Estructura/Arquitectura de Computadores

Grado de Informática

Simulación

Departament d'Enginyeria Informàtica I Matemàtiques

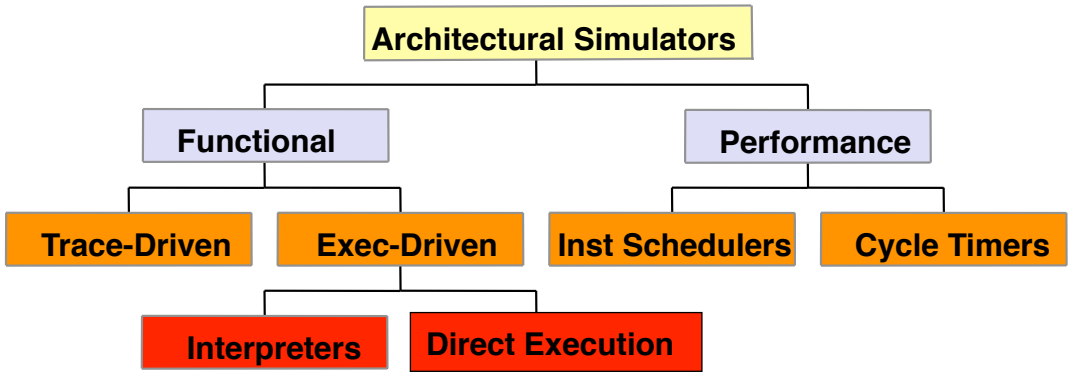
Universitat Rovira i Virgili

Tarragona, Spain

Índice

Simuladores

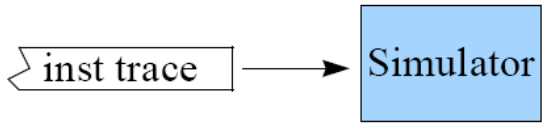
- **Clasificación de Simuladores**



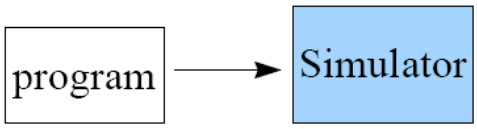
Functional vs Performance

- **Functional Simulators**
  - Implementan la arquitectura (ISA)
  - Realizan una ejecución real
  - Implementan lo que el programador ve
- **Performance Simulators**
  - Implementan la microarquitectura (implementación del ISA)
  - Modelan los recursos del sistema
  - Preocupados por el tiempo
  - Implementan lo que el programador no ve
  - También llamados timing simulators

Trace-Driven vs Execution Driven



- **Trace-Driven**
  - Simulador lee traza de instrucciones captada en ejecución previa
  - Fácil de implementar
  - No hay información, ni evaluación de ejecución especulativa
  - No se modela un comportamiento real del todo



- **Execution-Driven**
  - Simulador ejecuta el programa. No hay ejecución previa
  - Difícil de implementar
  - Permite la evaluación de la ejecución especulativa
  - Se valida un comportamiento real

Schedulers vs Cycle Timers

- **Instruction Schedulers**
  - Simulador lanza instrucciones cuando recursos están disponibles
  - Las instrucciones se procesan una a una
  - Simple pero menos detallado

- **Cycle Timers**
  - Simulador sigue el estado de la microarquitectura ciclo a ciclo
  - Estado del simulador = estado de la microarquitectura
  - Perfecto para la simulación de la microarquitectura

