UNIVERSITA' DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

CACHES D-NUCA TRIANGOLARI: MODELLAZIONE E VALUTAZIONE DEL CONSUMO DI POTENZA DINAMICO

Relatori:

Ch.mo Prof. Cosimo Antonio Prete _____

Ch.mo Prof. Marco Avvenuti _____

Ing. Pierfrancesco Foglia _____

Candidato:

Taglione Emiliano _____

Anno Accademico 2003/2004

ABSTRACT

I processori hanno frequenze di funzionamento sempre crescenti, ed integrano caches di dimensioni sempre maggiori; ad esempio l'Intel Xeon 3 GHz (core Gallatin-4M) ha 4Mb di cache di terzo livello, Intel Pentium-M (core Dothan) ha 2Mb di secondo livello, Intel Itanium (core Madison 9M) ha 9Mb di terzo livello.
Dato che la cache L2 occupa circa la metà dell'area di silicio del chip, questo componente incide molto sul costo del processore, sulle sue prestazioni, e sul consumo di potenza.

In questo lavoro di tesi sarà valutato il consumo di potenza dinamico per le architetture di cache ad accesso non uniforme (Dinamic NUCA).
A tal fine è stata modellata un' architettura per le cache D-NUCA e Triangular D-NUCA; ricavati dei parametri per la stima del consumo dinamico; effettuate simulazioni per determinare il consumo dinamico della cache.
I parametri di consumo sono stati ottenuti con il software per la modellazione di memorie cache CACTI.
Le simulazioni sono state effettuate modificando il simulatore Sim-Alpha (simulatore del processore ALPHA 21264) adattato all'utilizzo di cache TD-NUCA , i benchmark scelti: *176.gcc*, *181.mcf*, *256.bzip2*, *300.twolf*, sono appartenenti alla suite SpecInt 2000.
I risultati ottenuti portano alla conclusione che le cache TD-NUCA Decrescenti sono risultate in assoluto le migliori sia in termini di consumo che di prestazioni su tre dei quattro benchmark; ne consegue che ulteriori affinamenti in questa tecnica progettuale sono auspicabili.

SOMMARIO

Capitolo 1

INTRODUZIONE

Frequenze di clock sempre più elevate, core sempre più piccoli e memorie caches sempre più grandi, questo è l'attuale scenario nel mondo dei microprocessori.

L'aumento delle frequenze di funzionamento dei processori comporta ritardi di propagazione dei segnali sempre più incidenti sulle latenze di ogni componente del chip. Con la tecnologia corrente si riescono a minimizzare i tempi di risposta, ma il ritardo introdotto dai canali di collegamento non può essere diminuito.

Attualmente si realizzano cache per cui si ipotizzano tempi di latenza costanti senza considerare che, in una cache a più vie, quelle più lontane dal controller avranno in futuro dei ritardi di propagazione sempre più elevati (causa principale dei conflitti di canale) che pregiudicheranno il rispetto delle specifiche dell'intera cache.

Alla luce di queste considerazioni un modello di cache che consideri tempi di accesso uniformi risulterà inefficiente.

Per questo motivo è necessario ricercare modellazioni alternative.

L'idea di base sta nel creare caches con tempi di accesso non-uniformi, funzione del particolare banco cui si accede, in modo che ogni via abbia un ritardo di accesso indipendente dalle vie precedenti.

In tale ottica è possibile ritardare i tempi di risposta delle vie finali della cache in modo da consentire un minor consumo energetico; oppure costruire ogni via con una diversa tecnologia, così da creare vie più veloci che consumano di più, e vie più lente che consumano di meno.

Uno studio dell'Università del Texas ([2] e [6]) ha proposto la creazione di un dinamismo fra le vie della cache in modo che i blocchi di dati in esse contenuti vengano gestiti con la politica LRU.
Questo studio è stato ripreso per estendere il concetto di Dynamic-NUCA a quello di Triangular Dynamic NUCA [10].
Compito di questo lavoro sarà elaborare uno studio il cui scopo è dare una stima del consumo di potenza dinamico di queste nuove tecnologie di caches.
Pertanto è stato sviluppato un modello di architettura fisica per le cache D-NUCA ed uno TD-NUCA che potesse rispettare le specifiche di progettazione logica.
Per determinare i parametri di consumo derivanti da tale modello è stato utilizzato il software per la modellazione di memorie caches CACTI [12].
Questo software permette la modellazione di consumo di potenza, tempi di accesso, ed area, per cache standard.
Per adattare i dati forniti dal simulatore al modello di cache NUCA, è stato necessario sviluppare un modello matematico che le rappresentasse.
Considerando che queste cache vengono accedute sequenzialmente dalla via più vicina al controller alla via più lontana, è stato possibile individuare un algoritmo per la determinazione del consumo in caso di hit e di miss.
Le simulazioni sono state effettuate con il software SimAlpha [1] [7] (simulatore del processore Alpha 21264), questo simulatore è stato modificato per implementare il calcolo del consumo di potenza ; i benchmark utilizzati sono quelli appartenenti alla suite SpecInt 2000.
I risultati ottenuti sono discussi nel capitolo 5 Conclusioni.

Capitolo 2

LE CACHES D-NUCA E LE TD-NUCA

Nella sezione seguente verranno esposte le metodologie di progettazione e gestione delle cache NUCA acronimo che sta ad indicare Non Uniform Cache Arcitecture; e nella fattispecie verranno prese in esame le D-NUCA (Dinamic) e TD-NUCA (Triangular Dinamic)

2.1 D-NUCA CACHES

La famiglia di caches NUCA è composta da varie tipologie, in funzione dei metodi di accesso e mappaggio che si adottano, in sostanza hanno un'organizzazione in più livelli con latenze di accesso discrete, i banchi più vicini al controller della cache avranno una latenza di accesso minore, e quelli più lontani avranno la latenza massima. Ovviamente per permettere questo c'è bisogno di una rete di interconnessione che consenta ad ogni banco un accesso indipendente.
Le D-NUCA sono fra le varie architetture NUCA quelle che meglio interpretano il concetto di latenza variabile, esse infatti permettono una mobilità dei dati fra le varie vie che le compongono, causando migrazioni di blocchi più o meno recentemente usati che determinano un incremento delle prestazioni di queste ultime ([2] [6] [18]).

I vantaggi maggiori da questa architettura si possono ottenere posizionando i dati più frequentemente acceduti nelle posizioni più veloci ed allontanando da queste i meno usati. In questa sezione mostreremo la migrazione dei dati dai banchi più lontani al controller a quelli più vicini in modo dinamico e secondo politiche LRU. Di seguito saranno analizzate alcune politiche di gestione di questi tipi di caches con lo scopo di ridurre i tempi medi di accesso della L2 e di migliorare le prestazioni complessive.

Per stabilire le politiche di migrazione si deve rispondere a tre domande: come i dati sono mappati nei banchi, come è possibile ritrovare i dati, sotto quali condizioni i dati possono migrare.

2.1.1 Mappaggio dei blocchi

Per quanto riguarda questo aspetto, i due estremi sono: mappare un blocco in modo statico in un solo banco, oppure mappare il blocco in modo dinamico in qualsiasi banco. Un compromesso tra flessibilità di piazzamento dei dati e velocità nel ritrovarli fra i banchi, è la soluzione a spread sets, in cui la matrice di banchi che costituisce la cache è considerata come un struttura set-associative ed ogni banco corrisponde ad una via di un particolare insieme. La collezione dei banchi che compongono un insieme è detta bank set e la sua cardinalità definisce l'associatività della cache. Ad esempio una cache associativa a 4 insiemi e composta da 32 banchi, ha 8 bank sets di 4 banchi l'uno.
Tre possibili politiche di mappaggio sono le seguenti:

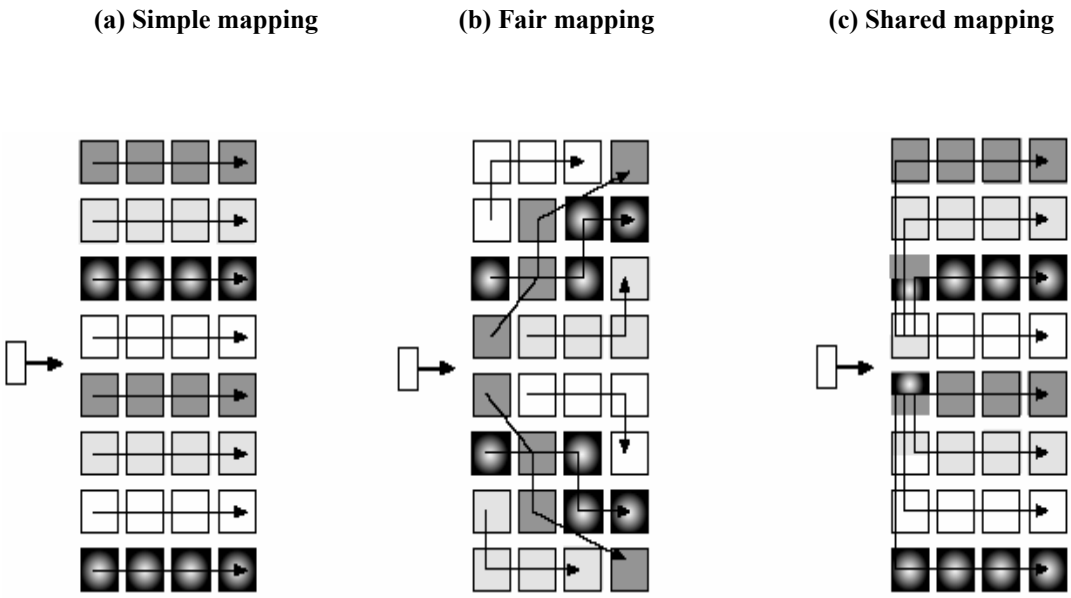

Figura 1: Mappaggio dei bank sets nei banchi.

Un bank set è la collezione di banchi che costituisce un insieme. La sua cardinalità definisce l'associatività. I bank sets sono distribuiti sui banchi fisici nel modo indicato

dalle frecce. Con il simple mapping i bank sets laterali saranno svantaggiati rispetto a quelli centrali, poiché più lontani dal controller. Con il fair mapping la disparità rimane considerevole a livello della prima via (primo banco) di ogni bank set. Nello shared mapping, i primi banchi (più veloci) sono condivisi da ogni bank set.

Nel simple mapping, mostrato in figura 3a, ogni riga di banchi assume il ruolo di bank set ed ogni banco che la compone appartiene ad una diversa via dello stesso insieme. La ricerca di un dato avviene quindi selezionando il particolare bank set, selezionando l'insieme giusto all'interno di questo ed infine comparando i tag via dopo via. Sono presenti però degli inconvenienti: primo, il numero di insiemi desiderati nella cache può non corrispondere con il numero delle righe che si vengono a creare con i bank set; secondo, i tempi di accesso dei bank sets non sono uniformi. Infatti, ipotizzando la porta di ingresso alla cache nel centro delle righe, i bank sets centrali avranno una latenza minore di quelli laterali.

La seconda politica, la fair mapping (figura 3b), aggiunge un po' di complessità per risolvere i problemi sopra citati. Il mappaggio dei bank sets sui banchi fisici segue adesso le frecce mostrate in figura. Con questo modello il mappaggio è fatto in modo che i tempi medi di accesso degli interi bank sets risultino simili. Proprio perché i bank sets non corrispondono più alle righe di banchi fisici è possibile variare il numero di banchi (e quindi di vie) che compongono i bank sets stessi. Anche con questa soluzione, i banchi più vicini al controller di ogni bank set (che saranno i più acceduti), non sono equidistanti al controller stesso e ciò può creare una disparità fra i vari insiemi (almeno a livello della prima via). L'ultima soluzione, la shared mapping di figura 3c, permette a tutti i bank sets di condividere i banchi più vicini al controller in modo da fornire la stessa latenza (quella minore possibile) ai banchi più usati di ogni insieme. Tutto ciò ovviamente comporta un certo overhead di complessità nella gestione dei banchi stessi.

2.1.2 Ricerca dei dati

Una volta selezionato il corretto bank set a partire dall'indirizzo, e selezionato il corretto insieme all'interno di questo, la ricerca del blocco fra le vie della cache prosegue secondo due possibili modalità: ricerca incrementale o multicast. La soluzione a ricerca incrementale consiste nella comparazione dei tag partendo dalla via più vicina al controller e proseguendo allontanandosi da questo solo in caso di miss della via considerata. È ovvio che la ricerca in ogni via deve prima attendere la terminazione della ricerca nella via precedente.

Questa politica, dato che genera solo lo stretto numero necessario di messaggi di ricerca (o pacchetti di accesso) all'interno della rete di interconnessione, permette di tenere basso il consumo di energia, ma se il blocco dovesse trovarsi nelle ultime vie oppure non fosse presente in cache, si avrebbe una notevole riduzione delle prestazioni.

Con la politica a ricerca multicast, i tag di tutte le vie vengono comparati in parallelo permettendo un notevole risparmio di tempo in caso di miss. Per attuare questa soluzione però, dal controller devono essere diretti tanti messaggi di ricerca quanta è la cardinalità del bank set, ma solo uno di questi tornerà indietro con il risultato. Questa tecnica quindi consuma una maggiore quantità di energia rispetto al caso precedente, ed inoltre i troppi messaggi scambiati nella rete possono limitare leggermente le prestazioni a causa della maggiore possibilità di conflitti, sia di canale che di banco. Per questo sono state studiate ulteriori tecniche ibride:

la limited multicast, secondo la quale solo le prime M vie delle N presenti sono utilizzate in parallelo per poi procedere in modo sequenziale;

la partitioned multicast, nella quale le vie sono partizionate in sottoinsiemi che sono acceduti in parallelo fra loro con una ricerca incrementale al loro interno;

la limited search, nella quale si parte con una ricerca incrementale fra le prime vie per poi finire in modo parallelo consentendo di limitare la latenza nel worst-case.

È evidente che queste politiche sono un compromesso fra il tempo di accesso alla cache ed il consumo di energia.

2.1.3 Migrazione dei blocchi

L'obbiettivo della migrazione è quello di massimizzare le hit nei banchi più vicini e veloci avvicinando i blocchi maggiormente acceduti. È naturale quindi usare una politica di scambio dei blocchi basata sull'LRU. Una politica di questo tipo, che promuove ad ogni accesso i blocchi in posizioni più vicine, genera però molti movimenti di dati nella rete di interconnessione, aumentando il consumo di energia e la possibilità di conflitti. Per trovare un compromesso sono state studiate varie soluzioni.

In generale il problema della migrazione è risolto grazie alla definizione di tre diversi meccanismi che rispondono ai problemi di inserzione dei dati in cache, di espulsione e di promozione degli stessi. Per quanto riguarda l'inserzione di un blocco in seguito al verificarsi di una miss sullo stesso, è possibile scegliere l'inserzione in coda alla lista delle vie (in pratica l'inserzione avviene nella via più lontana), in modo da evitare di allontanare blocchi più importanti e frequentemente acceduti. Così facendo però, si verrà a generare un maggior numero di scambi nel caso lo stesso blocco sia nuovamente acceduto in seguito.

In alternativa è possibile inserire il nuovo blocco in testa o in una posizione casuale all'interno del bank set allontanando il blocco vittima dell'inserzione. È possibile inoltre limitare il numero delle copie dei blocchi da allontanare applicando una politica zero-copy, rimpiazzando cioè direttamente la vittima scelta per l'inserimento (potrebbe però essere eliminato un blocco importante nel caso di inserzione in testa o nel mezzo), o one-copy, copiando cioè la vittima dell'inserzione in una posizione più lontana e rimpiazzare dalla cache quest'ultimo blocco. Va da se che con una tecnica one-copy, il blocco rimpiazzato sarà poco importante, ma sarà

generato uno scambio in più all'interno della cache con conseguente aumento del numero degli accessi ai banchi e quindi del consumo di energia e del numero dei possibili conflitti.

Per quanto riguarda l'espulsione dei blocchi è ritenuta ragionevole l'ipotesi di buttar fuori il blocco in ultima posizione (il meno recentemente usato) in quanto è quello meno importante fra quelli presenti in cache oppure quella di espellere un blocco in posizione casuale.

Le tecniche di rimpiazzamento da usare potranno essere allora l'inserzione in coda con una zero-copy ed una estrazione finale dalla coda, oppure una inserzione nel mezzo (o casuale) con una conseguente one-copy in coda e l'estrazione da questa, o infine una inserzione in testa, una successiva one-copy ed una estrazione casuale.

Infine, il protocollo di promozione è sostanzialmente uno solo al quale si può scegliere di applicare o meno qualche variante. La promozione viene detta generazionale perché propone di scambiare ogni blocco acceduto in cache con quello immediatamente più vicino al controller in seguito al verificarsi di una hit sul blocco stesso.

Questa tecnica segue l'algoritmo LRU, infatti i blocchi più acceduti migreranno velocemente nelle prime posizioni a scapito di quelli meno importanti che si allontaneranno da queste. Per limitare il numero di scambi effettuati, ci sono varie possibilità determinate dalla variazione del numero delle hit necessarie ad un blocco per effettuare una migrazione in avanti e dalla variazione della distanza (in numero di banchi) che questo può coprire ad ogni promozione.

2.2 TD-NUCA CACHES

Sulla base degli studi compiuti per le D-NUCA nel lavoro [10] sono state modellate le TD-NUCA; obiettivo conseguito dallo studio di queste ultime è stato di ottenere a parità di area di silicio occupata, prestazioni più elevate, oppure prestazioni equivalenti con un'area di silicio però fortemente ridotta. E' importante notare che risparmiare sulla superficie del chip porta ad una minore complessità dei circuiti, quindi sia ad un minor consumo energetico che ad una migliore dissipazione del calore.

Come mostrato in figura, per una cache rettangolare con n vie in cui la grandezza di pagina vale per ipotesi 2^{n-1}, si ha un numero totale di banchi dato da $n*2^{n-1}$.

Nella equivalente cache triangolare il numero di banchi utilizzati, è dato dalla formula $\sum(i=0, n-1)(2^i)$ corrispondente al valore 2^n-1. Il rapporto di questi due valori al tendere di $n\rightarrow\infty$ tende a n, vale a dire che per un alto numero di vie il risparmio del numero di banchi delle caches triangolari è lineare nel numero di vie.

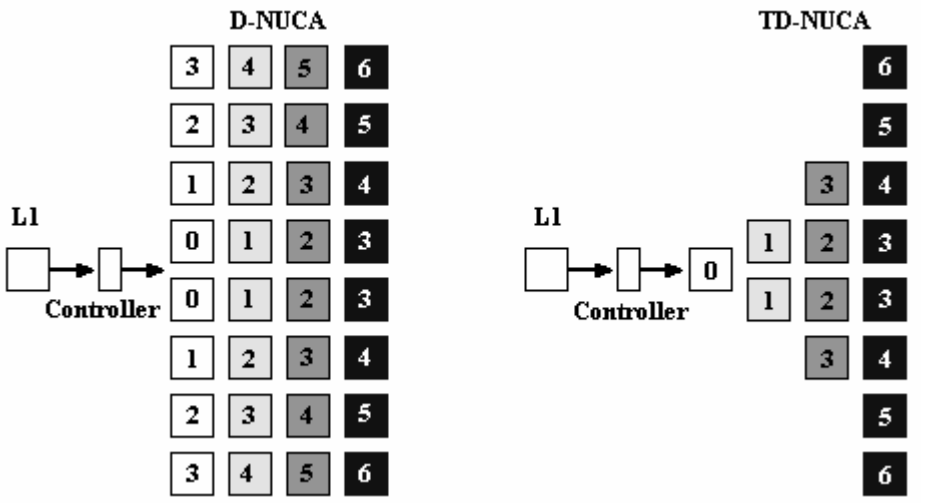

Figura 2: Struttura dei banchi di una cache D-NUCA Triangolare a confronto con una D-NUCA rettangolare.

La TD-NUCA ha solo 15 banchi contro i 32 della normale D-NUCA con un risparmio di circa il 50% sull'area del chip.

Ogni banco ha riportata la latenza che impiega un pacchetto di accesso per arrivarvi partendo dal controller (tempo di routing).

Quello che cambia fra le due architetture è il posizionarsi dei singoli blocchi nei banchi. Supponiamo cioè di prendere un blocco il cui indirizzo è tale per cui debba essere posizionato nel primo bank set della cache rettangolare (il seguente esempio è mostrato in figura 3): i tempi di routing possibili per la ricerca di questo blocco sono 3 (se il blocco dovesse trovarsi nella prima via), 4 (se il blocco dovesse trovarsi nella seconda via), 5 (nella terza via) e 6 (nella quarta), con una media, in caso di hit per la ricerca, di 4.5 cicli di clock.

Considerando lo stesso blocco durante una ricerca nella cache triangolare: i tempi di routing possibili sono adesso di 0 (nel caso si trovasse nella prima via), 1 (nel caso si trovasse nella seconda via), 3 (se si trovasse nella terza via) e 6 (se si trovasse nella quarta via), con una media di 2.5 cicli di clock.

