtype3.jpg.ppm ./data_djpeg/80x60xtype3.jpg	74594
16	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/160x120xtype3.jpg.ppm ./data_djpeg/160x120xtype3.jpg	2.48E+00
17	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/320x240xtype3.jpg.ppm ./data_djpeg/320x240xtype3.jpg	9.24E+00
18	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/640x480xtype3.jpg.ppm ./data_djpeg/640x480xtype3.jpg	3.53E+00
19	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/80x60xtype4.jpg.ppm ./data_djpeg/80x60xtype4.jpg	81674
20	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/160x120xtype4.jpg.ppm ./data_djpeg/160x120xtype4.jpg	2.69E+00
21	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/320x240xtype4.jpg.ppm ./data_djpeg/320x240xtype4.jpg	9.51E+00
22	/home/more/Bureau/MORE/benchs/jpeg/bin/djpeg-O1.powerpc-eabi.bin	-dct int -ppm -outfile ./data_djpeg/640x480xtype4.jpg.ppm ./data_djpeg/640x480xtype4.jpg	3.67E+00
23	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/11ko.txt	44317
24	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/11ko.htm	86388
25	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/11ko.exe	1.34E+00
26	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/11ko.jpg	1.51E+00
27	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/30ko.txt	83098
28	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/30ko.htm	2.00E+00
29	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/30ko.exe	2.79E+00
30	/home/more/Bureau/MORE/benchs/gzip/src/gzip-O1.powerpc-eabi.bin	./data_gzip/30ko.jpg	3.84E+00

Figure 11 : Données brutes issu d'un tableur

Nous voulons dans cet exemple traiter les données concernant le bench bitcounts (ligne 2 à 6) est obtenir le diagramme de la Figure 12:

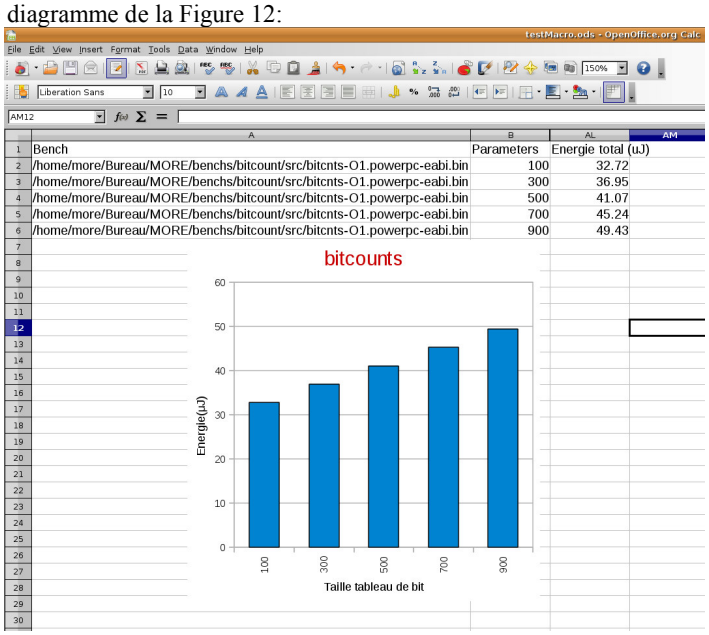


Figure 12: Diagramme produit par une macro

Voici la macro qui permet de créer le diagramme ci dessus ainsi qu'un deuxième diagramme qui traite le bench gzip. Le code est suffisamment explicite pour que vous compreniez ce qui s'y passe.