Índice

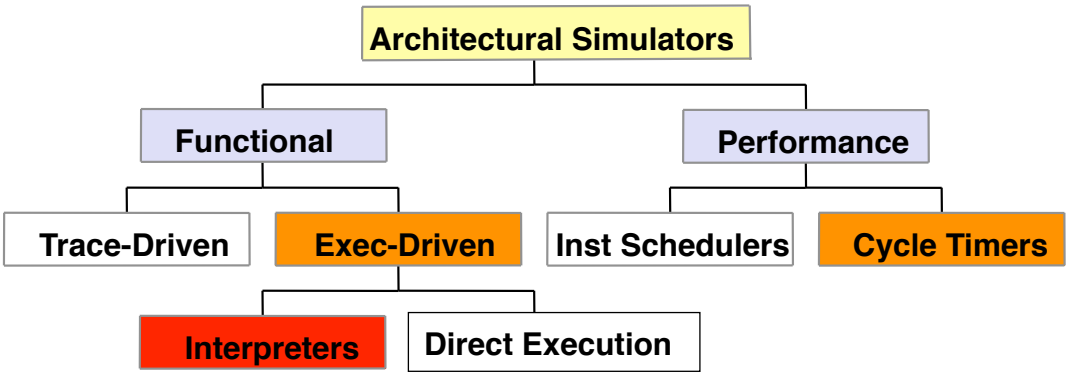
Visión General

Introducción

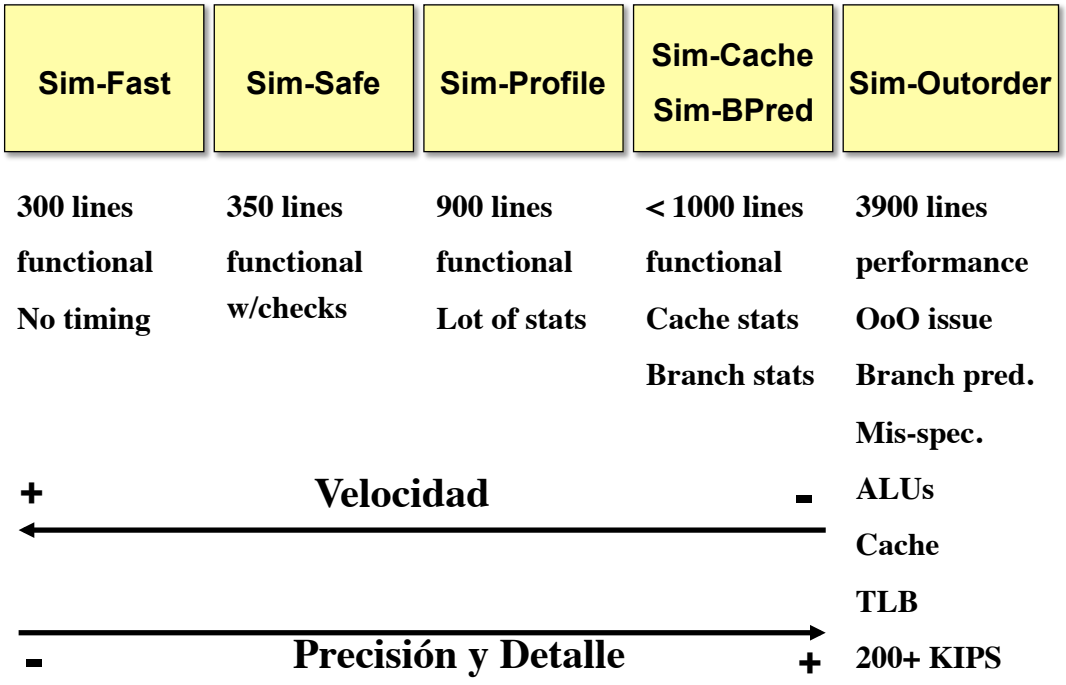
- **Conjunto de herramientas para la simulación**
  - desde simples procesadores moniclo
  - a complejos procesadores superescalares
    - ejecución fuera de orden
    - jerarquía de memoria completa
- **Universidad de Wisconsin-Madison**
- **Licencia libre y código abierto para fines académicos**
- **Ampliamente usado por la comunidad investigadora**
- **Limitaciones:**
  - Single core
  - No proporciona datos sobre consumo