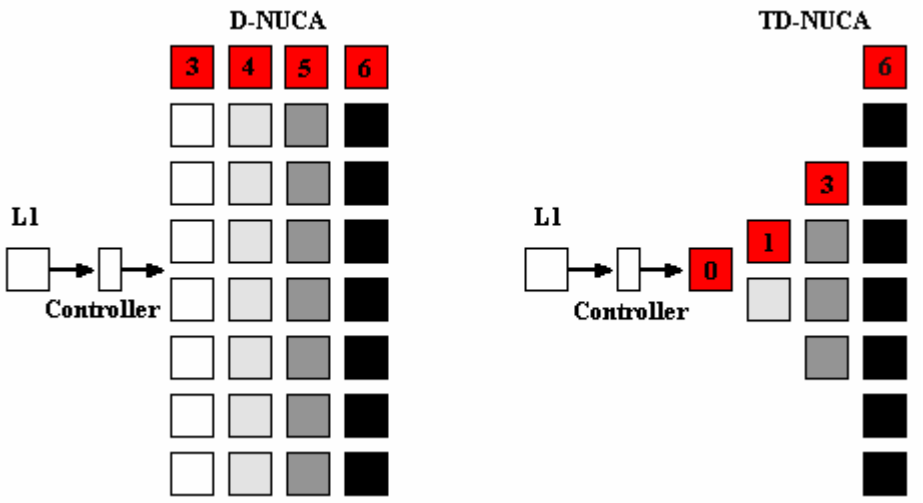

Figura 3: Posizionamento di uno stesso blocco nelle 4 vie di una cache D-NUCA 8x4, e nelle 4 vie di una cache TD-NUCA [1,2,4,8].

Si può notare che in una cache triangolare il blocco risulta mediamente più vicino al controller fornendo tempi di accesso tipici di una cache di più piccole dimensioni.

Osserviamo ora la suddivisione dell'indirizzo. Essendo il numero di banchi in ogni via non costante, è ovvio che la parte dell'indirizzo che indica l'indice del banco in cui posizionare un particolare blocco deve per forza di cose essere a grandezza variabile. La figura 4 mostra per una cache da 4 vie ognuna delle quali composta da 1, 2, 4 e 8 banchi rispettivamente, la suddivisione dell'indirizzo e la posizione del campo utilizzato per fare la scelta sul posizionamento del banco (campo bank).

Considerando uno spazio di indirizzamento a 32bit, dimensione dei blocchi di 64 Byte e banchi contenenti 4096 insiemi; la prima via, composta da 1 banco comprendente 4096 insiemi, avrà la parte indice dell'indirizzo usato per accedervi di 12bit e non sarà presente il campo bank; la seconda via è composta da 2 banchi e comprende 2*4096 insiemi, la grandezza dell'indice dovrà allora essere di 13bit che considereremo suddivisi nei 12bit meno significativi costituenti il campo index e nel bit più significativo che permette la scelta del banco in cui cercare il blocco.

Continuando a percorrere le vie, ad ogni passo il bit meno significativo del campo tag passerà a costituire il campo bank dell'indirizzo che aumenterà di una unità alla volta.

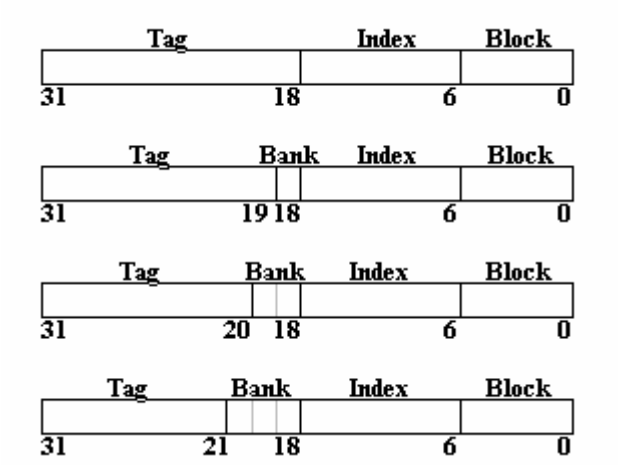

Figura 4: Suddivisione dell'indirizzo di un byte per una cache TD-NUCA.

È considerato uno spazio di indirizzamento a 32bit, la dimensione dei blocchi vale 64Byte, i banchi contengono 4096 insiemi e sono presenti 4 vie rispettivamente composte da 1, 2, 4 e 8 banchi dalla prima all'ultima. Il campo tag assorbe i rimanenti bit più significativi dell'indirizzo.

L'accesso alla prima via non necessita della parte bank dell'indirizzo in quanto è presente un solo banco (prima suddivisione). L'accesso alla seconda via deve considerare 1bit per la scelta del banco in cui è posizionato il blocco, sottraendolo alla parte meno significativa della parte tag (seconda suddivisione). L'accesso alla terza via utilizza 2bit del campo bank diminuendo ulteriormente la grandezza del tag (terza suddivisione). L'accesso all'ultima via necessita di 3bit per la scelta del banco e lascia solo 11bit al campo tag (quarta suddivisione).

2.2.1 Mappaggio dei blocchi

Data questa nuova tipologia di indirizzamento, ogni blocco può essere mappato nei banchi di ogni via. La funzione che definisce il mapping è adesso più complicata perché al raddoppiare della dimensione della pagina di memoria da una via alla successiva, si deve considerare la doppia possibilità che ha un blocco di posizionarsi nella via successiva. Si verrà così a creare un albero binario di possibili posizioni che può assumere un blocco con un particolare indirizzo dalla prima via all'ultima. Definire le posizioni di un blocco in ogni via e quindi definire la forma dell'albero significa perciò definire la funzione di mapping. Tutti i bank set avranno quindi alcuni banchi (i primi) in comune fra loro.

E' importante osservare inoltre che la funzione che calcola il mapping non deve essere complicata ma deve essere eseguita dal controller in un singolo ciclo di clock in modo da non introdurre ulteriori latenze nella ricerca di un blocco.

I due possibili mapping in analogia con gli equivalenti del caso D-NUCA.

a) Simple mapping **b) Fair mapping**

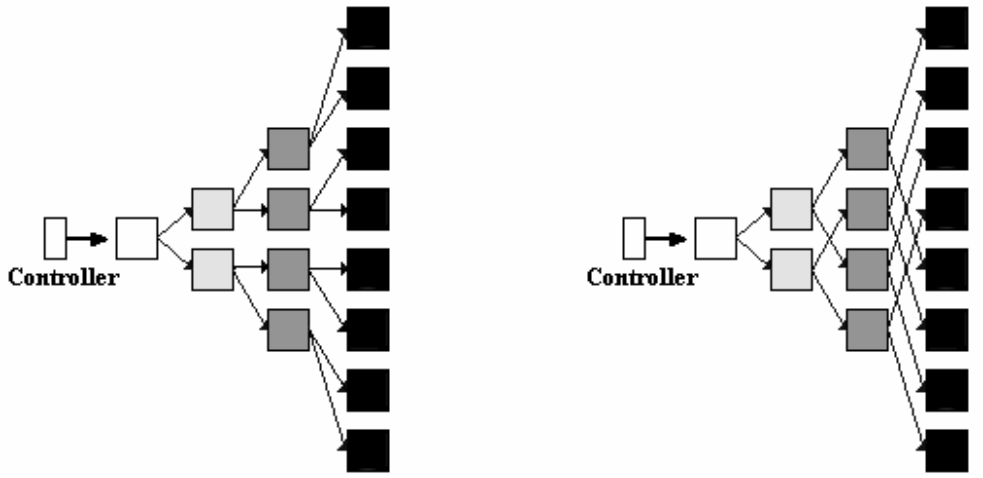

Figura 5: Possibili mapping dei blocchi nei banchi.

Nel simple mapping la funzione di mapping produce l'albero binario mostrato a sinistra. Si nota che i bank set laterali hanno tempi medi di accesso molto differenti da quelli più centrali.

Nel fair mapping, con l'albero binario di destra, si ha la minima varianza fra i tempi medi di accesso di ogni bank set.

Nel simple mapping, mostrato in figura 5a, un blocco in posizione *i* di una via si può mappare nella via successiva (se ha dimensione doppia) nelle posizioni *i*2* e *i*2+k* (dove *k* è la dimensione del banco). Questo mapping però, penalizza quei blocchi i cui indirizzi li portano ad essere caricati nei banchi laterali che avranno una latenza media di accesso più alta.

I bank set laterali vengono quindi penalizzati maggiormente rispetto a quelli centrali.

E' possibile realizzare questo mapping aggiungendo ad ogni via un bit nella parte bank dell'indirizzo (con riferimento alla figura 6a) e considerando quest'ultimo bit come il meno significativo del campo bank. Il campo bank dovrà quindi essere letto da destra verso sinistra.

Nel fair mapping, mostrato in figura 5b, la funzione che definisce la posizione dei blocchi nei banchi impone che un blocco in posizione *i* di una certa via si debba mappare nella via successiva (se ha dimensione doppia) nelle posizioni *i* e *i+k*n* (dove *n* è il numero di banchi della via precedente). Con questa tecnica si può affermare che lo scostamento dalla media dei tempi medi di accesso di ogni bank set, è il minimo possibile.

Questa funzione di mapping si realizza aggiungendo alla parte bank dell'indirizzo un bit per ogni via e considerando quest'ultimo come il meno significativo (figura 6b). Il campo bank si verrà quindi a leggere normalmente da sinistra verso destra.

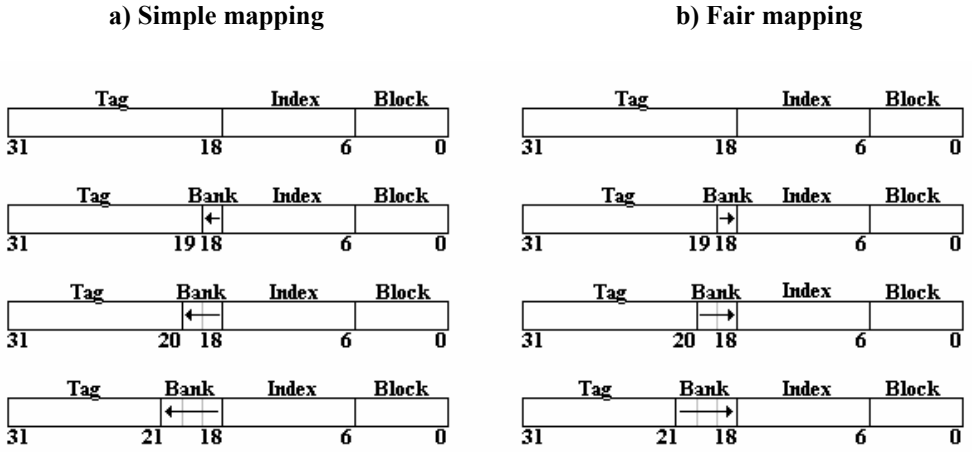

Figura 6: suddivisione dell'indirizzo e verso di lettura del campo bank nei due tipi di mapping.

Con il simple mapping il campo bank ha i bit più significativi a destra mentre quelli più a sinistra, sottratti al campo tag, sono considerati i meno significativi. Per implementare invece il fair mapping il campo bank deve essere letto normalmente da sinistra verso destra.

Come abbiamo visto il fair mapping è più equo del simple mapping in considerazione del fatto che ogni bank set ha un tempo di routing più vicino alla media totale.

Lo svantaggio principale del fair mapping rispetto al simple mapping è dovuto al diverso routing che si rende necessario per la distribuzione dei pacchetti di accesso ai banchi obiettivo della ricerca di un blocco.

Il fair mapping genererà dunque un maggior numero di pacchetti di accesso in cache aumentando la probabilità di conflitti di canale (i conflitti di banco rimangono gli stessi in quanto il numero di banchi acceduti nei due diversi mapping rimane costante).

2.2.2 Routing

Il routing definisce il percorso intrapreso da ogni pacchetto di accesso per arrivare ai banchi obiettivo di ogni operazione di cache (lettura, scrittura, promozione…).

In una cache D-NUCA i pacchetti di accesso seguono i canali mostrati in grigio in figura 7a spostandosi prima verticalmente fino ad arrivare alla giusta riga del banco (e del bank set) e poi orizzontalmente fino a raggiungere la colonna corretta. Durante la ricerca di un blocco in un bank set ad esempio, ci si sposterà sulla giusta riga e poi, a distanza di 1 ciclo di clock l'uno dall'altro, si accederà ad ogni banco, dalla prima via all'ultima sfruttando i minimi tempi possibili per arrivare a destinazione.

I canali che si rendono necessari a questo routing (detto routing 1) sono quelli mostrati in figura 7.

Volendo estendere questo tipo di routing alle caches triangolari, devono essere aggiunti dei canali verticali anche alle vie successive alla prima in quanto questo è l'unico modo per raggiungere i banchi più esterni delle vie più grandi. Il routing 1 per le caches triangolari quindi impone che di via in via si debba raggiungere prima la riga mèta della ricerca, e poi la colonna (figura 7b), in modo analogo a quanto avviene per le caches rettangolari. Così facendo, i tempi di routing di ogni banco sono gli stessi delle cache D-NUCA, cioè sono i minimi possibili.

Con questo routing un pacchetto di accesso per la ricerca di un blocco passa per ogni banco del bank set, vi accede, e poi prosegue nella via successiva. Ciò è possibile perché abbiamo implicitamente supposto di utilizzare il simple mapping in modo da avere banchi consecutivi del bank set fisicamente vicini tra loro. Si può ad esempio notare come i due banchi più in alto della quarta via in figura 7b siano acceduti dal banco a loro più vicino della via precedente cioè quello più in alto della terza via.

a) Routing 1 in D-NUCA **b) Routing 1 in TD-NUCA** **c) Routing 2 in TD-NUCA**

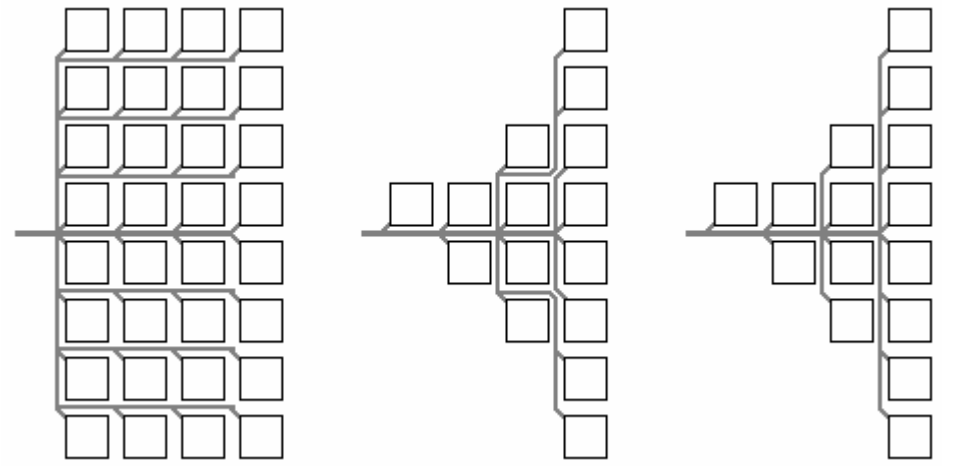

Figura 7: Diverse tipologie di routing nelle due architetture di caches.

Il routing usato nelle caches D-NUCA (routing 1) impone ad ogni pacchetto di spostarsi sulla giusta riga in corrispondenza della prima colonna. In seguito il pacchetto potrà viaggiare solo in orizzontale senza spostarsi da una riga all'altra. In grigio scuro sono riportati i canali necessari per implementare questo routing.

È possibile creare un equivalente del routing 1 da usare nelle caches TD-NUCA (solo con il simple mapping). Come si vede, ogni pacchetto si sposta prima di riga in riga (verticalmente) e solo dopo prosegue nelle vie successive. A differenza del caso precedente però, si rendono comunque necessari gli spostamenti verticali anche nelle vie successive alla prima.

Il routing 2, usato solo nel fair mapping, fa proseguire i pacchetti prima orizzontalmente di colonna in colonna e poi verticalmente fino a giungere alla riga corretta. Questa differenza può cambiare il numero di conflitti generati.

Se invece usassimo il routing 1 con il fair mapping, essendo presenti dei percorsi più lunghi da ogni via alla successiva (le diramazioni dell'albero sono più "lunghe"), non sarebbe possibile ottenere i tempi di routing migliori.

Il routing 2, che sfrutta i canali mostrati in figura 7c, permette di raggiungere i banchi spostandosi prima sulla giusta colonna e successivamente in modo verticale fino a raggiungere la giusta riga. Così,

per raggiungere i banchi appartenenti ad un bank set, sarà lanciato un pacchetto che procederà in orizzontale, e da ogni colonna si diramerà procedendo sia nella successiva colonna, sia in verticale all'interno della via. Ad ogni istante quindi lo stesso accesso al bank set ha in cache più pacchetti, ognuno dei quali procede verticalmente in una diversa via.

In tal modo si può ottenere, anche per questa tipologia di mapping, gli stessi tempi delle caches rettangolari. Il prezzo da pagare è il maggior numero di pacchetti in circolazione nella cache, proprio perché in ogni colonna dal canale centrale si genera un nuovo pacchetto per proseguire il cammino verticalmente.

Il maggior numero di pacchetti si tradurrà in un maggior numero di conflitti sui canali.

In definitiva le migliori soluzioni adottate per le caches TD-NUCA sono da scegliere fra il simple mapping abbinato all'uso del primo tipo di routing, e il fair mapping abbinato al routing numero 2.

a) Routing 1 con simple mapping **b) Routing 2 con fair mapping**

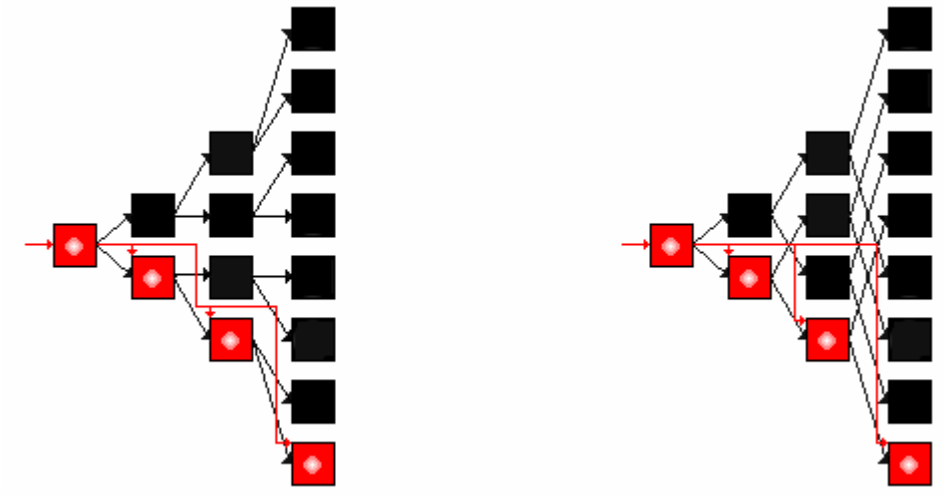

Figura 8: esempio di accesso ad un bank set (in rosso) nei due diversi routing.

I tempi di accesso ai singoli banchi sono gli stessi nei due routing, quello che cambia è il percorso intrapreso dai pacchetti di accesso. Nel routing 2 la lunghezza del percorso preso dai pacchetti (la lunghezza totale della riga rossa) è maggiore rispetto a quella

del caso 1 (7 lati di banco contro 6). Come conseguenza, le probabilità di conflitti di canale nei due routing possono cambiare, e più precisamente ci aspetteremo più conflitti nel secondo tipo di routing.

In figura 8 è mostrato l'esempio di accesso ad un bank set (costituito dai banchi in rosso) con le due modalità di routing e i rispettivi mapping. La riga in rosso mostra il percorso preso dal pacchetto (o dai pacchetti nel caso di routing 2) nel raggiungere tutti i banchi.

Dalla comparazione delle due figure si osserva che l'accesso all'intero bank set fa percorrere al pacchetto di accesso 6 canali (si intende 1 canale come il lato di 1 banco) nel caso del routing 1, mentre nel caso del routing 2 ne devono percorrere 7. Questo maggiore impiego di risorse si traduce in un maggior consumo energetico ed in una maggiore probabilità di conflitto fra i pacchetti di accesso. D'altra parte il routing 2 è l'unico modo che si ha con il fair mapping per ottenere i migliori tempi di accesso ai singoli banchi della cache.

2.2.3 Politiche di rimpiazzamento e promozione

Il rimpiazzamento si snoda nella scelta delle posizioni in cui inserire il nuovo blocco e da cui estrarre il blocco da rimpiazzare.
Per quanto riguarda l'inserzione sono state prese in considerazione due politiche: inserire nell'ultima via della cache, oppure in una scelta a caso.
Più precisamente l'inserzione avviene in un blocco vuoto, se esiste, oppure seguendo una delle due politiche appena menzionate.
Dopo l'inserzione deve essere scelta la posizione dalla quale sarà estratto il blocco da rimpiazzare. L'unica politica scelta è quella di estrazione dal fondo. Ovviamente, se la posizione in cui inserire non coincide con quella da cui estrarre si rende necessaria una operazione di one-copy con la quale viene portato il blocco occupante la posizione di inserzione, nel blocco con la posizione di estrazione (ma solo dopo aver fatto il writeback di quest'ultimo blocco).
Per ottimizzare l'estrazione, il dato viene posizionato in un blocco vuoto, se esiste (in modo da evitare il writeback), altrimenti nell'ultima via.
È da tenere in considerazione che il bank set in cui scegliere la posizione da inserire è quello relativo all'indirizzo del blocco che deve essere inserito in cache, mentre il bank set in cui scegliere la posizione di estrazione è quello riferito all'indirizzo del blocco in cui avviene l'inserzione.
Per quanto riguarda la promozione, questa avviene scambiando il blocco che ha subito la hit con il blocco vittima, scelto risalendo le frecce nell'albero del mapping, è inoltre possibile variare in modo flessibile il numero di vie di avanzamento del blocco, da un minimo di 0 (assenza di promozione) ad un massimo corrispondente all'associatività della cache (promozione assoluta).
Rimane da osservare che è possibile il caso in cui un blocco debba essere promosso in una delle vie precedenti, e lo scambio con il blocco che occupa la nuova posizione non sia possibile.