Voici quelques ressources qui peuvent vous permettre de mieux comprendre le code:
http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html
http://ooo.developpez.com/faq/?page=Calc
http://wiki.services.openoffice.org/wiki/Documentation/BASIC_Guide

```
Sub Main

   ' add bench here
   Dim benchNbr as integer
   benchNbr = 2
   Dim benchList(0 to benchNbr-1) As string
   Dim benchPatterns(0 to benchNbr-1) As string
   benchList(0)="bitcounts"
   benchPatterns(0)="bitcnts"
   dataRange = ThisComponent.Sheets(0).GetCellRangeByName("A1:AK" &
dataNbr(ThisComponent.Sheets(0)))

   For I = 0 To benchNbr-1
      filteringData(benchList(I), benchPatterns(I), dataRange)
   next I

   computeBitcount(benchList(0))
   computeGZIP(benchList(1))

End Sub

sub clean
   'For I = 0 To benchNbr-1
   ' getByIndex()
   'next I
end sub

'hide columns selected by the range
sub hideColumns(aera as New com.sun.star.table.CellRange)
   aeraAddr = aera.getRangeAddress()
   For I = aeraAddr.StartColumn To aeraAddr.EndColumn

   ThisComponent.getSheets().getByIndex(aeraAddr.sheet).Columns(I).IsV
isible = false
   Next I
end sub

'compute the count of data
Function dataNbr(sheet as object)
   Dim oCurseur As Object
   oCurseur = sheet.createCursor
   oCurseur.gotoEndOfUsedArea(False)
   dataNbr = oCurseur.RangeAddress.EndRow+1
end Function

'filtering and copy in new sheet
sub filteringData(benchName as string, benchPattern as string,
dataRange as Object)
   Dim aDestAddr As New com.sun.star.table.CellAddress
   Dim aFilterFields(0) As new com.sun.star.sheet.TableFilterField

   If not ThisComponent.Sheets.hasByName(benchName) Then
```

```
  ThisComponent.sheets.insertNewByName(benchName,thisComponent.sheets
.getCount)
  End if

  sheet = ThisComponent.getSheets().getByName(benchName)
  aFilterDesc = dataRange.createFilterDescriptor(True)
  'destination address
  Dim destCell As Object
  destCell = sheet.getCellByPosition( 0, 0 )
  ' filter definition
  ' filter column
  aFilterFields(0).Field    = 0
  aFilterFields(0).IsNumeric = False
  aFilterFields(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
  aFilterFields(0).StringValue = ".*" & benchPattern & ".*"

  ' filter setting up
  aFilterDesc.setFilterFields( aFilterFields() )
  aFilterDesc.OutputPosition = destCell.getCellAddress ' destination
address
  aFilterDesc.CopyOutputData = True     'with data copy
  aFilterDesc.ContainsHeader = True   ' with column label
  aFilterDesc.SkipDuplicates = True ' without duplicate data
  aFilterDesc.UseRegularExpressions = true

  dataRange.filter(aFilterDesc)

end sub

'create a data pilot from a data range to a destination cell
sub createDataPilot(dataRange as Object, destCell As Object)
  dataRangeAddr = dataRange.getRangeAddress()
  sheet=ThisComponent.getSheets().getByIndex(dataRangeAddr.sheet)
  benchName=sheet.getName()

  ' get the DataPilotTables collection
  Dim DPTables As Object
  DPTables = sheet.getDataPilotTables()
  ' create a DataPilot descriptor
  Dim DPDesc As Object
  DPDesc = DPTables.createDataPilotDescriptor()
  DPDesc.setSourceRange(dataRangeAddr)
  DPDesc.setPropertyValue("IgnoreEmptyRows",true)
  DPDesc.setPropertyValue("ShowFilterButton",false)
  DPDesc.setPropertyValue("ColumnGrand",false)
  DPDesc.setPropertyValue("RowGrand",false)

  ' get the DataPilot fields
  Dim Fields As Object
  Fields = DPDesc.getDataPilotFields()
  ' define one field for properties
  Dim aFieldObj As Object
  ' set the 1st col as row field
  aFieldObj = Fields.getByIndex(0)
  aFieldObj.setPropertyValue("Orientation",
com.sun.star.sheet.DataPilotFieldOrientation.COLUMN)

  ' set the 2nd col as row field
  aFieldObj = Fields.getByIndex(1)
  aFieldObj.setPropertyValue("Orientation",
com.sun.star.sheet.DataPilotFieldOrientation.ROW)
```

```
   'set the 3rd col as data field (no calc its SUM)
   aFieldObj = Fields.getByIndex(2)
   aFieldObj.setPropertyValue("Orientation",
com.sun.star.sheet.DataPilotFieldOrientation.DATA)
   'aFieldObj.setPropertyValue("Function",
com.sun.star.sheet.GeneralFunction.SUM)

   cellAddress =  destCell.getCellAddress

   if DPTables.hasByName(benchName) then
      DPTables.getByName(benchName)
   else
      DPTables.insertNewByName(benchName, cellAddress, DPDesc)
   end if

end sub

sub computeBitcount(sheetName as string)
   sheet = ThisComponent.getSheets().getByName(sheetName)
   aera = sheet.getCellRangeByName( "C1:AK1" )
   hideColumns(aera)
   Cell = sheet.getCellRangeByName("AL1")
   Cell.String = "Energie total (uJ)"
   Cell = sheet.getCellRangeByName("AL2")
   Cell.Formula = "=AK2/1000"
   aera = sheet.getCellRangeByName( "AL2:AL" & dataNbr(sheet))
   aera.fillseries(com.sun.star.sheet.FillDirection.TO_BOTTOM,com.sun.
star.sheet.FillMode.SIMPLE,0,1,0)