Tipo de Simulador

- **En color las opciones incluidas en Simplescalar**



Simplescalar Suite



Sim-Fast, Sim-Safe

- **Sim-Fast**
  - Simulador funcional
  - Optimizado en velocidad
  - No asume jerarquía de memoria
  - No valida las instrucciones
- **Sim-Safe**
  - Simulador funcional
  - Optimizado en velocidad
  - No asume jerarquía de memoria
  - Valida las instrucciones

Sim-Profile, Sim-BPred

- **Sim-Profile**
  - Más que simulador es una herramienta de profiling
  - Permite la recolección de información de profiling
  - Genera informes detallados sobre:
    - Tipos de instrucción
    - Tipos de saltos
    - Tipos de accesos a memoria
    - etc
- **Sim-Bpred**
  - Simulador centrado en parte específica del procesador
  - Analiza predictores de saltos
    - taken, not taken, perfect, bimodal, 2level, hybrid
  - Genera estadísticas de aciertos y fallos del predictor
  - No se reflejan los efectos del predictor en el tiempo de ejecución

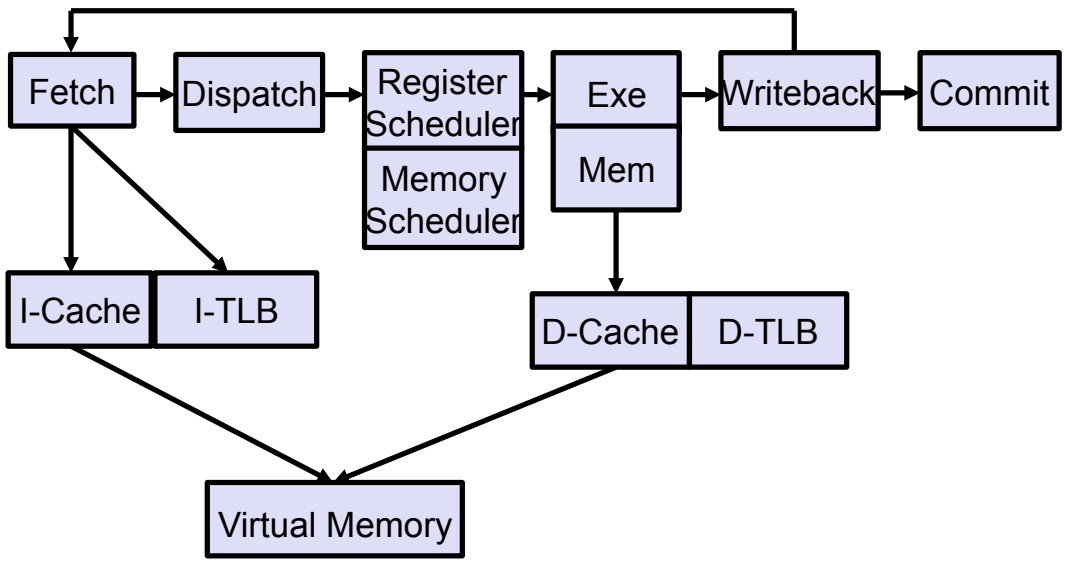
Sim-Cache

- **Simulador específico y rápido**
- **Centrado en la jerarquía de memoria del procesador**
- **Genera estadísticas de aciertos y fallos en jerarquía**
- **No refleja efectos de memoria en tiempo de ejecución**
- **Permite simular:**
  - 2 niveles de caché
  - separar (o no) datos de instrucciones
  - TLB
  - Vaciado (flush) y compresión
- **Ideal si sólo nos centramos en memoria**

Sim-Outorder

- **El simulador más complicado y detallista**
- **Intenta simular todas las partes de un procesador**
  - Jerarquía de memoria
  - Predictor de saltos
  - Ejecución fuera de orden
  - Banco de registros
  - Estaciones de reserva
  - Buffers de memoria
- **Permite parametrizar la mayoría de sus componentes**
- **Se puede modificar el código para extender/añadir nuevas funcionalidades**

Sim-Outorder (Hardware)