Il blocco vittima della promozione potrebbe cioè avere un indirizzo incompatibile con il banco in cui è caricato il blocco da promuovere.
In tale caso il blocco da promuovere proviene da una particolare diramazione dell'albero di mapping (una delle due frecce possibili nell'andare a ritroso da una via alla precedente), mentre il blocco vittima può seguire solo l'altra diramazione (nello spostamento verso la via successiva).
In tali casi si può: o evitare la promozione, oppure salvare in memoria, invalidare il blocco vittima e successivamente eseguire l'operazione di promozione (lo scambio è reso possibile).

2.2.4 Tecniche di ricerca

Analogamente alle caches rettangolari sono proposte le tecniche di ricerca incrementale e multicast.

La ricerca incrementale procede all'accesso del banco in una via solo se il banco relativo alla via precedente ha risposto con una miss.

La ricerca multicast procede all'accesso contemporaneo dei banchi di tutte le vie senza aspettare da queste nessuna risposta. La ricerca multicast ovviamente produrrà più accessi ai banchi, quindi consumerà una quantità maggiore di energia, ma consentirà di avere dei tempi di risposta minori, soprattutto nei casi di hit alle ultime vie o nei casi di miss. Come compromesso a queste tecniche sono nate soluzioni ibride che consentono ricerche parzialmente multicast e parzialmente incrementali.

Un'osservazione sulle prestazioni di queste tecniche riguarda l'uso della ricerca incrementale con il routing 2.

Con esso infatti si nota come possa essere estremamente inefficiente tale ricerca in quanto un pacchetto dopo aver acceduto al banco di una via, deve tornare nella riga centrale della cache, spostarsi nella via successiva, e risalire di nuovo la colonna per riportarsi sulla giusta riga.

I tempi di arrivo ai banchi vengono così allungati notevolmente.

Con il routing 1 non si verifica questo fenomeno poiché i pacchetti per portarsi nella via successiva non devono tornare nella riga centrale ma possono proseguire in orizzontale.

2.2.5 TD-NUCA Decrescenti

Infine, si può pensare di rovesciare il triangolo formato dai banchi di cache, in modo da porre la via costituita da più banchi vicina al controller, e la via più piccola più lontano.

Una cache con questa forma presenta ovviamente un aumento delle prestazioni rispetto a quella formata dal triangolo crescente in quanto le vie più vicine al controller sono adesso costituite da un maggiore numero di banchi. Le vie più veloci quindi saranno anche le più grandi.

Quello che ci aspetteremo sarà semplicemente una riduzione dei tempi di accesso alla cache senza nessuna sostanziale modifica di ogni altro indice.

La figura 9 mostra come vengono estese le due modalità di mapping viste nel paragrafo 2 alle caches triangolari decrescenti. Si nota che gli alberi sono rovesciati, e che le possibilità di mapping di un blocco nel passare dalla via cui appartiene alla successiva invece di sdoppiarsi, si dimezzano (frecce che si riuniscono in figura 9).

Questo fatto causa meno problemi nella promozione, che è così sempre possibile, poiché la posizione del blocco da promuovere non risulta mai incompatibile con il blocco vittima di questa operazione, dato che per quest'ultimo non esiste la doppia possibilità (doppia freccia) nello spostamento verso la via successiva.

Per il resto continuano a valere immutate le politiche di gestione come il mapping verticale, il routing, le tecniche di rimpiazzamento, e le tecniche di ricerca viste nei paragrafi precedenti.

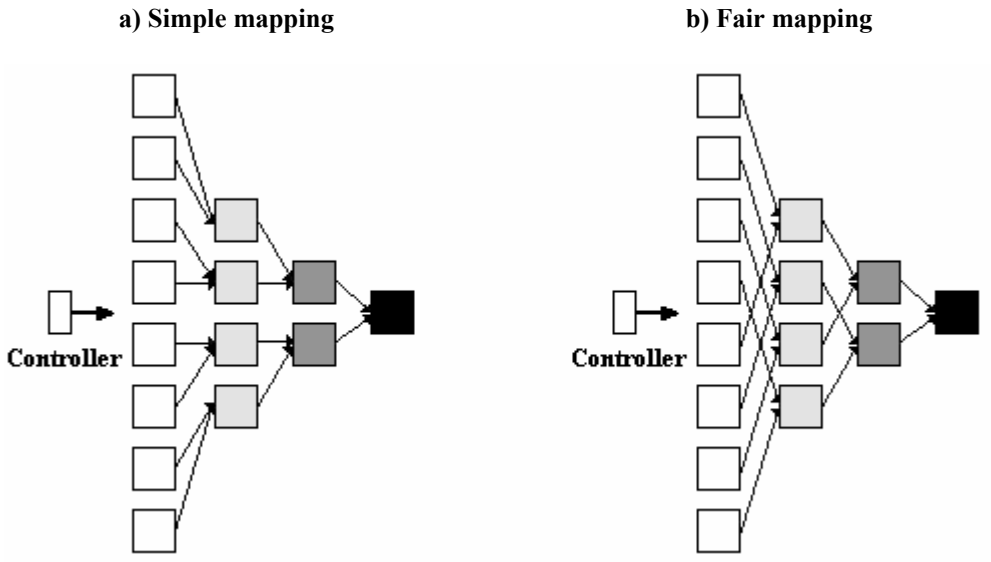

Figura 9: Simple mapping e fair mapping per una TD-NUCA decrescente.

Con le caches TD-NUCA decrescenti l'albero del mapping viene rovesciato. Le frecce adesso, invece di sdoppiarsi, si riuniscono, come conseguenza ci sono meno problemi nella politica di promozione che ora è sempre possibile.

Capitolo 3

IL CONSUMO DI POTENZA

In questa sezione verranno esposte le tecniche per la stima dei consumi nei circuiti integrati, in seguito si spiegherà la metodologia adottata per determinare i parametri di consumo delle caches NUCA; infine si utilizzeranno tali parametri per effettuare le simulazioni necessarie alla stima finale.

3.1 CONSUMO DI POTENZA NEI CIRCUITI INTEGRATI

Il consumo di potenza nei circuiti integrati è determinato da una parte legata al consumo statico ed una legata al consumo dinamico; l'equazione che da la misura di tale quantità è:

$$\mathbf{P = \alpha C V^2 f + I_{off} V}$$

Dove **α** rappresenta il fattore medio di switching dei transistor, **C** la capacità, **V** la tensione di alimentazione, **f** la frequenza di clock e $\mathbf{I_{off}}$ la corrente di leakage.
Il primo termine di questa equazione rappresenta la potenza dinamica e il secondo la potenza statica.
Negli ultimi anni l'effetto dell'aumento della frequenza operazionale ha comportato un innalzamento dei consumi dinamici, mitigata solo in parte dall'effetto della riduzione della sezione dei transistor. Inoltre sebbene la tensione di alimentazione sia diminuita, con essa è diminuita anche la

tensione di soglia, e date le elevate frequenze si è avuto un incremento dell'incertezza nello switching dei transistor per effetto del rumore.
Per quanto concerne la corrente di leakage, essa è legata alla temperatura(**T**) ed alla tensione di soglia($\mathbf{V_t}$) in questa maniera:

$$\mathbf{I_{off} \approx e^{(-Vt/T)}}$$

Da ciò si evince che l'abbassamento della tensione di soglia produce un'innalzamento della corrente di leakage e conseguentemente un'aumento dell'energia statica.
La maggior parte degli studi compiuti per ridurre il consumo di potenza si è comunque indirizzata nella riduzione del consumo dinamico, poiché il consumo derivante dalla corrente di leakage è quantificabile in misura minore.
Alcune tecniche per la riduzione del consumo sono comunque state sviluppate nei lavori [5] [16] e [17].
In riferimento a questi ultimi studi abbiamo tratto spunto per definire un modello di consumo che potesse soddisfare le nostre esigenze, prestando particolare attenzione a quello dinamico.
Lo strumento utilizzato in ambito scientifico per effettuare valutazioni sulla geometria e sui consumi delle caches è CACTI , a differenza dei lavori precedentemente menzionati noi abbiamo concentrato le nostre attenzioni sulla tecnologia di implementazione dei transistor a 50 nanometri che secondo la SIA Roadmap ([3] [11]) dovrebbe giungere in produzione non oltre il 2007.

3.2 IL MODELLO ARCHITETTURALE

Partendo dai modelli precedentemente esposti si sono potute delineare le possibili architetture da adottare.

Per quanto concerne le D-NUCA sono stati presi in esame i tagli di caches da 2,4,8,16 Megabytes e sono stati posti a confronto con i relativi tagli da 1,2,4,8 Megabytes di TD-NUCA nelle due configurazioni, Crescenti e Decrescenti.

Le caches D-NUCA sono state realizzate come caches a 4 vie set associative (per i tagli da 2 e 4 Mb), 8 vie set associative (per il taglio da 8 Mb) e 16 vie set associative (per le 16 Mb).

La stessa scelta è stata fatta per le TD-NUCA (4 vie per 1 e 2 Mb, 8 vie per 4 Mb, e 16 per 8 Mb).

In virtù di tale associatività sono state derivate le dimensioni in Megabytes per le varie vie; nella fattispecie una D-NUCA da 16Mb composta da 16 vie (essendo la geometria planare per tale cache 16x16) avrà la dimensione di ogni via pari ad 1Mb, stesso discorso vale per la D-NUCA da 8Mb (16x8): 8 vie da 1Mb ciascuna, 4 vie da 1 Mb per il taglio da 4Mb (8x4), e 4 vie da 512Kb per le 2Mb(4x4).

Il blocco base per ogni tipologia di cache è stato assunto di 64 Bytes ed il numero di sottobanchi in cui ogni via andava suddivisa è dato dalla geometria di ogni singolo taglio.

Più precisamente sia per i tagli da 16Mb che da 8Mb per ogni via il numero di sottobanchi è 16, mentre per il taglio da 4Mb sono 8 i sottobanchi, ed infine per il taglio da 2Mb sono 4.

Ognuna di queste vie inoltre è stata considerata come fully associative, poiché si deve pensare ad ognuna di esse come se fossero acceduti simultaneamente tutti i banchi ivi presenti (per poi passare alla via successiva in caso di miss), e questo perché tra le varie politiche di gestione delle caches è stata scelta quella che garantisse le prestazioni migliori (per fare ciò è necessario che ognuna delle vie componenti la cache possa permettere il raggiungimento dell'altra).

Discorso un po' più complesso è quello effettuato sulle TD-NUCA; per queste infatti (fermo restando la dimensione base del blocco da 64 bytes e la fully associativity) la geometria è influenzata dal formato triangolare, più precisamente si ha la seguente schematizzazione:

TD-NUCA 8Mb Decrescenti:

- Vie 1-2-3-4 1Mb in 16 sottobanchi
- Vie 5-6-7-8 512Kb in 8 sottobanchi
- Vie 9-10-11-12 256Kb in 4 sottobanchi
- Vie 13-14-15-16 128Kb in 2 sottobanchi

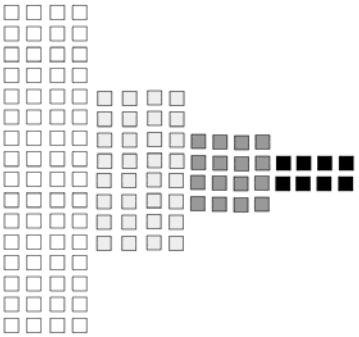

Figura 10: TD-NUCA 8Mb Decrescenti .

Nella cache TD-NUCA decrescente da 8 Mb la suddivisione in banchi delle varie vie e la loro dimensione è omogenea a gruppi di quattro

TD-NUCA 4Mb Decrescenti:

- Vie 1-2 1Mb in 16 sottobanchi
- Vie 3-4 512Kb in 8 sottobanchi
- Vie 5-6 256Kb in 4 sottobanchi
- Vie 7-8 128Kb in 2 sottobanchi

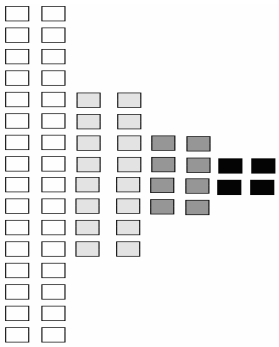

Figura 11: TD-NUCA 4Mb Decrescenti .

Nella cache TD-NUCA decrescente da 4 Mb la suddivisione in banchi delle varie vie e la loro dimensione è omogenea a gruppi di due

TD-NUCA 2Mb Decrescenti:

- Via 1 1Mb in 8 sottobanchi
- Via 2 512Kb in 4 sottobanchi
- Via 3 256 Kb in 2 sottobanchi
- Via 4 128 Kb in 1 sottobanco

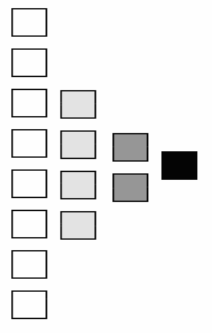

Figura 12: TD-NUCA 2Mb Decrescenti .

Nella cache TD-NUCA decrescente da 2 Mb la suddivisione in banchi delle varie vie e la loro dimensione è diversa per ogni via

TD-NUCA 1Mb Decrescenti:

- Via 1 512Kb in 4 sottobanchi
- Via 2 256Kb in 2 sottobanchi
- Via 3 128Kb in 1 sottobanco
- Via 4 128Kb in 1 sottobanco

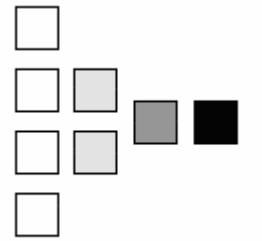

Figura 13: TD-NUCA 1Mb Decrescenti .

Nella cache TD-NUCA decrescente da 1 Mb la suddivisione in banchi delle varie vie e la loro dimensione è differente per le prime due e uguale per le ultime due

Per quanto riguarda le TD-NUCA crescenti si ha la stessa tipologia di architettura, però invertita.

3.2.1 Determinazione dei parametri di consumo

Una volta assunta questa configurazione sono stati valutati i consumi per ogni singola via, in ciò ci è stato d'ausilio il software per la modellazione delle memorie caches: CACTI.
CACTI è un software sviluppato dalla Compaq Computer Corporation Western Research Laboratory di Palo Alto in California, esso integra la modellazione di tempi di accesso, area, proporzioni, e consumo di potenza per le memorie caches, permettendo tramite la valutazione di tali quantità una progettazione coerente con i vari impieghi.
Nella versione da noi utilizzata (3.2) sono state introdotte delle migliorie frutto delle ultime ricerche in ambito microelettronico per la riduzione dei consumi, quali i Sense Amplifier; una migliore gestione della distribuzione spaziale dei componenti, atta ad ottenere la migliore configurazione possibile in termini di accesso e consumo oltre che di minimo spazio occupato, è stata inoltre introdotta la possibilità di modellare i sottobanchi in base alle proprie esigenze, rendendo tale strumento molto più versatile rispetto alle versioni precedenti.

La sintassi da utilizzare è la seguente:

```
cacti 1048576 64 1 0.50 16
```

il primo parametro sta ad indicare la dimensione della memoria caches espressa in bytes, nell'esempio 1Mb, il secondo indica la cacheline, ossia il numero di connessioni necessarie ad indirizzare il blocco base di cache, nel caso in esame il blocco è da 64 byte, il terzo l'assosiatività della cache, in questo caso la cache è fully associative, il quarto indica la tecnologia di modellazione, sopra di 50 nanometri, in ultimo sono indicati il numero di sottobanchi in cui suddividere la cache.
Ci sono anche altre opzioni ma nel nostro caso sono queste quelle che abbiamo preso in considerazione, per una trattazione più dettagliata si rimanda a [12].

3.2.2 Valutazione consumi cache standard

Al fine di determinare un parametro di confronto per le caches NUCA, sono state svolte alcune simulazioni con CACTI per quanto riguarda le caches standard ed i risultati ottenuti sono riportati di seguito.
Per tutti i grafici che seguiranno è stata presa come dimensione della cacheline (blocco)64 byte, e le caches sono tutte fully associative.

Come prima tipologia prenderemo in considerazione delle caches il cui banco base è di 64 Kb e analizzeremo come variano consumi e tempi di accesso al variare, oltre che della tecnologia di produzione, anche del numero di banchi in cui le cache saranno aggregate.

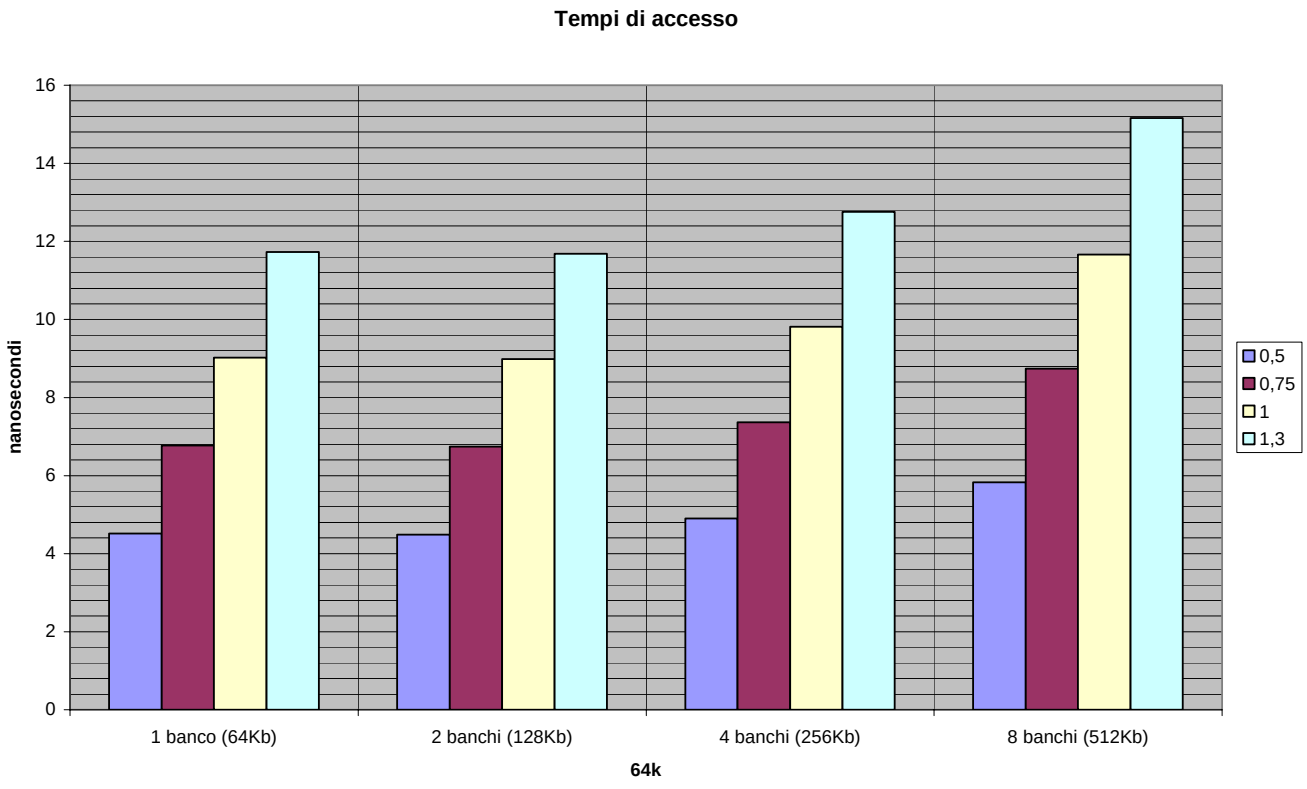

Grafico 1: Tempi di accesso espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 64Kb 128Kb 256Kb 512Kb.