   aera = sheet.getCellRangeByName( "B1:AL" & dataNbr(sheet))
   graph(sheet.getName(),aera.getRangeAddress(),true,"Taille tableau
de bit",true,"Energie(µJ)", false)
end sub

sub computeGZIP(sheetName as string)
   sheet = ThisComponent.getSheets().getByName(sheetName)
   aera = sheet.getCellRangeByName( "C1:AK1" )
   hideColumns(aera)

   Cell = sheet.getCellRangeByName("AL1")
   Cell.String = "type"
   Cell = sheet.getCellRangeByName("AL2")
   Cell.Formula = "=MID(B2;LEN(B2)-3;3)"
   Cell = sheet.getCellRangeByName("AM1")
   Cell.String = "taille"
   Cell = sheet.getCellRangeByName("AM2")
   Cell.Formula =
"=VALUE(MID(B2;SEARCH(""[:digit:]*ko"";B2);SEARCH(""ko"";B2)-
SEARCH(""[:digit:]*ko"";B2)))"
   Cell = sheet.getCellRangeByName("AN1")
   Cell.String = "Energie total (µJ)"
   Cell = sheet.getCellRangeByName("AN2")
   Cell.Formula = "=AK2/1000"

   aera = sheet.getCellRangeByName( "AL2:AL" & dataNbr(sheet))
   aera.fillseries(com.sun.star.sheet.FillDirection.TO_BOTTOM,com.sun.
star.sheet.FillMode.SIMPLE,0,1,0)
   aera = sheet.getCellRangeByName( "AM2:AL" & dataNbr(sheet))
   aera.fillseries(com.sun.star.sheet.FillDirection.TO_BOTTOM,com.sun.
star.sheet.FillMode.SIMPLE,0,1,0)
   aera = sheet.getCellRangeByName( "AN2:AL" & dataNbr(sheet))
   aera.fillseries(com.sun.star.sheet.FillDirection.TO_BOTTOM,com.sun.
star.sheet.FillMode.SIMPLE,0,1,0)
```

```
  dataRange = sheet.getCellRangeByName( "AL1:AN" & dataNbr(sheet))
  destCell = sheet.getCellRangeByName("AP1")
  createDataPilot(dataRange,destCell)

  oCurseur = sheet.createCursorByRange(destCell)
  'oCurseur.gotoEnd()
  oCurseur.collapseToCurrentRegion()
  Cible = oCurseur.getRangeAddress()

  Cible.StartRow = Cible.StartRow + 1

  graph(sheet.getName(),Cible,true,"Taille(Ko)",true,"Energie(µJ)",
true)
end sub

' create a graph from a data range
sub graph(benchName as string, dataRangeAddr as Object,
firstColIsLabel as boolean, xlabel as string, firstRowIsLabel as
boolean, ylabel as string, hasLegend as boolean)
  'dataRangeAddr = dataRange.getRangeAddress()
  sheet=ThisComponent.getSheets().getByIndex(dataRangeAddr.sheet)
  Dim Rect As New com.sun.star.awt.Rectangle
  Dim Source(0) As New com.sun.star.table.CellRangeAddress
  Dim oCharts As Object , oChart As Object
  'sheet where we will create the graph
  oCharts = sheet.Charts
  '--- specify position and size of graph ---
  Rect.X = 5000 'horizontal
  Rect.Y = 5000 'vertical
  Rect.Width = 10000 'Largeur
  Rect.Height = 10000 'Hauteur

  Source(0) = dataRangeAddr

  'Création du graphique
  if ocharts.hasByName(benchName) then
     oCharts.getByName(benchName)
  else
     oCharts.addNewByName(benchName,Rect ,Source(),True , True)
  end if



  'Place le graphique dans la feuille de calcul
  oChart = oCharts.getByName(benchName).EmbeddedObject

  With oChart
     'Type de graphique (Scatters)
     .Diagram =
oChart.createInstance("com.sun.star.chart.BarDiagram")

     'Lissage des lignes (0: pas de lissage, 1: spline cubique, 2:
spline B)
     .Diagram.SplineType = 0
     'Change le symbole des points
     .Diagram.SymbolType =
com.sun.star.chart.ChartSymbolType.SYMBOL1


     'Dim col As New com.sun.star.awt.Gradient
     'col.Style = com.sun.star.awt.GradientStyle.LINEAR
     'col.StartColor = RGB(255,0,0)
```

```
      'col.EndColor = RGB(0,255,0)
      'col.StartIntensity = 150
      'col.EndIntensity = 150
      'col.Angle = 450
      'col.StepCount = 100

      'dataSeries = .Diagram.GetDataRowProperties(0)

      'dataSeries.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
      'dataSeries.FillGradient = col


      'Modifie la couleur de fond du graphique
      .Diagram.wall.FillColor =RGB(150,150,150)

      'Spécifie la valeur maxi de l'étiquette des ordonnées
      '.Diagram.YAxis.Max = 50000

      'Attribue un titre à l'axe des abscisses
      .Diagram.HasXAxisTitle = True
      .Diagram.XAxisTitle.string = xlabel

      'Attribue un titre à l'axe des ordonnées
      .Diagram.HasYAxisTitle = True
      .Diagram.YAxisTitle.string = ylabel

      'La première ligne(colonne) contient les étiquettes des séries
      .DataSourceLabelsInFirstColumn = firstColIsLabel
      .DataSourceLabelsInFirstRow = firstRowIsLabel

      'Roration des étiquettes pour l'axe des abscisses
      .Diagram.XAxis.TextRotation = 9000 '90 degrés

      'redéfinit la taille des étiquettes pour les abscisses et les
ordonnées
      .Diagram.YAxis.CharHeight = 8
      .Diagram.XAxis.CharHeight = 8

      'Attribue un titre au graphique puis un couleur à la chaîne de
caractères
      .HasLegend = hasLegend
      .HasMainTitle = True
      .Title.String = benchName
      .Title.CharColor = RGB(200,0,0)
      .Title.CharHeight =16

  End With
end sub
```