Instalación

Pasos de Instalación

- **Obtener última versión de `www.simplescalar.com`**
  - simplesim-3v0e.tgz
- **Descomprimir**
  - tar xvf simplesim-3v0e.tgz
- **Configurar uno de dos ISA disponibles**
  - make config-pisa
  - make config-alpha
- **Compilar todas las herramientas**
  - make
- **Ahora ya se pueden realizar simulaciones**
  - sim-outorder, sim-cache, sim-fast, etc

Ejecución y Configuración

Ejecutar Simulador

- **El formato para ejecución por línea de comandos es:**

```
simulador {-parametros} benchmark
 {argumentos} >& output_file_name
```

- **Simulador**
  - sim-cache, sim-outorder, etc
- **Formato de los parámetros de simulación:**

```
-parametroX valor
```

  - Se ponen por línea de comandos uno detrás de otro
- **Benchmarks**
  - spec2006, minibench, etc

Parámetros Generales

- **`-config nom_fichero`**
  - permite poner todos los parámetros en un fichero
  - en el fichero cada parámetro va en una línea diferente
- **`-fastfwd valor`**
  - número de instrucciones iniciales saltadas sin simular
  - las instrucciones se ejecutan pero no recolectan estadísticas
  - sólo disponible en sim-outorder
- **`-max:inst valor`**
  - máximo número de instrucciones ejecutadas y simuladas
- **`-redir:sim nom_fichero`**
  - redirecciona salida del simulador en un fichero
- **`-redir:prog nom_fichero`**
  - redirecciona salida del benchmark en un fichero

Parámetros de Memoria (I)

Parámetro	Argumento	Valor por Defecto
-cache:dl1	<string>	dl1:128:32:4:l
L1 data cache configuration {<config>|none} (see below).		
-cache:dl1lat	<int>	1
L1 data cache hit latency (cycles).		
-cache:dl2	<string>	ul2:1024:64:4:l
L2 data cache configuration {<config>|none} (see below).		
-cache:dl2lat	<int>	6
L2 data cache hit latency (cycles).		

Parámetros de Memoria (II)

Parámetro	Argumento	Valor por Defecto
-cache:il1	<string>	il1:512:32:1:l
L1 inst cache configuration {<config>|dl1|dl2|none} (see below).		
-cache:il1lat	<int>	1
L1 instruction cache hit latency (cycles).		
-cache:il2	<string>	dl2
L2 instruction cache configuration {<config>|dl2|none} (see below).		
-cache:il2lat	<int>	6
L2 instruction cache hit latency (cycles).		

Parámetros de Memoria (III)

Parámetro	Argumento	Valor por Defecto
-cache:flush	<true|false>	false
Flush caches on system calls.		
-cache:icompres	<true|false>	false
Convert 64-bit inst addresses to 32-bit inst equivalents.		
-mem:lat	<int list...>	18 2
Memory access latency (<first_chunk> <inter_chunk>).		
-mem:width	<int>	8
Memory access bus width (bytes).		

Parámetros de Memoria (IV)

Parámetro	Argumento	Valor por Defecto
-tlb:itlb	<string>	itlb:16:4096:4:l
Instruction TLB configuration {<config>|none} (see below).		
-tlb:dtlb	<string>	dtlb:32:4096:4:l
Data TLB configuration {<config>|none} (see below).		
-tlb:lat	<int>	30
Inst/data TLB miss latency (cycles).		

Configuración de Caché

`<name>:<nsets>:<bsize>:<assoc>:<repl>`

- **<name>**
  - nombre de la cache que se define (il1, dl1, il2, dl2, ul2, tlb)
- **<nsets>**
  - número de sets/conjuntos de la caché
- **<bsize>**
  - tamaño de bloque de la cache (bytes)
- **<assoc>**
  - asociatividad de la caché
- **<repl>**
  - política de reemplazo: 'l'-LRU, 'f'-FIFO, 'r'-random