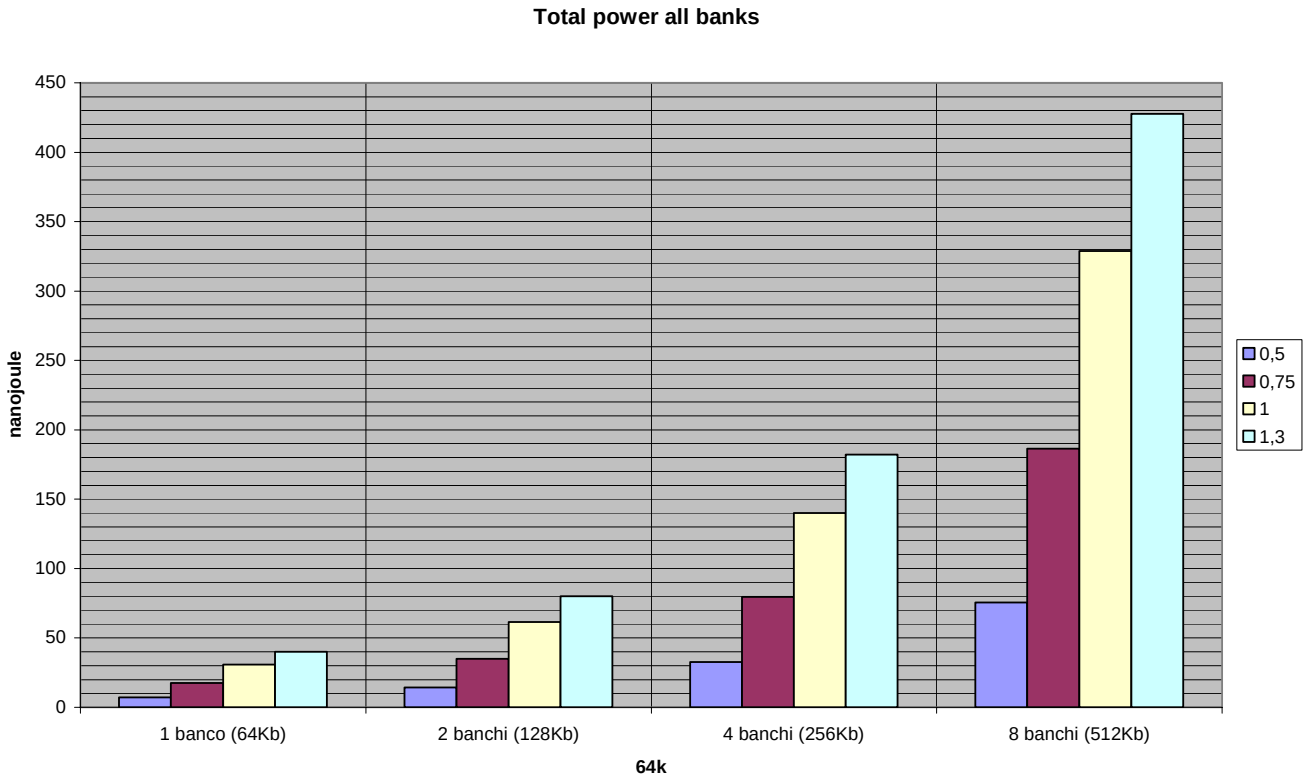

Grafico 2: Consumo di potenza di tutti i banchi espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 64Kb 128Kb 256Kb 512Kb.

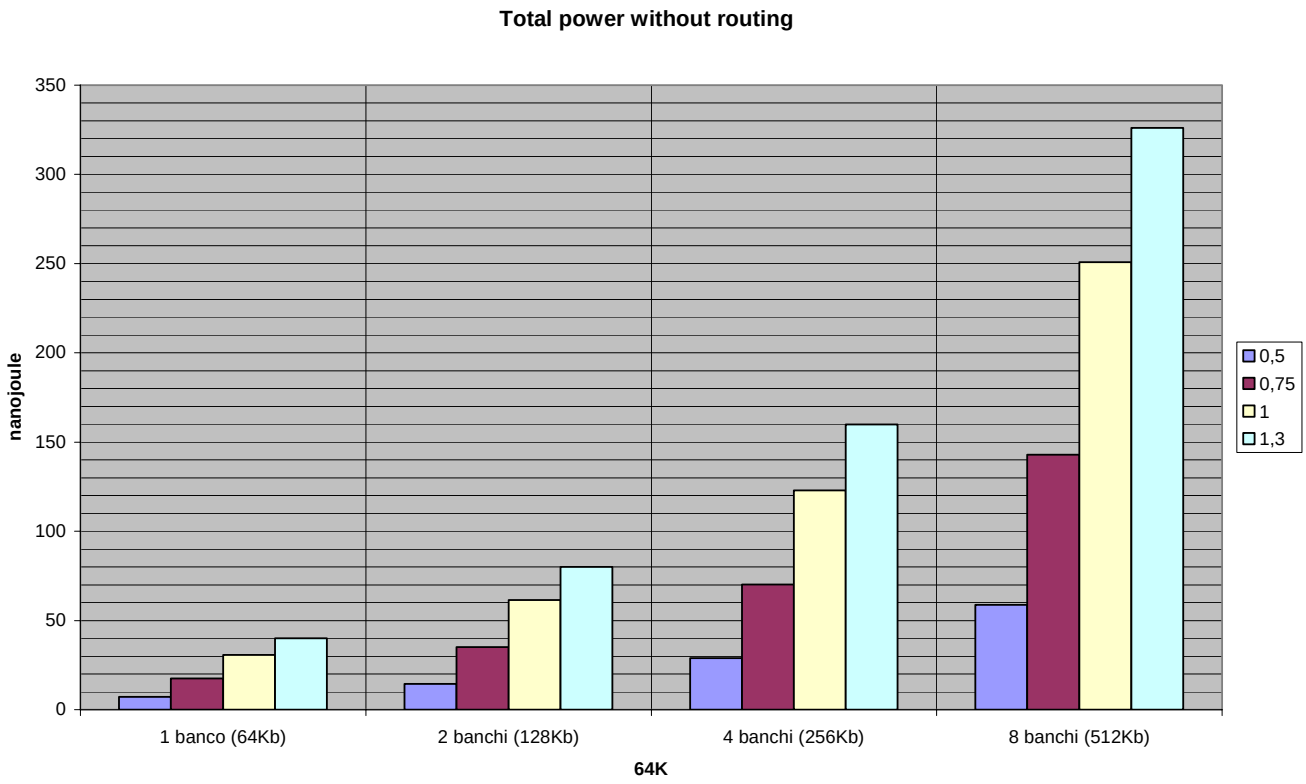

Grafico 3: Consumo di potenza senza routing espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 64Kb 128Kb 256Kb 512Kb.

Per questa architettura di cache si può constatare che all'aumentare del numero dei banchi (e conseguentemente della dimensione della cache)c'è anche un aumento significativo di consumi, meno per i tempi di accesso.

Vedremo ora cache la cui dimensione base del banco è di 128Kb l' aggregazione in sottobanchi rimane la stessa, ovviamente si modelleranno caches di dimensioni doppie rispetto alle precedenti:

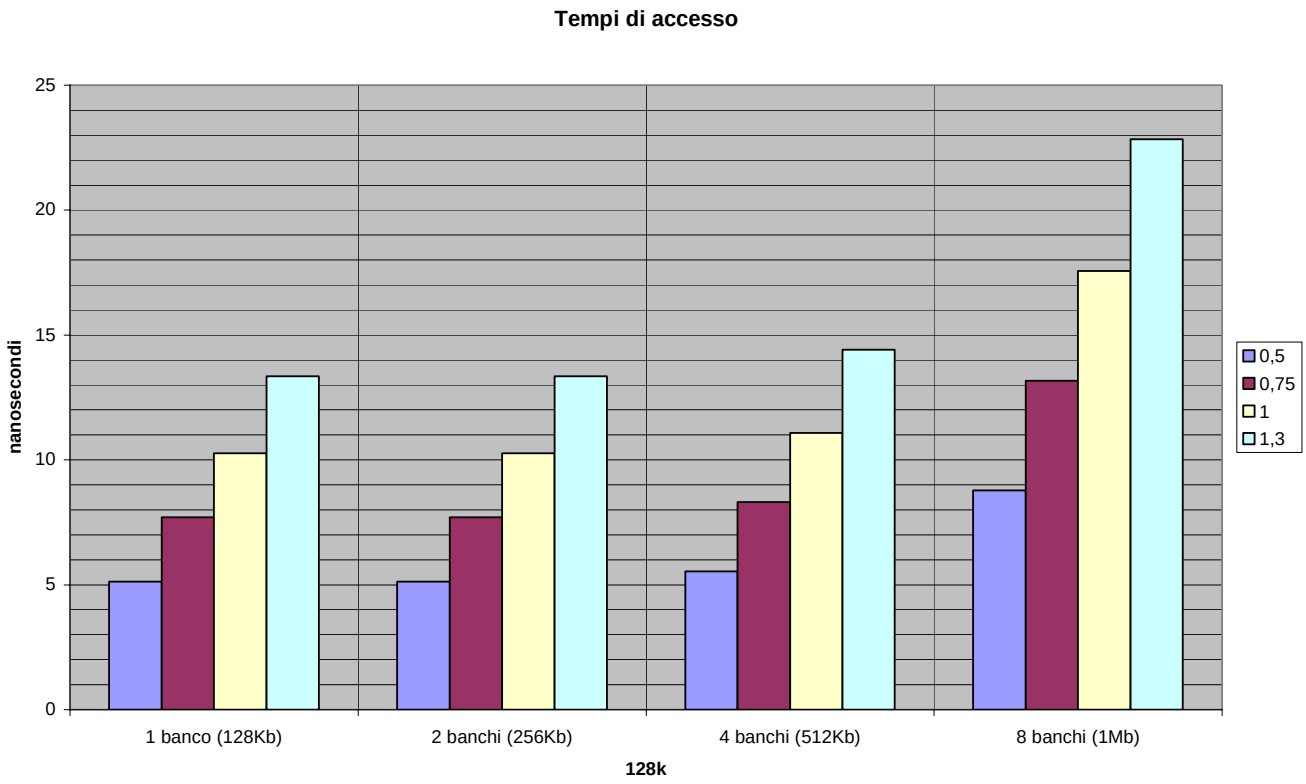

Grafico 4: Tempi di accesso espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 128Kb 256Kb 512Kb 1Mb.

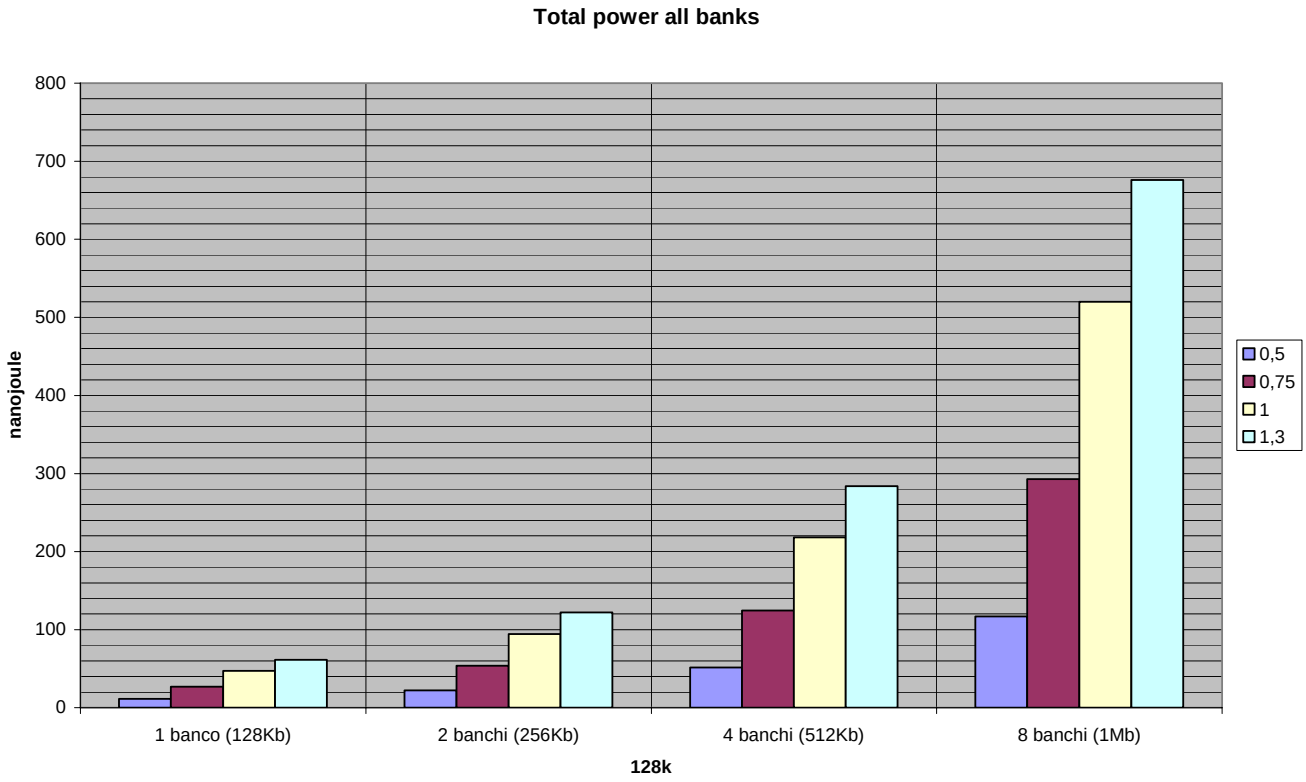

Grafico 5: Consumo di potenza di tutti i banchi espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 128Kb 256Kb 512Kb 1Mb.

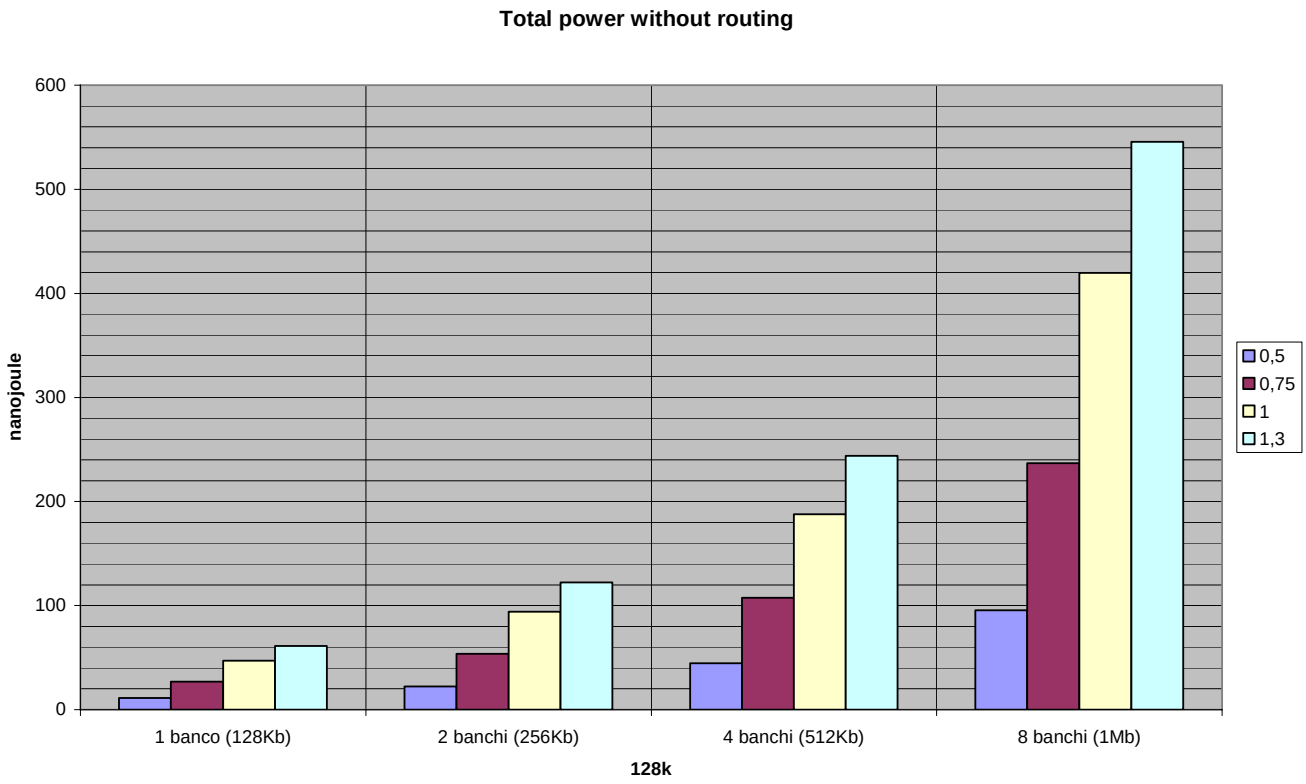

Grafico 6: Consumo di potenza senza routing espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 128Kb 256Kb 512Kb 1Mb.

Per queste cache si possono fare le stesse affermazioni fatte in precedenza; vediamo ora gli stessi grafici di sopra considerando però che la dimensione base del banco è di 256Kb l' aggregazione in sottobanchi rimane la stessa, perciò si otterranno caches di dimensione 256Kb, 512Kb, 1Mb, 2Mb.

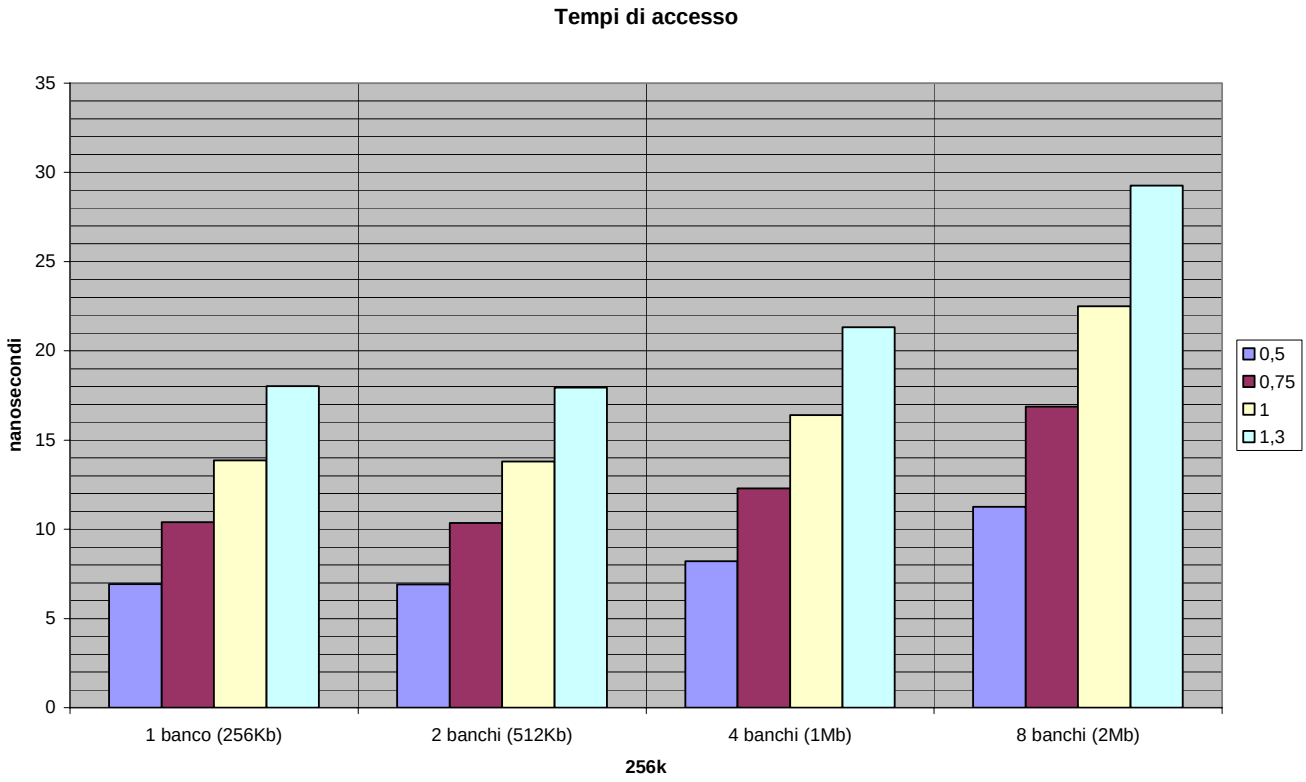

Grafico 7: Tempi di accesso espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 256Kb 512Kb 1Mb 2Mb.

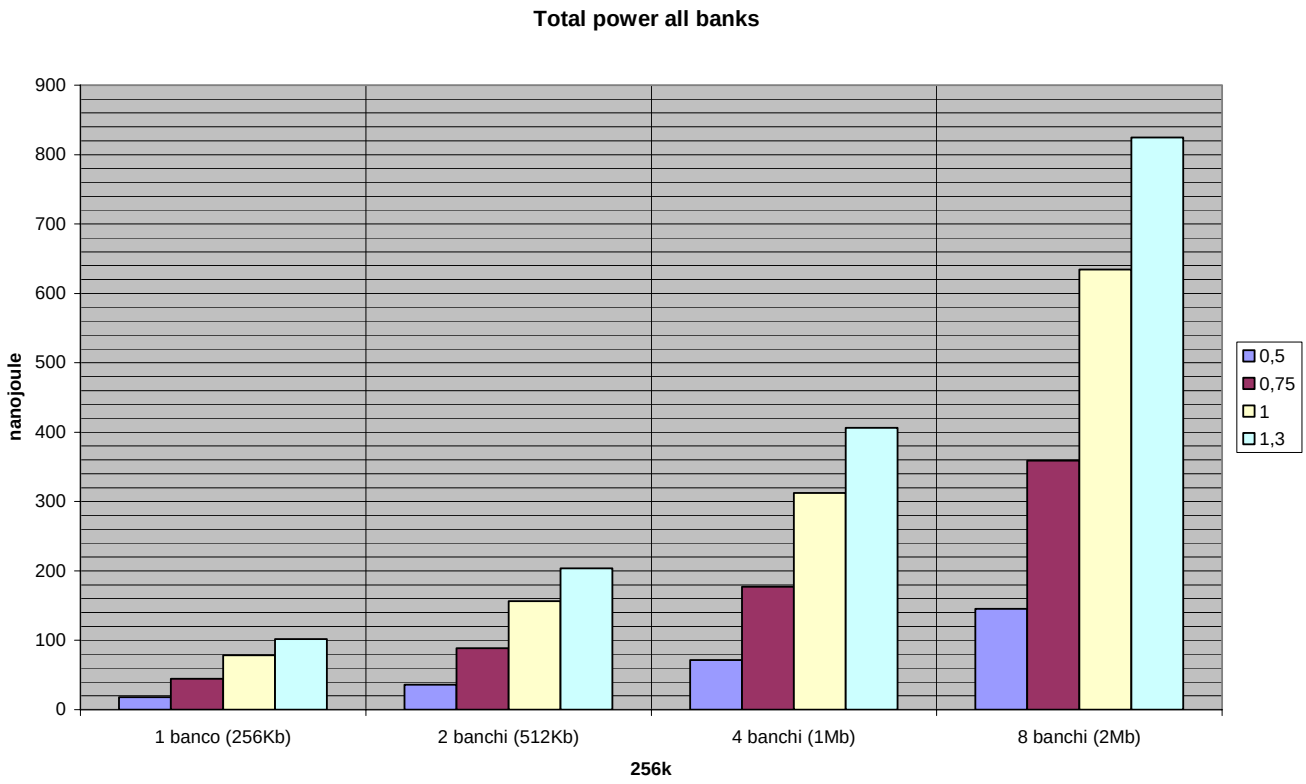

Grafico 8: Consumo di potenza di tutti i banchi espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 256Kb 512Kb 1Mb 2Mb.