Figure 13 : Macro de generation des diagrammes

13 Documentation Doxygen dans Otawa

14 Guide d'utilisation de l'outil de calcul de l'énergie

L'utilisation de l'outil consiste en une ligne de commande. Nous allons considérer 2 cas, un sans SPM et l'autre avec SPM.
Dans le cas sans SPM, voici des exemples de commande :

- ./simulatorConso --processorConfig config1.xml --cacheConfig idcache.xml ~/benchs/fir/fir

Lance le simulateur avec la configuration processeur config1.xml, la configuration des caches idcache.xml, avec une hiérarchie mémoire par défaut composée uniquement d'une DRAM de 4Go, sur le bench fir

- ./simulatorConso --processorConfig config1.xml --cacheConfig idcache.xml --memoryConfig memory.xml ~/benchs/fir/fir

Lance le simulateur comme précédemment mais avec une hiérarchie mémoire donnée que l'on supposera sans SPM.

Lorsque vous lancez une de ces commandes vous obtiendrais un affichage ressemblant à ceci :

```
Cache path: ./MORE/LORIA/simuLoria/idcache.xml

Info: (I804) /IEEE_Std_1666/deprecated: sc_sensitive_neg is
deprecated use sc sensitive << with neg() instead

Info: (I804) /IEEE_Std_1666/deprecated: sc_sensitive_pos is
deprecated use sc_sensitive << with pos() instead
----------run-------------
j'ai une hiérarchie mémoire par défaut

Info: (I804) /IEEE_Std_1666/deprecated: sc_start(double) deprecated,
use sc_start(sc_time) or sc_start()

         Stats from the cache system
**********************************************************************
                            Instruction Cache       Data Cache
----------------------------------------------------------------------
Number of cache accesses     14361                  3472
Number of cache hits         14236                  3382
Number of cache misses       125                    90
Number of cache reads        14361                  2429
Number of cache writes       0                      1043
nbr of c. read misses dirty  0                      0
nbr of c. write misses dirty 0                      0
----------------------------------------------------------------------

         Stats from the memory
**********************************************************************
                          SPM       DRAM
----------------------------------------------------------------------
Number of accesses                     680
Number of READ accesses              680
Number of WRITE accesses             0
Number of SPM successes
Number of SPM fails
----------------------------------------------------------------------

            Cacti energy unit (in nJ)
**********************************************************************
                READ             WRITE
----------------------------------------------------------------------
icache :           0.0407833        0.0440015
dcache :           0.00572382       0.00516604
spm :              no specified     no specified
DRAM :          44.7483          44.7421
----------------------------------------------------------------------

            Energy estimation (in nJ)
**********************************************************************
                   ALL          READ             WRITE
```

```
---------------------------------------------------------------------
energy total :     31033.8
energy CPU :       0
energy BUS :       0
energy memory :        31033.8       31028.4       5.38818
energy icache :        585.689       585.689       0
energy dcache :        19.2913       13.9032       5.38818
energy SPM :
energy DRAM :      30428.8       30428.8       0
energy decomp :    0
```