Configuración de Caché

- **Ejemplos**
  - `-cache:dl1 dl1:4096:32:1:l`
  - `-dtlb dtlb:128:4096:32:r`
- **Los niveles de caché se pueden unificar**
  - **Ejemplo 1:** 2o nivel de memoria caché está unificado

    ```
    -cache:il1 il1:128:64:1:l
    -cache:il2 dl2
    -cache:dl1 dl1:256:32:1:l
    -cache:dl2 ul2:1024:64:2:l
    ```

  - **Ejemplo 2:** 1er nivel de memoria caché está unificado

    ```
    -cache:il1 dl1
    -cache:dl1 ul1:256:32:1:l
    -cache:dl2 ul2:1024:64:2:l
    ```


Estadísticas Resultados

Estadísticas Simulador (Generales)

sim_num_insn	total number of instructions committed
sim_num_refs	total number of loads and stores committed
sim_num_loads	total number of loads committed
sim_num_stores	total number of stores committed
sim_num_branches	total number of branches committed
sim_elapsed_time	total simulation time (seconds)
sim_total_insn	total number of instructions executed
sim_total_refs	total number of loads and stores executed
sim_total_loads	total number of loads executed
sim_total_stores	total number of stores executed
sim_total_branches	total number of branches executed
sim_cycle	total simulation time (cycles)
sim_IPC	instructions per cycle
sim_CPI	cycles per instruction

Estadísticas Simulador (Memoria)

cache.accesses	total number of accesses
cache.hits	total number of hits
cache.misses	total number of misses
cache.replacements	total number of replacements
cache.writebacks	total number of writebacks
cache.invalidations	total number of invalidations
cache.miss_rate	miss rate (misses/reference)
cache.repl_rate	replacement rate (replacements/reference)
cache.wb_rate	writeback rate (wrbks/ref)
cache.inv_rate	invalidation rate (invs/ref)

- **Estas estadísticas se repiten para:**
  - il1, dl1, il2, dl2, ul2, itlb, dtlb

Índice

I. Simuladores

II. Simplescalar

III. Benchmarks

IV. Cacti

Test Benchmarks

- Simplescalar incorpora benchmarks de prueba
  - simplesim-3.0/tests-pisa/
  - test-math, test-printf, etc
- Pocas instrucciones, ejecución instantánea
- Sirven para validar el funcionamiento del simulador
- Ejemplos de ejecución

```
sim-cache simplesim-3.0/tests-pisa/bin.little/test-math
```

- Se puede probar simulador y todos los tests

```
make sim-tests
```

Spec CPU2000 (Overview)

- **SPEC: System Performance Evaluation Cooperative**
- **Sociedad sin ánimo de lucro**
- **Misión:**
  - establecer, mantener y distribuir un conjunto estandarizado de *benchmarks* que se pueden aplicar a las últimas generaciones de procesadores
- **Spec CPU2000:**
  - enfocado a sistemas de computadores de carácter general
  - representativos del 2000 al 2006
- **Se dividen en dos grupos**
  - SpecINT: programas con cálculos de aritmética entera
  - SpecFP: programas con cálculos de aritmética en coma flotante

Spec CPU2000 (CINT)

Benchmark	Description
164.gzip	Compression
175.vpr	FPGA place and route
176.gcc	C compiler
181.mcf	Combinatorial optimization
186.crafty	Chess
197.parser	Word processing, grammatical analysis
252.eon	Visualization (ray tracing)
253.perlbmk	PERL script execution
254.gap	Group theory interpreter
255.vortex	Object-oriented database
256.bzip2	Compression
300.twolf	Place and route simulator

Spec CPU2000 (CFP)

Benchmark	Description
168.wupwise	Physics/Quantum Chromodynamics
171.swim	Shallow water modeling
172.mgrid	Multi-grid solver: 3D potential field
173.applu	Parabolic/elliptic PDE
177.mesa	3-D graphics library
178.galgel	Computational Fluid Dynamics
179.art	Image Recognition/Neural Networks
183.equake	Seismic Wave Propagation Simulation
187.facerec	Image processing: face recognition
188.ammp	Computational chemistry
189.lucas	Number theory/primality testing
191.fma3d	Finite-element Crash Simulation
200.sixtrack	High energy nuclear physics accelerator design
301.apsi	Meteorology: Pollutant distribution