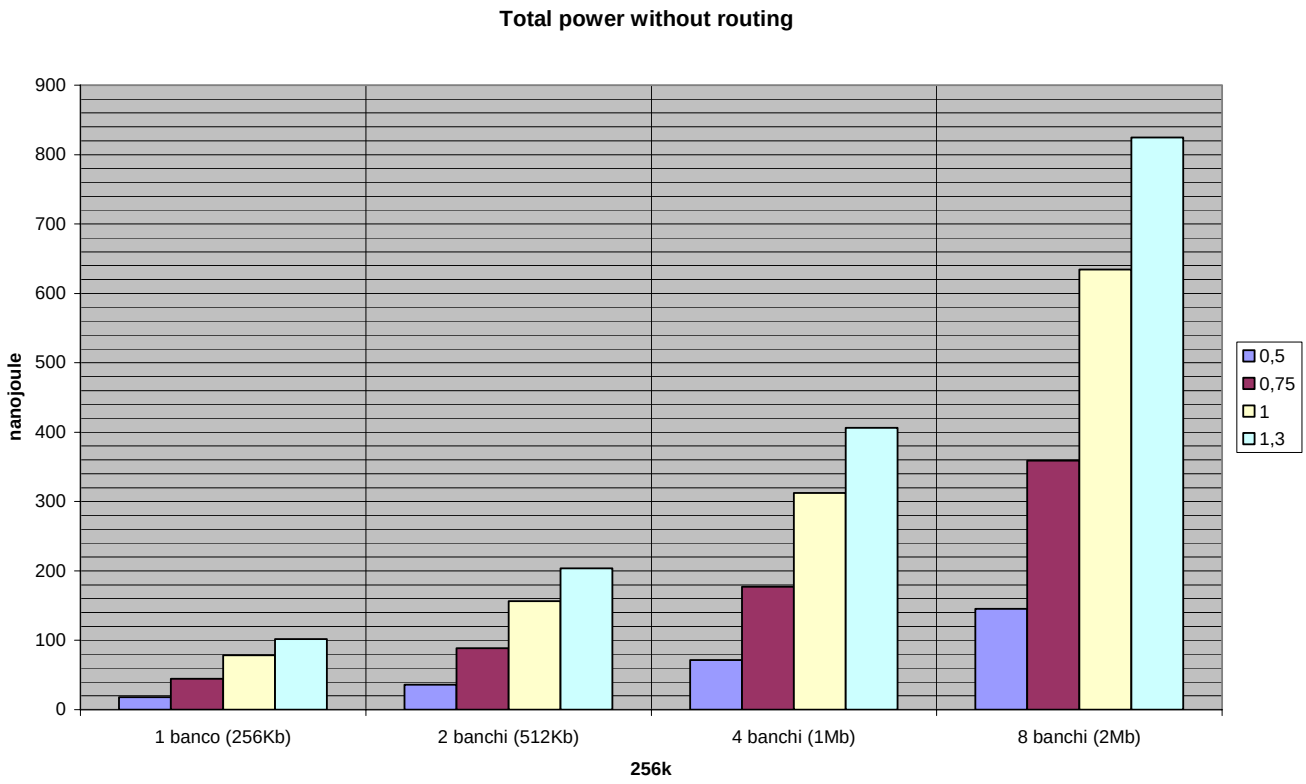

Grafico 9: Consumo di potenza senza routing espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 256Kb 512Kb 1Mb 2Mb.

Infine riportiamo i grafici relativi ad un banco base di 512Kb, cui corrisponderanno all'aumentare del numero dei banchi caches di dimensione: 512Kb 1Mb 2Mb 4Mb

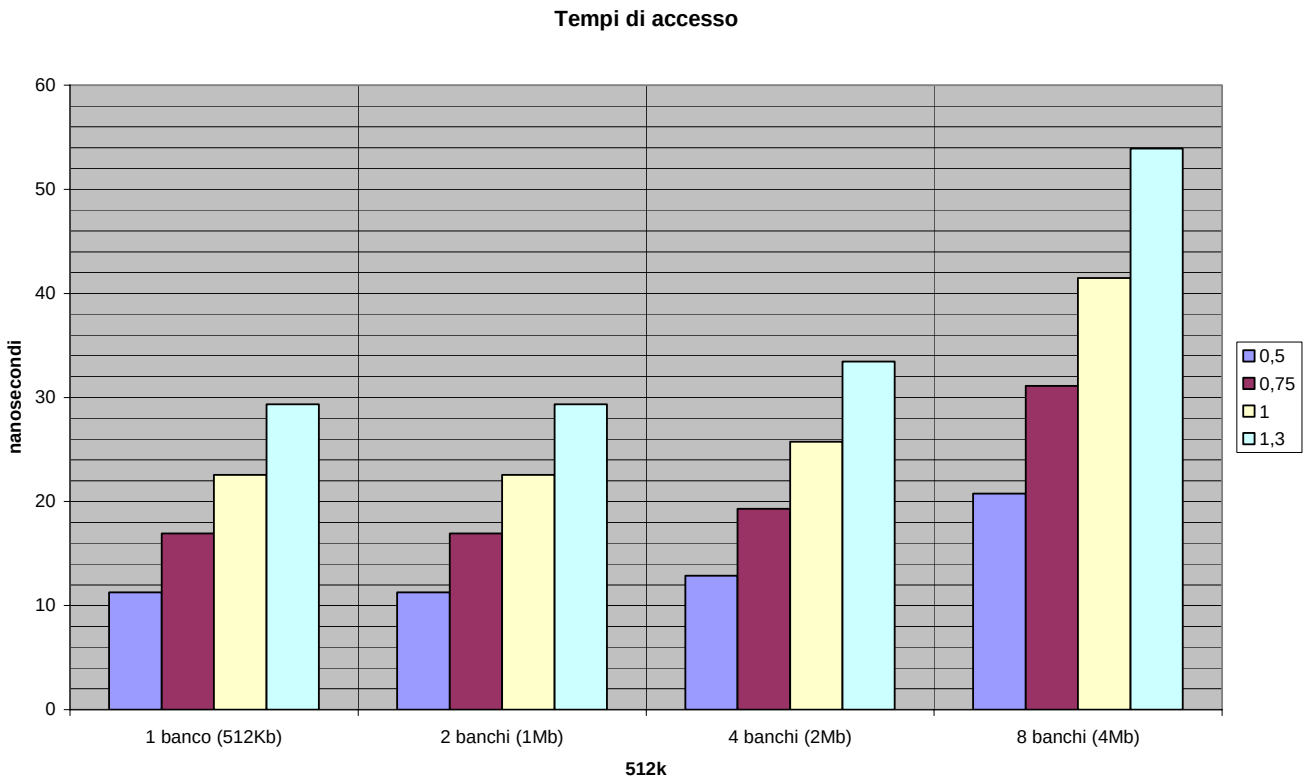

Grafico 10: Tempi di accesso espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 512Kb 1Mb 2Mb 4Mb.

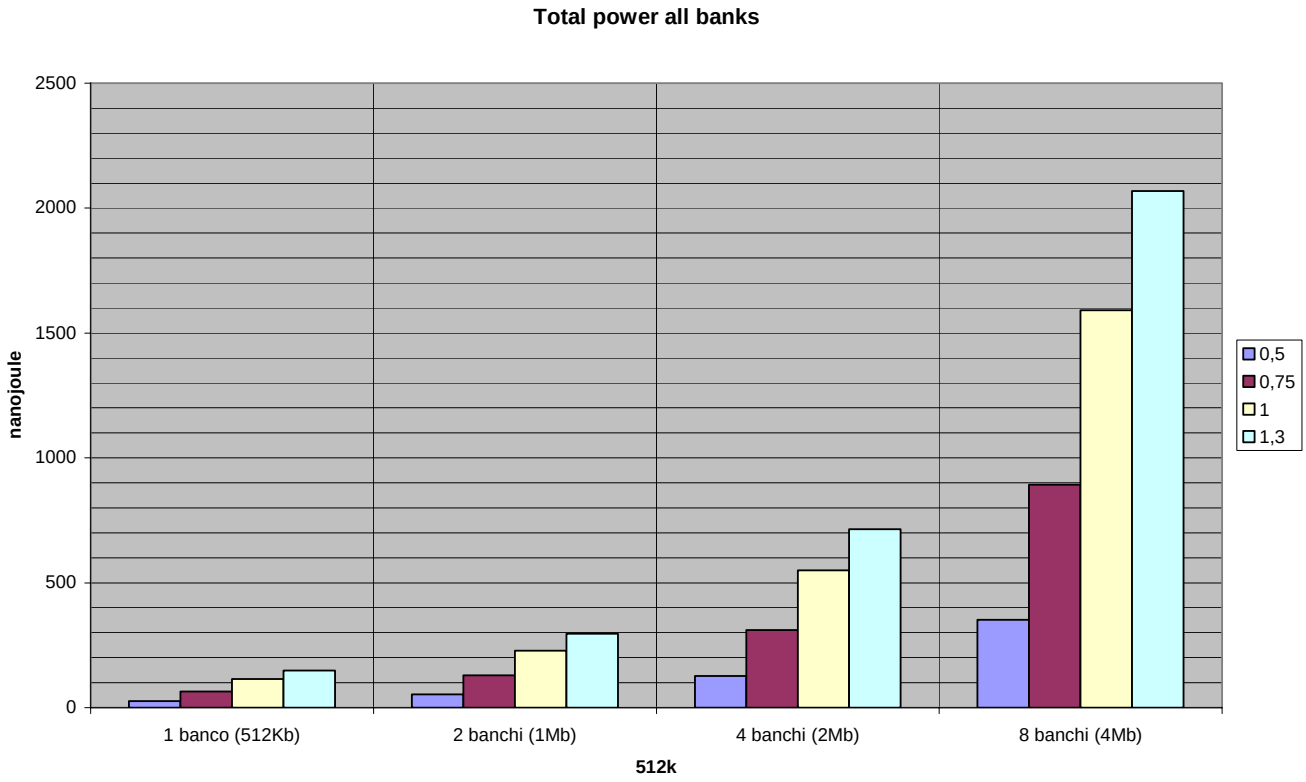

Grafico 11: Consumo di potenza di tutti i banchi espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 512Kb 1Mb 2Mb 4Mb.

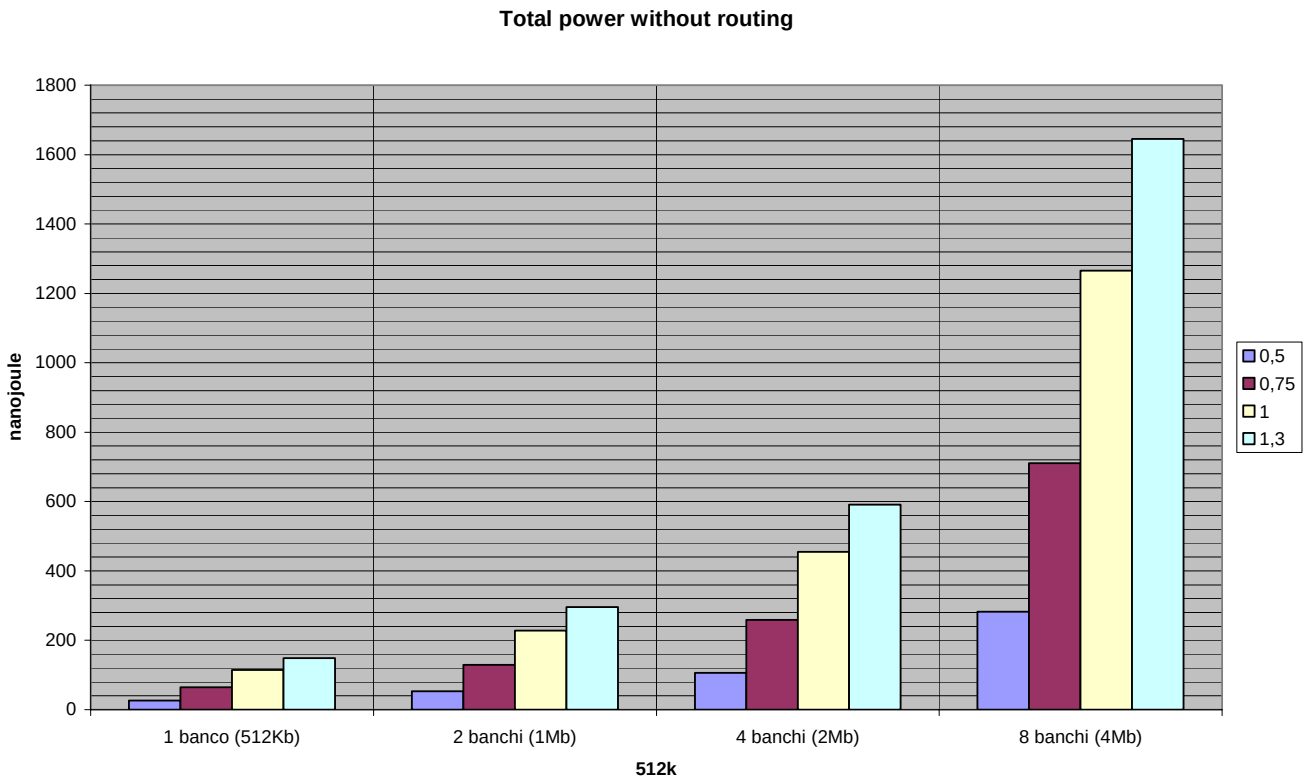

Grafico 12: Consumo di potenza senza routing espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8 cui corrispondono cache di 512Kb 1Mb 2Mb 4Mb.

In questa seconda serie di grafici invece metteremo in evidenza come cambiano tempi di accesso e consumi delle caches standard, per vari tagli, da 1Mb fino a 16Mb; al variare del numero dei sottobanchi che le compongono ed ovviamente al variare della tecnologia.

Cominciamo con il taglio da 1 Mb.

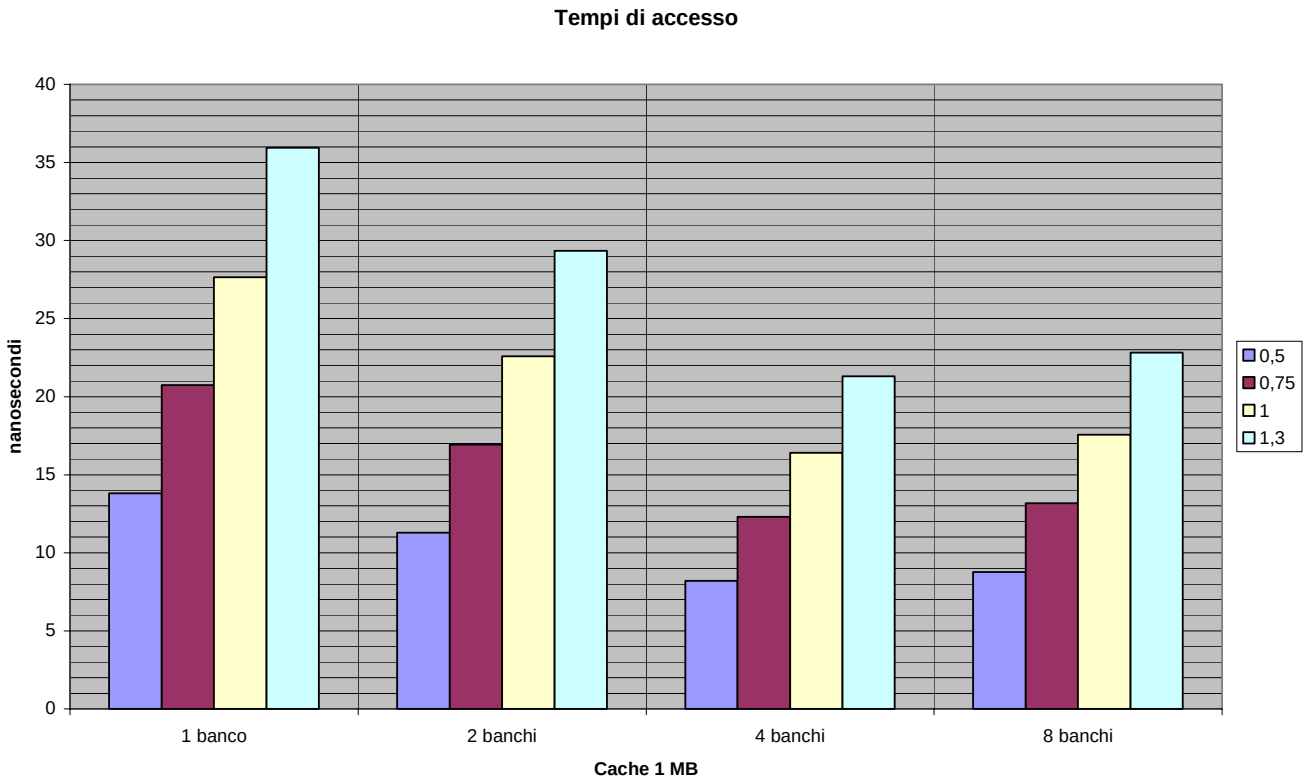

Grafico 13: Tempi di accesso ad una cache da 1Mb espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

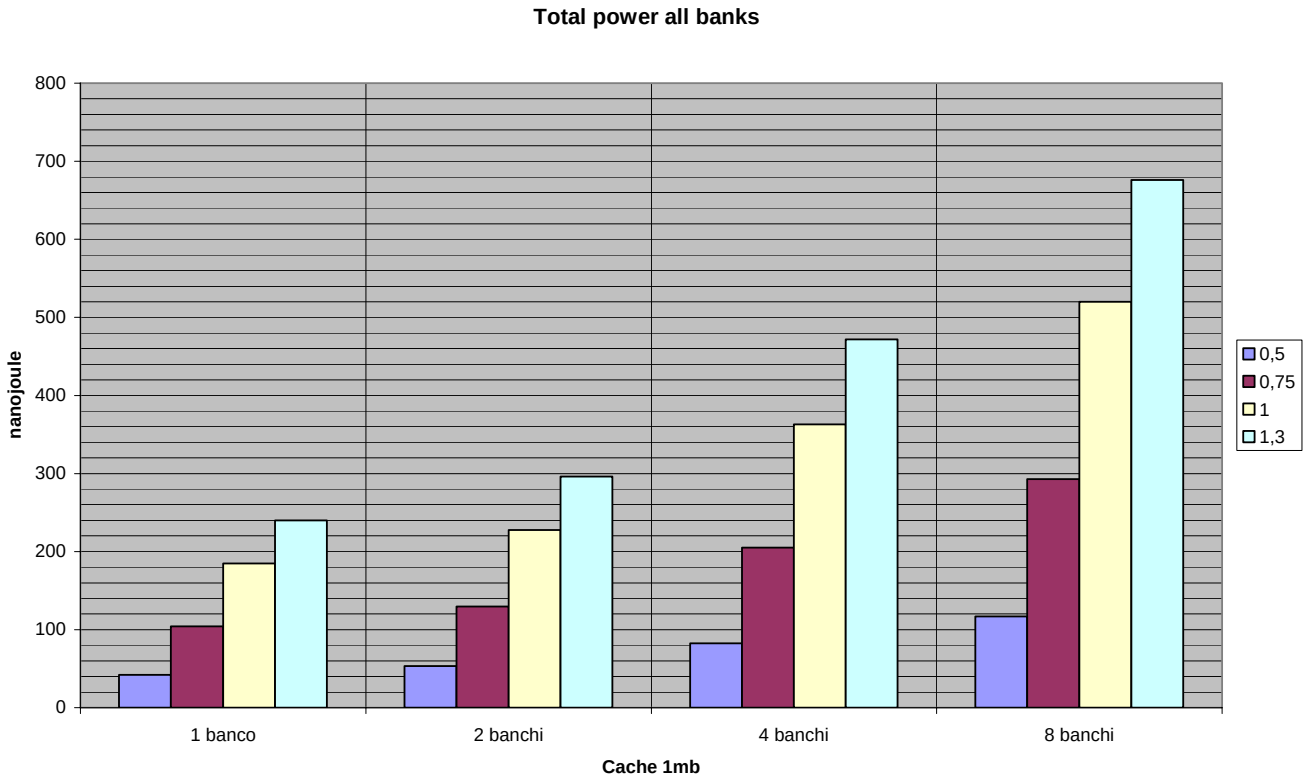

Grafico 14: Consumo di potenza di tutti i banchi per una cache da 1Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