Figure 14 : Exemple de sortie du simulateur de consommation

Dans le cas où l'on veut utiliser une SPM, il faudra alors procéder en 2 étapes. La 1er étape consiste à utiliser l'option « --accessDump targetFile.access » qui permet d'enregistrer les accès aux données dans un fichier :

- ./simulatorConso --processorConfig config1.xml --cacheConfig idcache.xml --accessDump ~/benchs/fir/data.access ~/benchs/fir/fir

Cette commande va enregistrer dans le fichier la suite des adresses des accès aux données dans le fichier ~/benchs/fir/data.access

La 2ème étape consiste maintenant à précharger le fichier avant de lancer la simulation grâce à l'option « --loadAccess targetFile.access --spmPolicy [firstU-sed|highFrequency|smallSizeFirst|highFrequencyOnSize] » :

- ./simulatorConso --processorConfig config1.xml --cacheConfig idcache.xml --memoryConfig memory.xml --loadAccess ~/benchs/fir/data.access --spmPolicy high-Frequency ~/benchs/fir/fir

Cette commande va lancer la simulation en chargeant la SPM (qu'on suppose configuré dans le fichier memory.xml) à partir du fichier ~/benchs/fir/data.access dont les adresses auront été triées par nombre d'apparitions.

Vous obtiendrez un affichage qui ressemblera à ceci :

```
MEMORY
- BANKS
   0:  SCRATCH PAD MEMORY 1
       address = 0x00000000|0x00000000, BlockSize = 4, size = 100, SPM
       on chip, writable
   1:  DRAM 1
       address = 0x00000000|0x00001000, BlockSize = 4, size = 1000000,
DRAM
       cached, on chip, writable

- BUSES
   0:  Bus1, LOCAL
Opening of /home/maha/Bureau/MORE/benchs/fir/data.access

Info: (I804) /IEEE_Std_1666/deprecated: sc_sensitive_neg is
deprecated use sc_sensitive << with neg() instead

Info: (I804) /IEEE_Std_1666/deprecated: sc_sensitive_pos is
deprecated use sc_sensitive << with pos() instead
----------run------------
j'ai une hiérarchie mémoire

Info: (I804) /IEEE_Std_1666/deprecated: sc_start(double) deprecated,
use sc_start(sc_time) or sc_start()
```

```
        Stats from the cache system
**********************************************************************
                            Instruction Cache       Data Cache
----------------------------------------------------------------------
Number of cache accesses    14361                   0
Number of cache hits        14236                   0
Number of cache misses      125                     0
Number of cache reads       14361                   0
Number of cache writes      0                       0
nbr of c. read misses dirty 0                       0
nbr of c. write misses dirty 0                      0
----------------------------------------------------------------------

        Stats from the memory
**********************************************************************
                            SPM          DRAM
----------------------------------------------------------------------
Number of accesses             4328          500
Number of READ accesses     3041         500
Number of WRITE accesses    1287         0
Number of SPM successes     3472
Number of SPM fails         0




            Cacti energy unit (in nJ)
**********************************************************************
             READ         WRITE
----------------------------------------------------------------------
icache :        0.0407833    0.0440015
dcache :        0.00572382   0.00516604
spm :           0.00167341   0.00119395
DRAM :       13.6486      13.6425
----------------------------------------------------------------------

            Energy estimation (in nJ)
**********************************************************************
                   ALL       READ      WRITE
----------------------------------------------------------------------
energy total :     7416.6
energy CPU :       0
energy BUS :       0
energy memory :       7416.6    7415.06   1.53661
energy icache :       585.689   585.689   0
energy dcache :       0         0         0
energy SPM :       6.62545   5.08884   1.53661
energy DRAM :      6824.28   6824.28   0
energy decomp :  0
```

Figure 15 : Autre exemple de sortie du simulateur de consommation d'énergie