Cargas de Trabajo (WorkLoads)

- **Cada benchmark dispone de 3 posibles "workloads"**
  - test
  - train
  - reference
- **TEST: ejecuta alrededor de 500 milones de Insts.**
- **TRAIN: ejecuta alrededor de 5000 millones de Insts.**
- **REFERENCE: ejecuta alrededor de 50.000 millones**
- **Lo ideal sería ejecutar todo el REFERENCE**
- **Habitualmente se coge un conjunto representativo**
  - Se saltan X millones (por ejemplo 200)
  - Se evalúan Y millones (por ejemplo 500)

Índice

I. Simuladores

II. Simplescalar

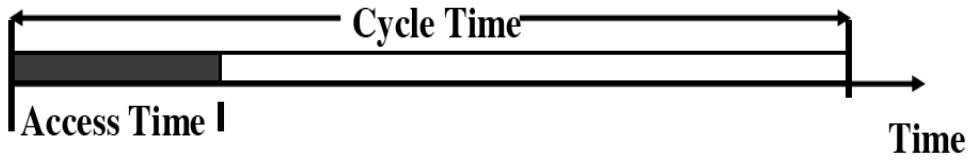
III. Benchmarks

IV. Cacti

CACTI (Visión General)

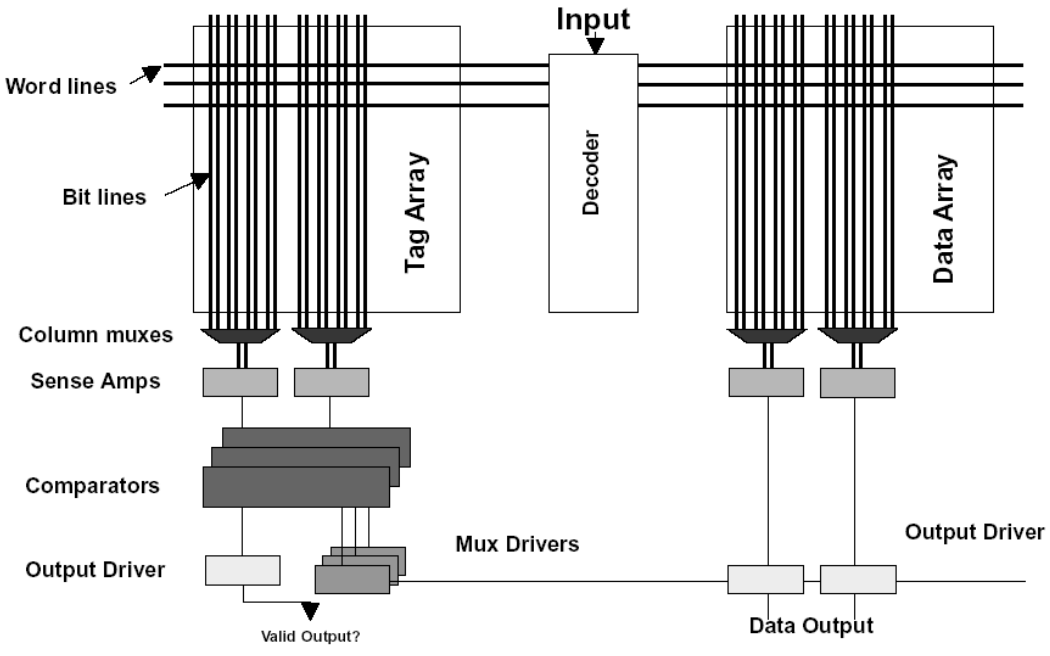
- Herramienta desarrollada por N. Jouppi en HP Labs
- Permite la evaluación de diseños de memoria caché
- A partir de unas características determinadas:
  - tamaño total, asociatividad, tamaño de bloque, tecnología, etc
- Modela la caché y calcula los valores de:
  - tiempo de acceso y tiempo de ciclo
  - área
  - consumo estático y dinámico
- Home Page
  - http://www.hpl.hp.com/research/cacti/
- Se proporciona en dos formas
  - Aplicación web interactiva
  - Código libre en C++