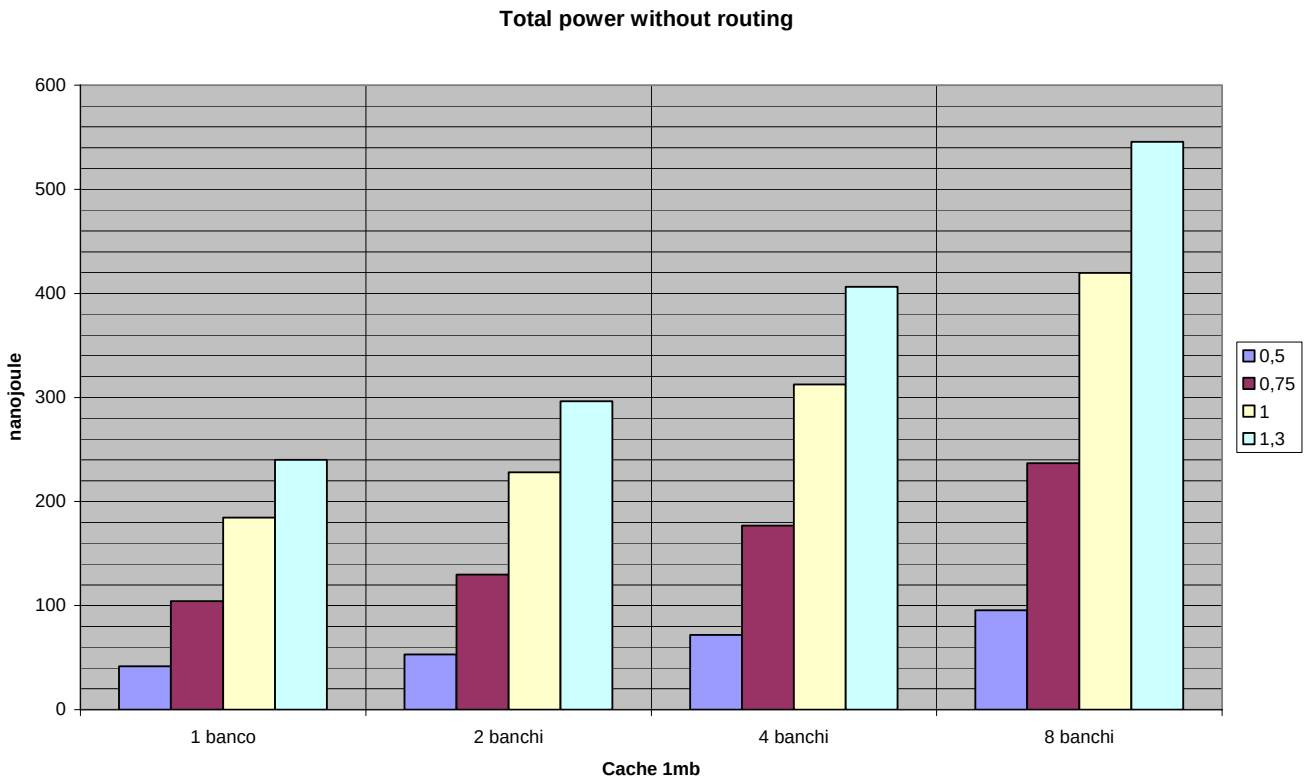

Grafico 15: Consumo di potenza senza routine per una cache da 1Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

Si può notare che mentre i tempi di accesso diminuiscono all'aumentare del numero di sottobanchi, raggiungendo la massima efficienza nella suddivisione in 4 sottobanchi, per quanto riguarda i consumi ci sia una situazione opposta, i consumi aumentano all'umentare dei sottobanchi.
Seguono ora i grafici per cache da 2Mb.

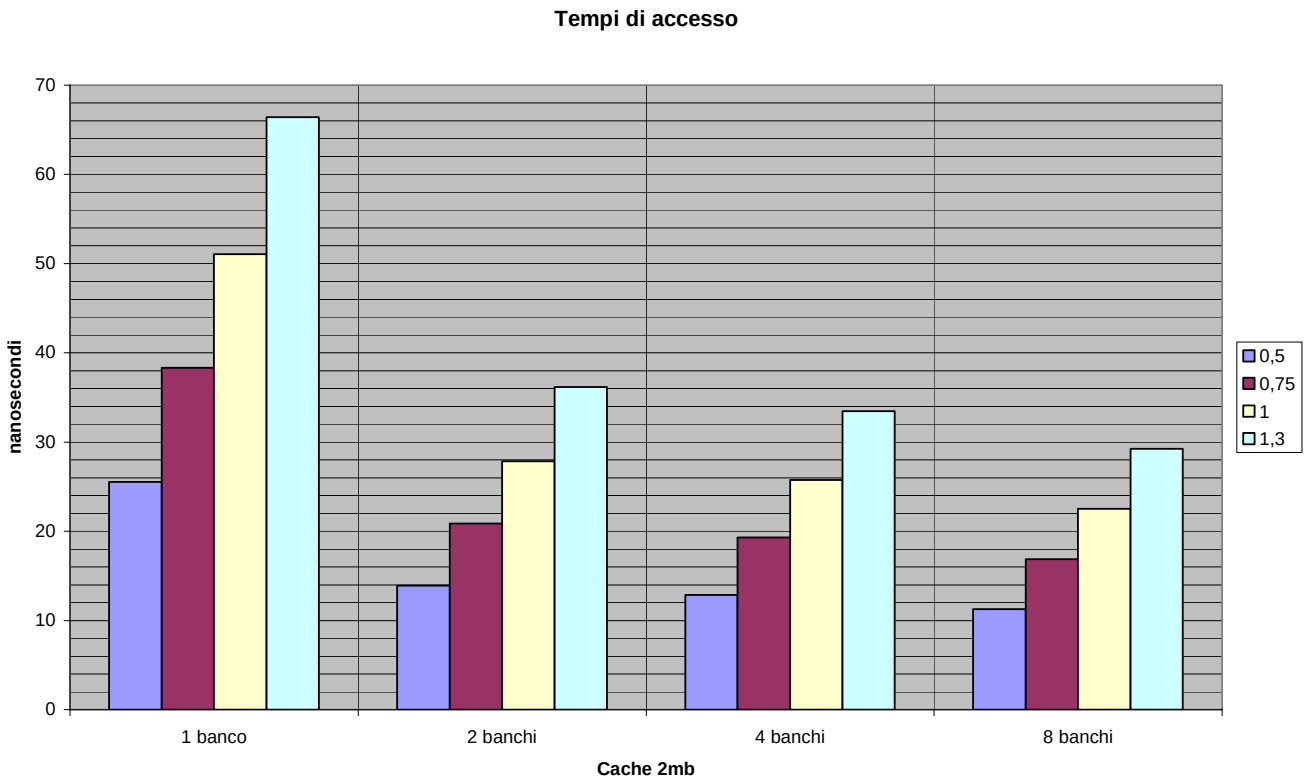

Grafico 16: Tempi di accesso ad una cache da 2 Mb espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

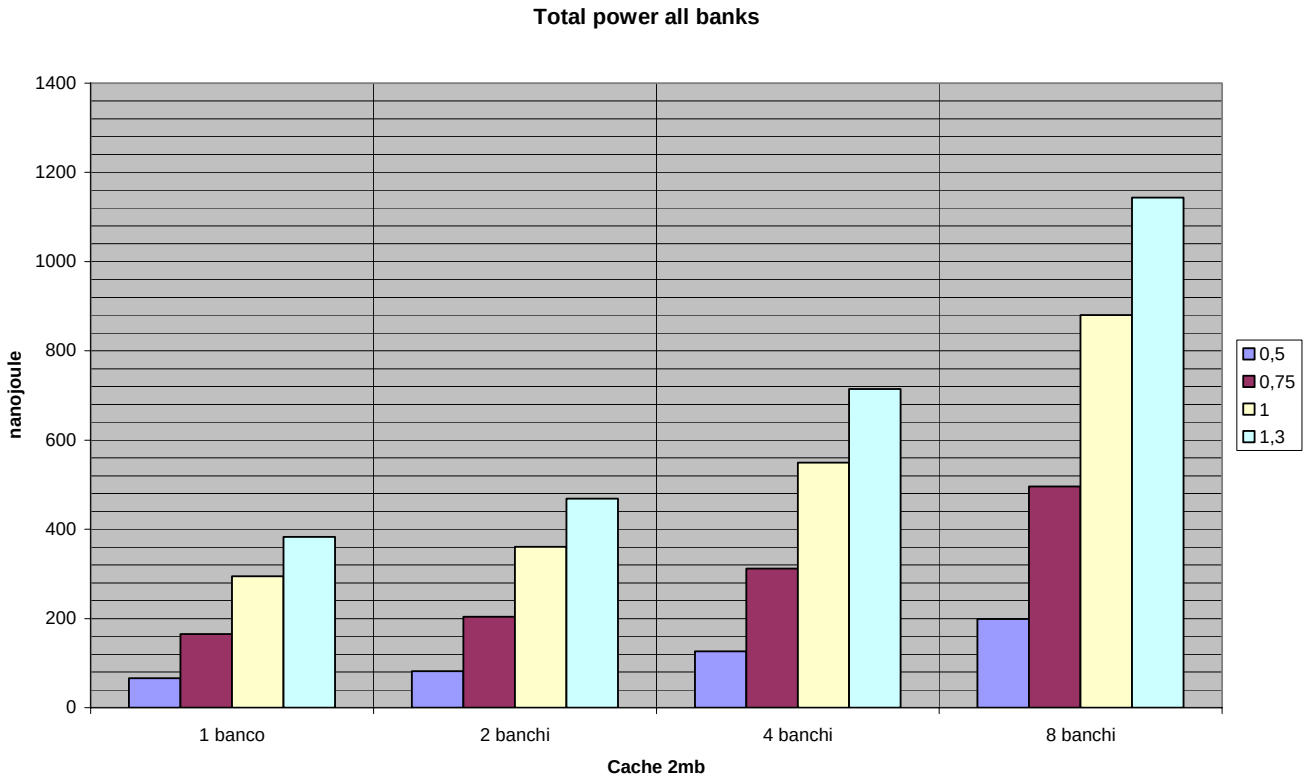

Grafico 17: Consumo di potenza di tutti i banchi per una cache da 2 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

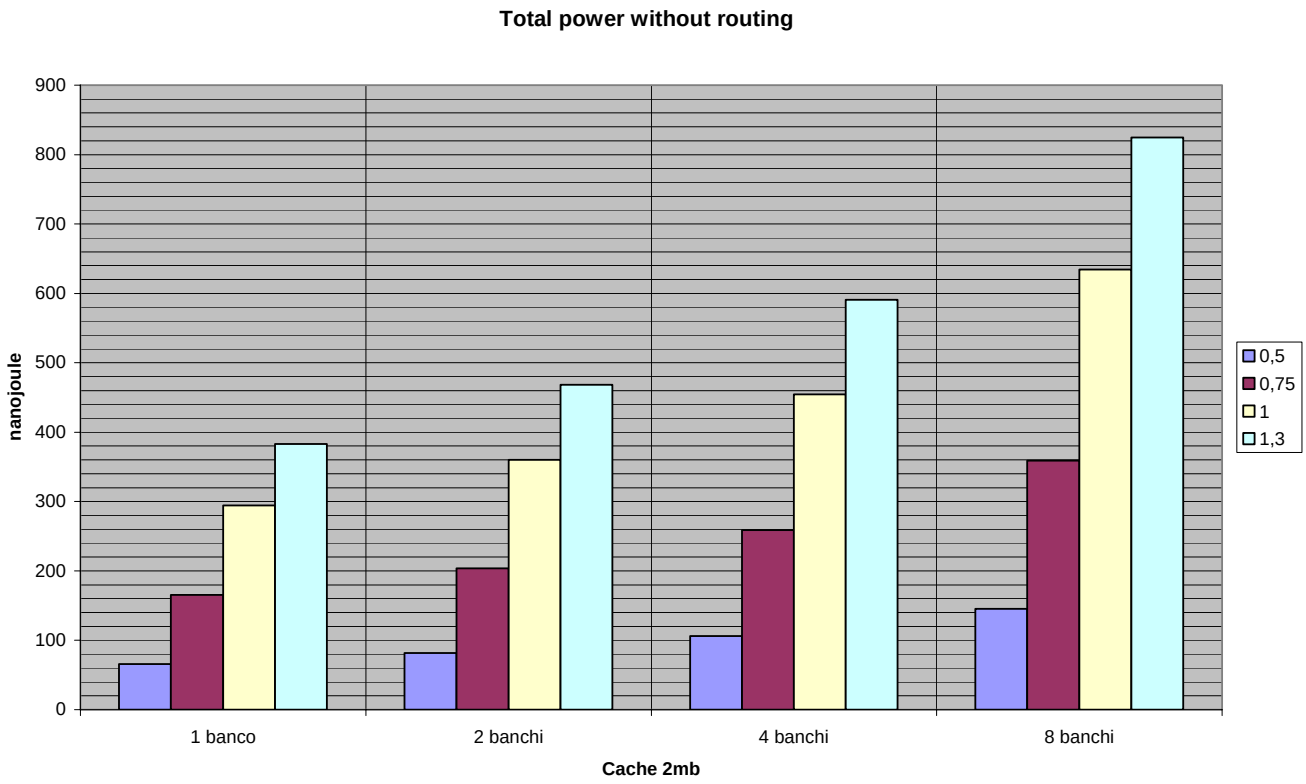

Grafico 18: Consumo di potenza senza routine per una cache da 2 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

Anche in questo caso come in precedenza si nota che i tempi di accesso diminuiscono all'aumentare del numero dei banchi(con questa dimensione si raggiungono i risultati migliori per 2 sottobanchi), mentre i consumi aumentano.

Di seguito i grafici relativi ad una cache da 4Mb.

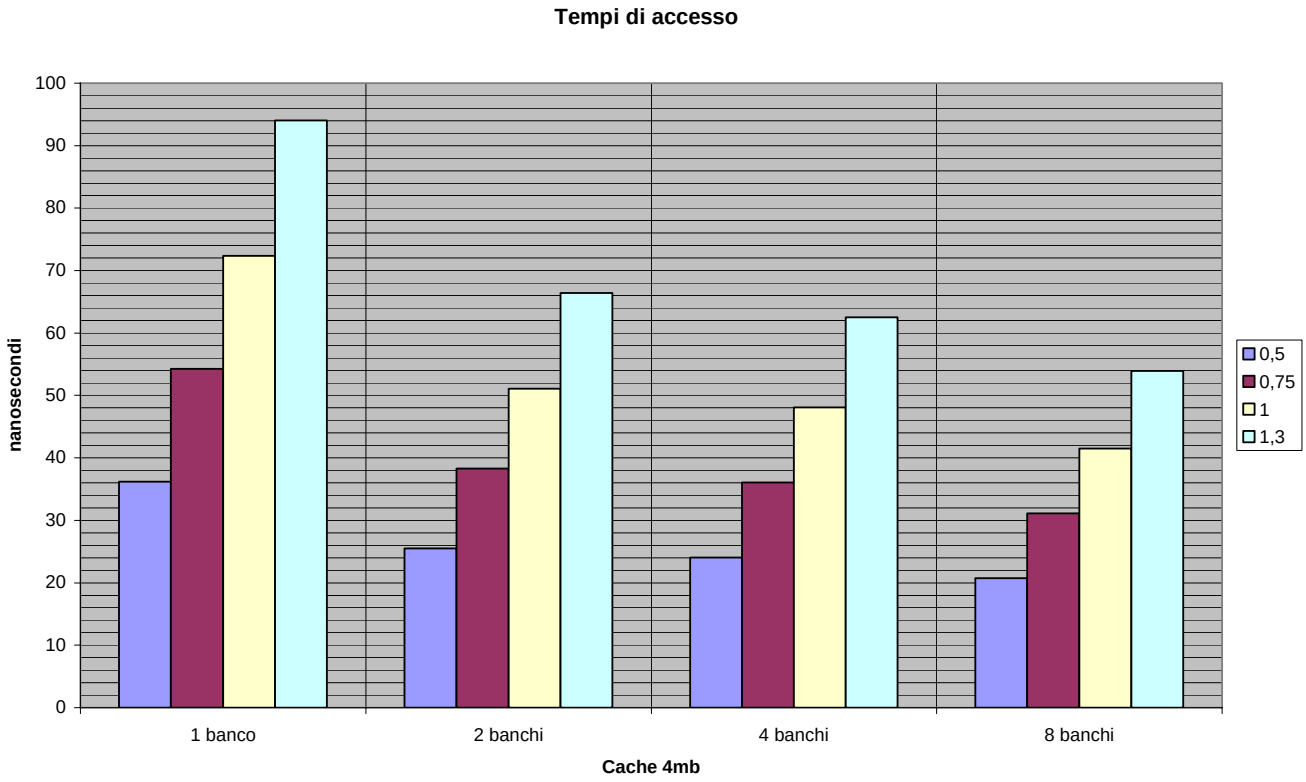

Grafico 19: Tempi di accesso ad una cache da 4 Mb espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

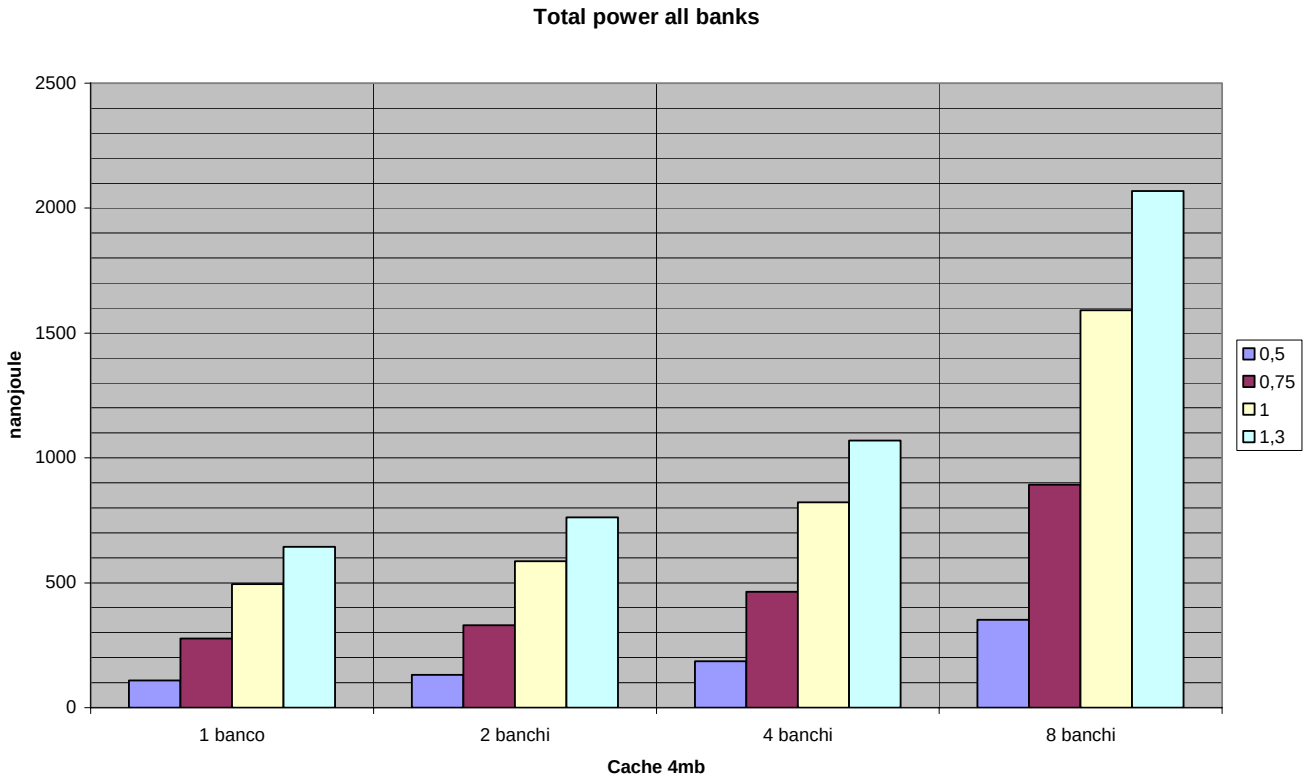

Grafico 20: Consumo di potenza di tutti i banchi per una cache da 4 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

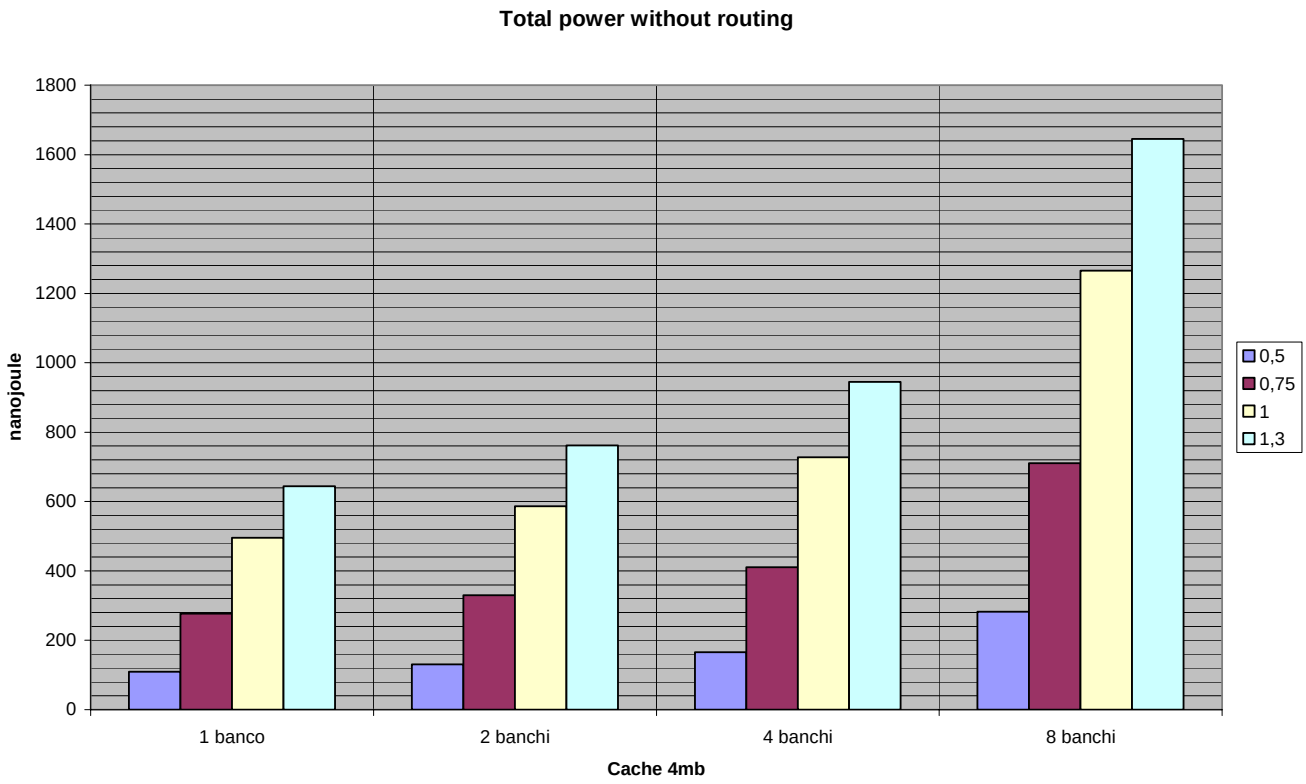

Grafico 21: Consumo di potenza senza routine per una cache da 4 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

Nulla cambia, eccezion fatta che i tempi di accesso sono minimi per il massimo numero di sottobanchi presi in considerazione(8).

Ora una cache da 8Mb.

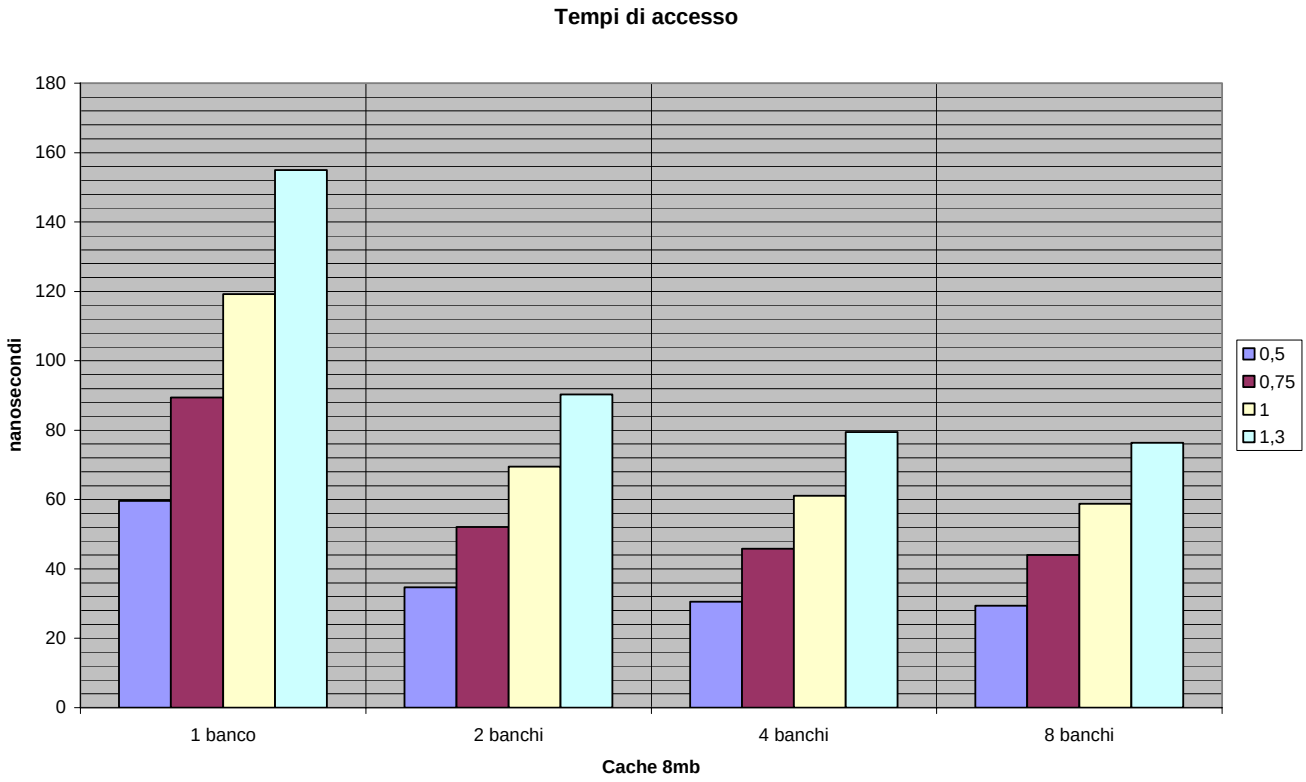

Grafico 22: Tempi di accesso ad una cache da 8 Mb espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

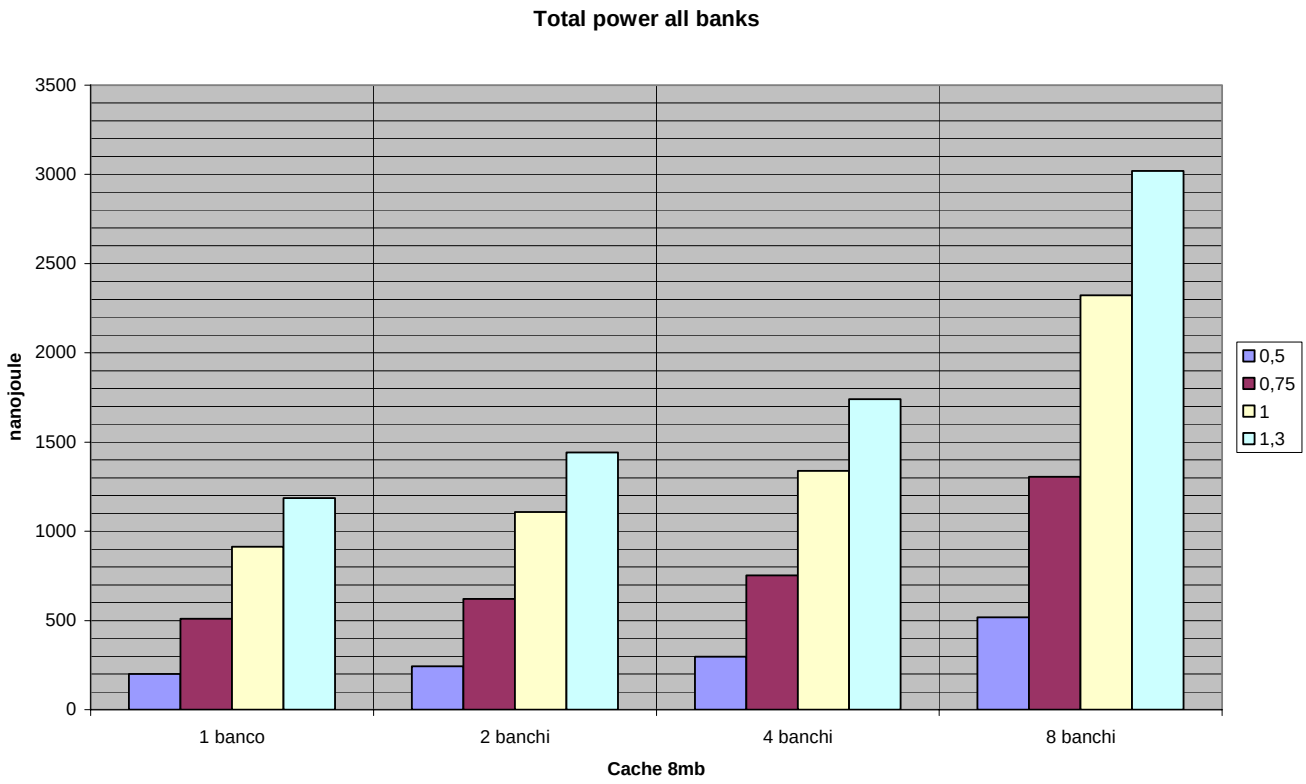

Grafico 23: Consumo di potenza di tutti i banchi per una cache da 8 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

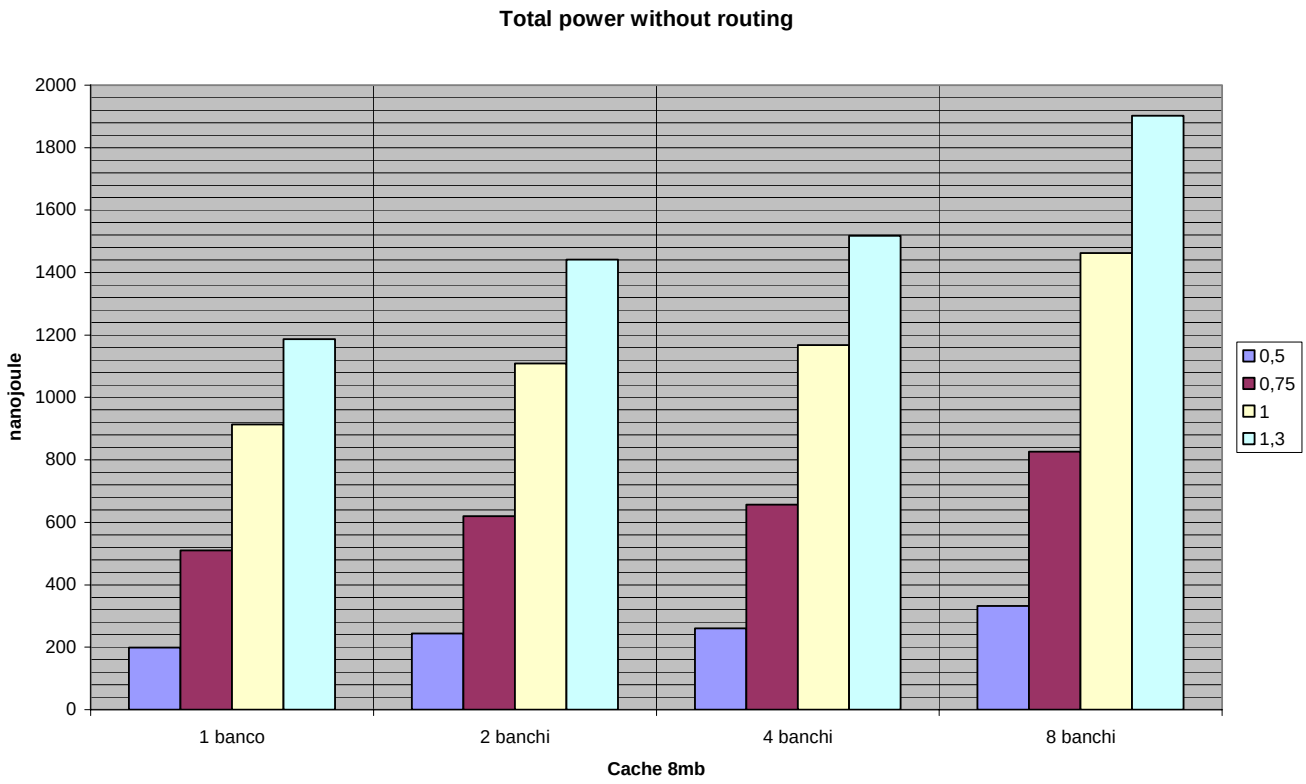

Grafico 24: Consumo di potenza senza routine per una cache da 8 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

Per questo taglio le stime, come in precedenza, indicano che all'aumentare dei sottobanchi, aumentano i consumi e diminuiscono i tempi di accesso. Seguono infine i grafici per una cache da 16Mb.

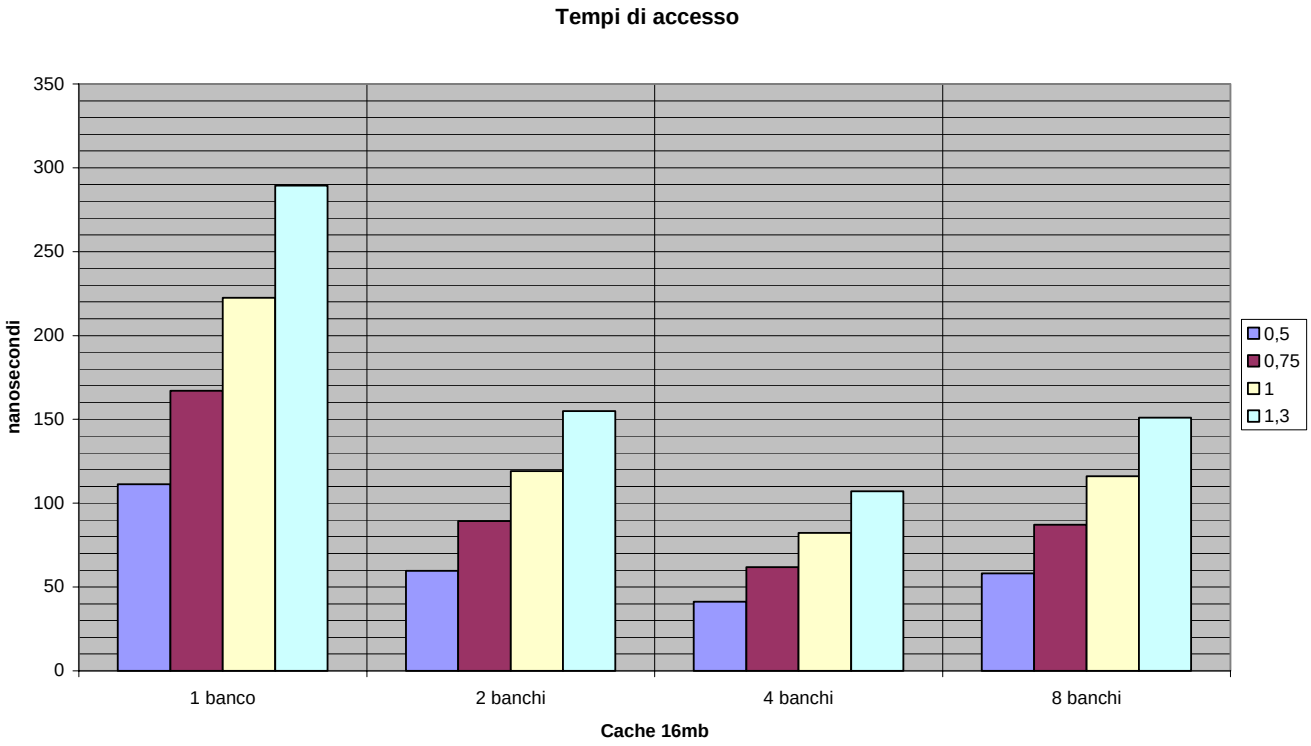

Grafico 25: Tempi di accesso ad una cache da 16 Mb espressi in nanosecondi al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

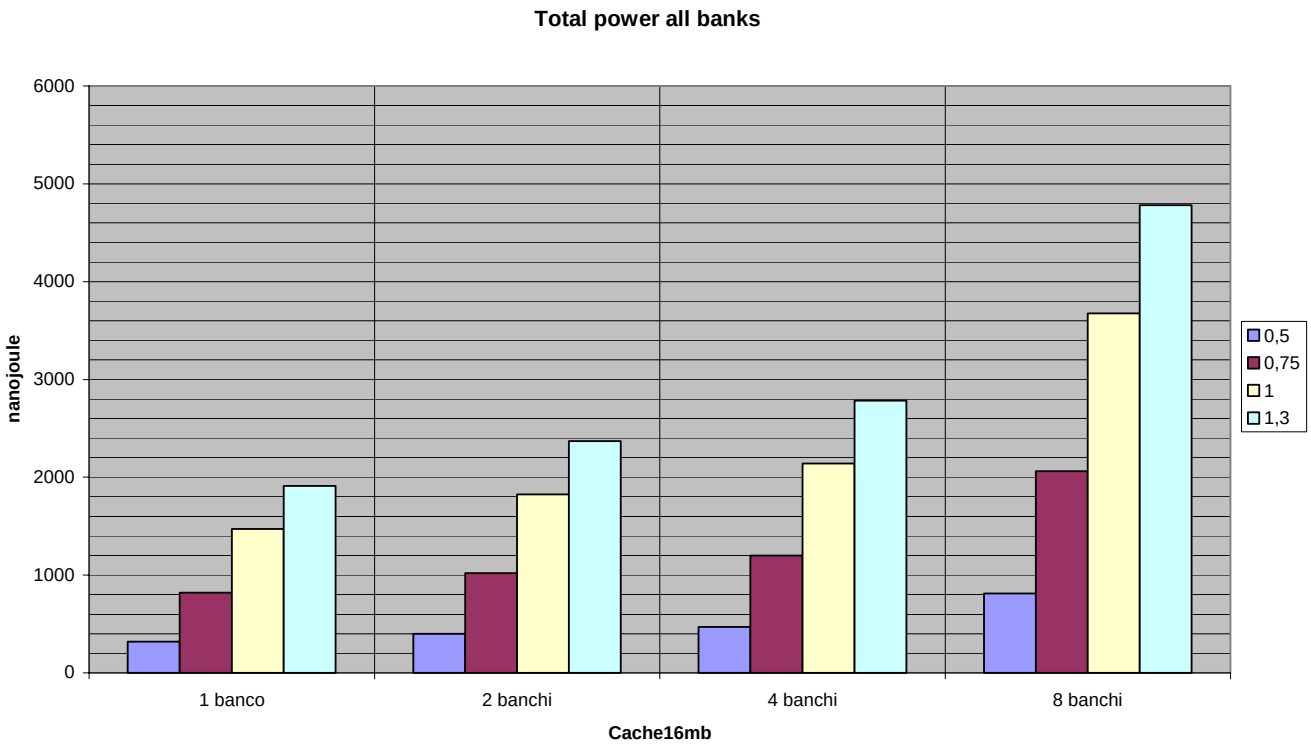

Grafico 26: Consumo di potenza di tutti i banchi per una cache da 16 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

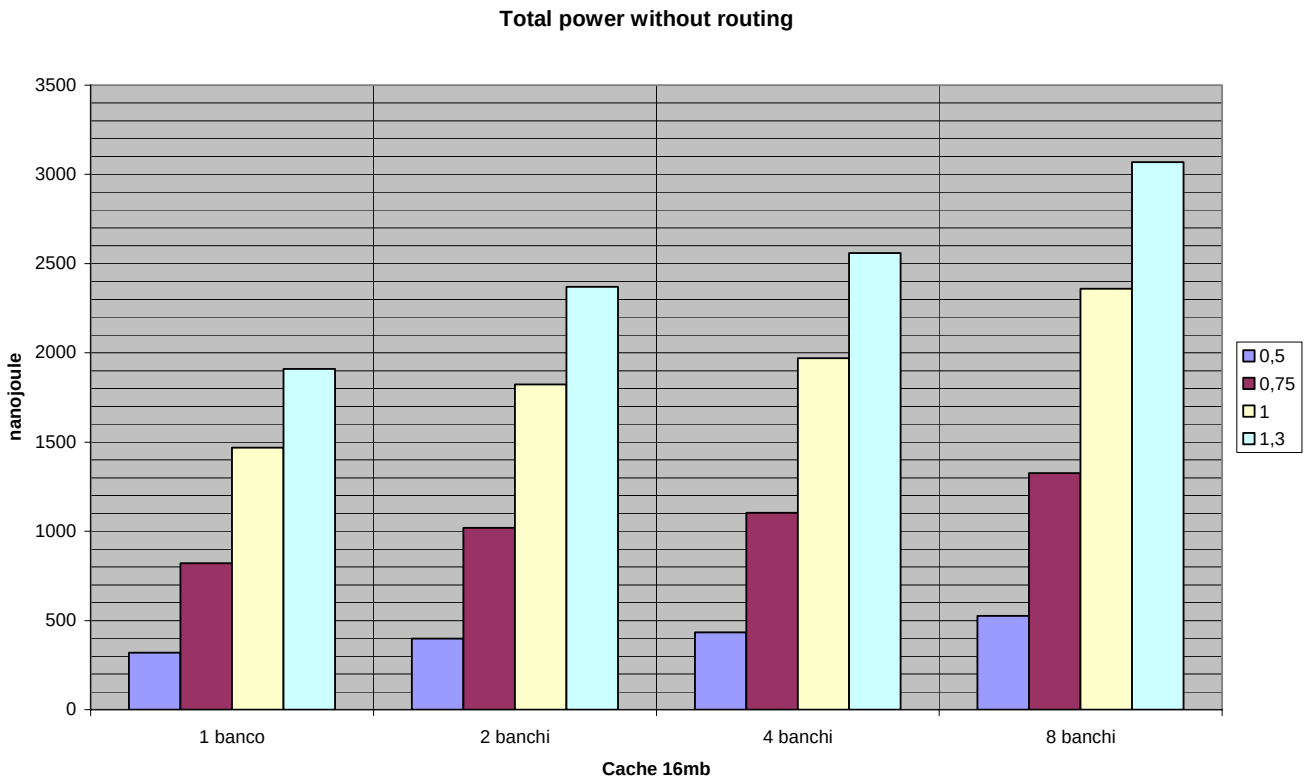

Grafico 27: Consumo di potenza senza routine per una cache da 16 Mb espresso in nanojoule al variare della tecnologia di produzione e del numero di sottobanchi componenti la cache 1,2,4,8.