Tiempo Acceso vs Tiempo Ciclo



- **Tiempo de Ciclo de Caché**
  - La frecuencia a la que se puede iniciar un acceso
  - Analogía: cuando un niño puede pedir su paga (ej: los sábados)
- **Tiempo de Acceso a Caché**
  - Una vez iniciado el acceso, cuanto se tarda en disponer del dato
  - Analogía: cuanto tarda el padre en darle la paga (ej: 1 minuto)
- **Tiempo de Ciclo >> Tiempo de Acceso**

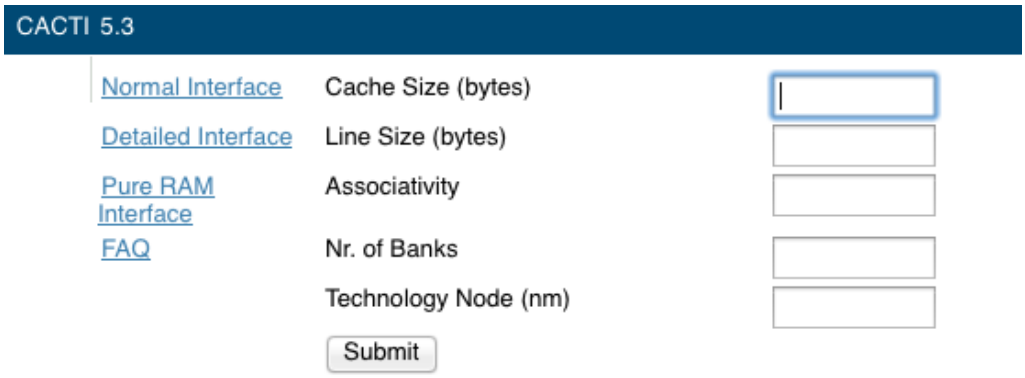
Módelo de Caché en CACTI



Versión Web Interactiva

- **Limitada pero suficiente para la mayoría de usuarios**

  http://quid.hpl.hp.com:9081/cacti/



Versión Código C++

- **Obtener última versión**
  - cacti65.tgz
- **Descomprimir**
  - tar xvf cacti65.tgz
- **Ir al directorio de CACTI y compilar**
  - make
- **Ejecutar**
  - cacti parámetros
- **Opciones para especificar parámetros:**
  - (1): por linea de comandos parámetro a parámetro
    - `cacti parámetro1 parámetro2 ... parámetroN`
  - (2): usar un fichero con todos los parámetros y especificar así:
    - `cacti –infile cache.cfg`

Fichero cache.cfg

- **# Cache size**
  - -size (bytes) 134217728
- **# Line size**
  - -block size (bytes) 64
- **# Associativity (to model Fully Associative set to zero)**
  - -associativity 1
- **# Read/Write Ports**
  - -read-write port 1
  - -exclusive read port 0
  - -exclusive write port 0
  - -single ended read ports 0
- **# Multiple banks connected using a bus**
  - -UCA bank count 1
- **# Technology**
  - -technology (u) 0.032

Resultados CACTI

- **Divide los resultados en:**
  - Cache Parameters
  - Time Components (data array and tag array)
  - Power Components (data array and tag array)
  - Area Components (data array and tag array)
- **Time components**
  - Access time (ns): 5.25228
  - Cycle time (ns): 10.866
- **Power Components**
  - Total read dynamic energy per read port(W): 3.22116900049
  - Total standby leakage power per bank (W): 26.2951606931
- **Area Components:**
  - Total area (mm^2): 301.269599031