3.2.3 Valutazione consumi NUCA

Per quanto riguarda le NUCA, le quantità tabulate di seguito sono riferite ad ogni singola via del taglio in esame, considerando la via come una singola cache fully associative, la cui dimensione del blocco base è pari a 64 bytes, tecnologia a 50 nanometri, e numero di sottobanchi pari a quelli elencati precedentemente per ogni via.

D – NUCA 16 MB	
Access time (ns)	8,48077
Total power all banks (nj)	165,723
Total power without routing (nj)	117,373
Total routing power (nj)	48,3497
Maximum Bank Power (nJ)	13,0543

Tabella 1: Parametri di riferimento per una D-.NUCA da 16 Mb.

Tali valori si riferisco ad una via di una D-NUCA cache di dimensione pari ad 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base pari a 64 bytes e numero di sottobanchi pari a 16.

D – NUCA 8 MB	
Access time (ns)	8,48077
Total power all banks (nj)	165,723
Total power without routing (nj)	117,373
Total routing power (nj)	48,3497
Maximum Bank Power (nJ)	13,0543

Tabella 2: Parametri di riferimento per una D-.NUCA da 8 Mb.

Tali valori si riferisco ad una via di una D-NUCA cache di dimensione pari ad 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base pari a 64 bytes e numero di sottobanchi pari a 16.

D – NUCA 4 MB	
Access time (ns)	8,78
Total power all banks (nj)	117,027
Total power without routing (nj)	95,2897
Total routing power (nj)	21,7371
Maximum Bank Power (nJ)	16,3932

Tabella 3: Parametri di riferimento per una D-.NUCA da 4 Mb.

Tali valori si riferisco ad una via di una D-NUCA cache di dimensione pari ad 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base pari a 64 bytes e numero di sottobanchi pari a 8.

D – NUCA 2 MB	
Access time (ns)	5,53926
Total power all banks (nj)	51,2527
Total power without routing (nj)	44,6835
Total routing power (nj)	6,56919
Maximum Bank Power (nJ)	14,4727

Tabella 4: Parametri di riferimento per una D-.NUCA da 2 Mb.

Tali valori si riferisco ad una via di una D-NUCA cache di dimensione pari ad 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base pari a 64 bytes e numero di sottobanchi pari a 4.

Per quanto riguarda le TD-NUCA invece è stato necessario differenziare oltre che in base al taglio, anche alla via le varie tipologie architetturali da considerare; come per il caso delle D-NUCA è stata considerata la tecnologia di realizzazione dei transistor a 50 nanometri e la fully associativity delle singole vie, sono però state differenziate le dimensioni di ogni singola via in base alle specifiche sopra esposte.

I dati rilevati sono quindi i seguenti:

TD – NUCA 8 MB	
4 x 1MB Bank	
Access time (ns)	8,48077
Total power all banks (nj)	165,723
Total power without routing (nj)	117,373
Total routing power (nj)	48,3497
Maximum Bank Power (nJ)	13,0543
4 x 512KB Bank	
Access time (ns)	5,82949
Total power all banks (nj)	75,7005
Total power without routing (nj)	58,7817
Total routing power (nj)	16,9187
Maximum Bank Power (nJ)	10,6073
4 x 256KB Bank	
Access time (ns)	4,90658
Total power all banks (nj)	32,5712
Total power without routing (nj)	28,8425
Total routing power (nj)	3,72875
Maximum Bank Power (nJ)	9,09102
4 x 128KB Bank	

Access time (ns)	4,49215
Total power all banks (nj)	14,435
Total power without routing (nj)	14,435
Total routing power (nj)	0
Maximum Bank Power (nJ)	7,22064

Tabella 5: Parametri di riferimento per una TD-.NUCA da 8 Mb.

Tali valori si riferisco alle quattro tipologie di vie necessarie alla rappresentazione di una TD-NUCA cache. Le prime quattro vie di dimensione 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi pari a 16, le successive quattro di dimensione 512 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi pari a 8, le successive quattro di dimensione 256 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi pari a 4, e le ultime quattro di dimensione 128 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi pari a 2.

TD – NUCA 4 MB	
2 x 1MB Bank	
Access time (ns)	8,48077
Total power all banks (nj)	165,723
Total power without routing (nj)	117,373
Total routing power (nj)	48,3497
Maximum Bank Power (nJ)	13,0543
2 x 512KB Bank	
Access time (ns)	5,82949
Total power all banks (nj)	75,7005
Total power without routing (nj)	58,7817
Total routing power (nj)	16,9187
Maximum Bank Power (nJ)	10,6073
2 x 256KB Bank	
Access time (ns)	4,90658
Total power all banks (nj)	32,5712
Total power without routing (nj)	28,8425
Total routing power (nj)	3,72875
Maximum Bank Power (nJ)	9,09102
2 x 128KB Bank	
Access time (ns)	4,49215
Total power all banks (nj)	14,435
Total power without routing (nj)	14,435
Total routing power (nj)	0
Maximum Bank Power (nJ)	7,22064

Tabella 6: Parametri di riferimento per una TD-.NUCA da 4 Mb.

Tali valori si riferisco alle quattro tipologie di vie necessarie alla rappresentazione di una TD-NUCA cache. Le prime due vie di dimensione 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 16, le successive due di dimensione 512 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base

di 64 bytes e numero di sottobanchi pari a 8, le successive due di dimensione 256 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 4, e le ultime due di dimensione 128 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 2.

TD – NUCA 2 MB	
1 x 1MB Bank	
Access time (ns)	8,78
Total power all banks (nj)	117,027
Total power without routing (nj)	95,2897
Total routing power (nj)	21,7371
Maximum Bank Power (nJ)	16,3932
1 x 512KB Bank	
Access time (ns)	5,53926
Total power all banks (nj)	51,2527
Total power without routing (nj)	44,6835
Total routing power (nj)	6,56919
Maximum Bank Power (nJ)	14,4727
1 x 256KB Bank	
Access time (ns)	5,13193
Total power all banks (nj)	22,3799
Total power without routing (nj)	22,3799
Total routing power (nj)	0
Maximum Bank Power (nJ)	11,194
1 x 128KB Bank	
Access time (ns)	5,13193
Total power all banks (nj)	11,217
Total power without routing (nj)	11,217
Total routing power (nj)	0
Maximum Bank Power (nJ)	11,217

Tabella 7: Parametri di riferimento per una TD-.NUCA da 2 Mb.

Tali valori si riferisco alle quattro tipologie di vie necessarie alla rappresentazione di una TD-NUCA cache. La prima via di dimensione 1 Mb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi pari a 8, la seconda via di dimensione 512 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 4, la terza via di dimensione 256 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 2, e l'ultima di dimensione 128 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi pari a 1.

TD – NUCA 1 MB	
1 x 512KB Bank	
Access time (ns)	5,53926
Total power all banks (nj)	51,2527
Total power without routing (nj)	44,6835
Total routing power (nj)	6,56919
Maximum Bank Power (nJ)	14,4727
1 x 256KB Bank	
Access time (ns)	5,13193
Total power all banks (nj)	22,3799
Total power without routing (nj)	22,3799
Total routing power (nj)	0
Maximum Bank Power (nJ)	11,194
1 x 128KB Bank	
Access time (ns)	5,13193
Total power all banks (nj)	11,217
Total power without routing (nj)	11,217
Total routing power (nj)	0
Maximum Bank Power (nJ)	11,217
1 x 128KB Bank	
Access time (ns)	5,13193
Total power all banks (nj)	11,217
Total power without routing (nj)	11,217
Total routing power (nj)	0
Maximum Bank Power (nJ)	11,217

Tabella 8: Parametri di riferimento per una TD-.NUCA da 1 Mb.

Tali valori si riferisco alle quattro tipologie di vie necessarie alla rappresentazione di una TD-NUCA cache. La prima via di dimensione 512 Kb, fully associative, con tecnologia di produzione 50 nanometri, dimensione del blocco base 64 bytes e numero di sottobanchi 4, la seconda via di dimensione 256 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi 2, la terza e quarta via di dimensione 128 Kb, fully associative, con tecnologia di produzione a 50 nanometri, dimensione del blocco base di 64 bytes e numero di sottobanchi 1.

Le tabelle sopra riportate sono ordinate secondo lo schema relativo ad una cache TD-NUCA decrescente(ossia le vie più vicine al controller della cache sono le più grandi), per ottenere le tabelle corrispondenti alle TD-NUCA crescenti basta invertire l'ordine di posizionamento delle vie(dalla più piccola alla più grande).

3.3 IL MODELLO SPERIMENTALE

Queste quantità così ottenute risultano però inutilizzabili per la valutazione dei consumi, è stato quindi necessario definire un modello ulteriore per quantificare un consumo associato alle miss ed hit che si avevano sulle caches.
Si sono quindi definite tre nuove quantità atte a determinare il consumo per accesso sulla via:

1) **Bank Power** = Total Power Without Routing / numero banchi via in esame
2) **Max Routing Power** = Maximum Bank Power – Bank Power
3) **Medium Routing Power** = Max Routing Power / numero banchi via in esame / 2

Le quantità sopra elencate stanno ad indicare rispettivamente:

1) Il consumo di un singolo banco appartenente alla via presa in considerazione
2) Il massimo consumo ottenibile per il routing all'interno della via
3) La media del consumo di routing sempre per quella via

Da queste quantità in seguito si ricava il consumo per accesso (Hit Way Power) su quella via, definito come somma del Bank Power e del Medium Routing Power:

Hit Way Power = Bank Power + Medium Routing Power

A questo punto non resta che definire un metodo per il calcolo del consumo dinamico sull'intera cache in funzione delle hit o miss sulla singola via.

3.3.1 Algoritmo per il consumo di potenza dinamico

Per determinare il consumo dell'intera cache bisogna considerare il numero totale di accessi alle singole vie di questa e più precisamente facendo distinzione fra hit ottenute sulle singole vie e miss sull'intera cache.
Questa distinzione nasce dal fatto che la quantità precedentemente stimata: Hit Way Power altro non è che il consumo ad 'accesso' ma nel caso di miss sulla cache, in realtà si sono gia compiuti *n* accessi ai tag delle *n* vie componenti la cache cui non è conseguita una hit, quindi per avere il consumo derivato dalle miss si deve considerare il numero totale di miss avute sulla cache e moltiplicarlo per la somma dei singoli consumi delle *n* vie componenti la cache.
Bisogna però considerare anche l'operazione di rimpiazzamento conseguente alla miss e quindi sommare ai consumi delle singole vie il consumo della prima (poiché è in questa che assumeremo che il dato andrà posizionato); in formula:

$$\textbf{Total Miss Power} =$$

$$\text{Total\_miss\_counter} * (\sum_{i=1}^{n} \text{Hit\_way\_power}[i] + \text{reload})$$

$$\text{reload} = \text{Hit Way Power } [1]$$

Per quanto concerne invece il calcolo del consumo derivato dalle hit, esso dovrà essere stimato per ogni via della cache; più in dettaglio si dovrà moltiplicare il numero totale di hit su quella via per la somma delle Hit Way Power delle vie precedenti (inclusa quella in esame) più il consumo dovuto alla promozione del dato oggetto della hit ed ancora più lo spostamento del dato che si trovava precedentemente nella locazione soggetta a promozione.
Per la tipologia di rimpiazzamento presa in esame (cioè ad una via) il consumo dovuto alla promozione è pari al consumo della via precedente a quella in cui si è avuta la hit, mentre quest'ultimo sarà il consumo da attribuire al rimpiazzamento.
In pratica per la via n-esima il consumo sarà così quantificabile:

$$\textbf{HitPower[n]} = \text{hit\_counter}[n] * \left(\sum_{i=1}^{n} \text{Hit\_way\_power}[i] + \text{promozione} + \text{rimpiazzamento}\right)$$

promozione = Hit Way Power[n-1] rimpiazzamento = Hit Way Power [n]

Infine per avere il cosumo totale basta fare:

$$\textbf{Total Hit Power} = \sum_{i=1}^{n} \text{Hit power}[i]$$

Capitolo 4

RISULTATI

In questa sezione verranno esposti i risultati ottenuti dalle simulazioni effettuate sulla base dei modelli definiti nel capitolo precedente.

4.1 COMPUTO DELLE HIT WAY POWER

Come gia detto per calcolare i consumi delle singole vie componenti le varie tipologie di caches prese in esame, è stato utilizzato CACTI 3.2, adattando poi i valori ottenuti al modello architetturale.
Nella tabella sottostante sono rappresentati i valori di consumo espressi in nanojoule per le caches a 16 vie cioè le D-NUCA da 16 MB e le TD-NUCA da 8 MB:

	D-NUCA	TD-NUCA DEC	TD-NUCA CRE
	16MB	8MB	8MB
Hit_Way_Power_1	7,514515234	7,514515234	7,218285
Hit_Way_Power_2	7,514515234	7,514515234	7,218285
Hit_Way_Power_3	7,514515234	7,514515234	7,218285
Hit_Way_Power_4	7,514515234	7,514515234	7,218285
Hit_Way_Power_5	7,514515234	7,551436719	7,445674375
Hit_Way_Power_6	7,514515234	7,551436719	7,445674375
Hit_Way_Power_7	7,514515234	7,551436719	7,445674375
Hit_Way_Power_8	7,514515234	7,551436719	7,445674375

Hit_Way_Power_9	7,514515234	7,445674375	7,551436719
Hit_Way_Power_10	7,514515234	7,445674375	7,551436719
Hit_Way_Power_11	7,514515234	7,445674375	7,551436719
Hit_Way_Power_12	7,514515234	7,445674375	7,551436719
Hit_Way_Power_13	7,514515234	7,218285	7,514515234
Hit_Way_Power_14	7,514515234	7,218285	7,514515234
Hit_Way_Power_15	7,514515234	7,218285	7,514515234
Hit_Way_Power_16	7,514515234	7,218285	7,514515234

Tabella 9: Valori Hit Way Power.

Tali valori si riferisco al consumo ad accesso da prendere in considerazione per D-NUCA e TD-NUCA (sia in configurazione crescente che decrescente) a 16 vie .

Come si può notare (e come era prevedibile del resto), l'andamento del consumo risulta costante sulle 16 vie per la cache D-NUCA, mentre assume un andamento crescente per le TD-NUCA crescenti con picco massimo nelle vie 9,10,11,12 causato dalla particolare geometria di queste ultime che porta ad un consumo per singolo banco superiore a quello delle vie successive.

Discorso analogo vale per le TD-NUCA decrescenti, ma data la geometria invertita va fatto per le vie 5,6,7,8.

Di seguito sono esposti i consumi per le cache da 8 vie, quindi D-NUCA 8 Mb, TD-NUCA 4Mb:

	D-NUCA	TD-NUCA DEC	TD-NUCA CRE
	8MB	4MB	4MB
Hit_Way_Power_1	7,514515234	7,514515234	7,218285
Hit_Way_Power_2	7,514515234	7,514515234	7,218285
Hit_Way_Power_3	7,514515234	7,551436719	7,445674375
Hit_Way_Power_4	7,514515234	7,551436719	7,445674375
Hit_Way_Power_5	7,514515234	7,445674375	7,551436719
Hit_Way_Power_6	7,514515234	7,445674375	7,551436719
Hit_Way_Power_7	7,514515234	7,218285	7,514515234
Hit_Way_Power_8	7,514515234	7,218285	7,514515234

Tabella 10: Valori Hit Way Power.

Tali valori si riferisco al consumo ad accesso da prendere in considerazione per D-NUCA e TD-NUCA (sia in configurazione crescente che decrescente) a 8 vie .

Anche per queste come nel caso precedente si possono notare i picchi delle TD-NUCA nelle vie 3 e 4 per le Decrescenti e 5 e 6 per le Crescenti.
Per ambedue le configurazioni sopraesposte (16 vie ed 8 vie) si può constatare che i consumi si attestano sugli stessi valori, data l'identica configurazione architetturale delle vie, l'unica differenza è data dal numero di queste.

Per quanto riguarda le cache a 4 vie i consumi raggiungono valori più elevati sia per il primo gruppo in esame e cioè D-NUCA 4Mb TD-NUCA 2Mb Crescenti e TD-NUCA 2Mb Decrescenti, che per il secondo D-NUCA 2Mb TD-NUCA 1Mb Crescenti e TD-NUCA 1Mb.

	D-NUCA	TD-NUCA DEC	TD-NUCA CRE
	4MB	2MB	2MB
Hit_Way_Power_1	12,19133672	12,19133672	11,217
Hit_Way_Power_2	12,19133672	11,58360313	11,1909625
Hit_Way_Power_3	12,19133672	11,1909625	11,58360313
Hit_Way_Power_4	12,19133672	11,217	12,19133672

Tabella 11: Valori Hit Way Power.

Tali valori si riferisco al consumo ad accesso da prendere in considerazione per D-NUCA e TD-NUCA (sia in configurazione crescente che decrescente) a 4 vie; in particolare per i tagli da 4 e 2 Mb.

In questo caso non si ha il picco che si era verificato nelle precedenti architetture TD-NUCA, frutto di un consumo per singolo banco più elevato rispetto alle rispettive vie D-NUCA, si può notare però come la via 3 per le TD-NUCA Decrescenti (2 per le Crescenti) consumi meno della via 4 (1 per le Crescenti) anche se di minore dimensione, anch'essa frutto della particolare architettura.

Stesso discorso fatto per l'architettura precedente vale infine per l'ultimo taglio preso in esame:

	D-NUCA	TD-NUCA DEC	TD-NUCA CRE
	2MB	1MB	1MB
Hit_Way_Power_1	11,58360313	11,58360313	11,217
Hit_Way_Power_2	11,58360313	11,1909625	11,217
Hit_Way_Power_3	11,58360313	11,217	11,1909625
Hit_Way_Power_4	11,58360313	11,217	11,58360313

Tabella 12: Valori Hit Way Power.

Tali valori si riferisco al consumo ad accesso da prendere in considerazione per D-NUCA e TD-NUCA (sia in configurazione crescente che decrescente) a 4 vie; in particolare per i tagli da 2 e 1 Mb.

4.2 COMPUTO DEI CONSUMI

A questo punto sono stati presi i valori di consumo delle Hit Way Power esposti nel paragrafo precedente e sono stati adottati come parametri in ingresso nel simulatore Sim-Alpha 1.0, gia precedentemente modificato nel lavoro [10] per implementare le politiche necessarie alla simulazione delle TD-NUCA ed ora aggiornato ulteriormente per il computo delle potenza dissipata durante il ciclo di simulazione con i vari benchmark.

Questo simulatore è un execution-driven ed è basato su una coda di eventi per la comunicazione fra i moduli. In particolare, la versione che abbiamo scelto, la Sim-Alpha 1.0, è basata sull'architettura Alpha 21264 ed è stata validata rispetto a questa da un recente lavoro svolto presso l'Università del Texas di Austin [1]. In più, la versione del simulatore che ci ha permesso di effettuare la presente ricerca integra il supporto per le caches D-NUCA ed è stata messa a punto in una ulteriore ricerca svolta sempre presso l'Università del Texas di Austin ([2] e [6]). I lavori [2] e [6], come descritto in precedenza, mostrano la nuova architettura di caches D-NUCA ed appoggiandosi al simulatore appena citato ne misurano le prestazioni.

Oltre a dover decidere la piattaforma di simulazione si sono dovuti trovare anche i giusti benchmarks su cui basare le prove. Le applicazioni di test scelte appartengono alla suite SpecInt 2000 ed in particolare sono stati usati i seguenti: *176.gcc*, *181.mcf*, *256.bzip2*, *300.twolf*. Questi 4 benchmarks sono stati scelti, selezionandoli dall'intero gruppo di Spec, a causa della loro tendenza a generare più miss in L1 così da stressare maggiormente la cache di livello successivo.

Inoltre, per massimizzare ulteriormente gli accessi alla cache L2, ogni prova di simulazione è stata svolta con un proprio FFWD (fast forwarding, ovvero avanzamento rapido della simulazione), in modo da saltare le prime istruzioni che causano pochi accessi in memoria e simulandone solo un certo numero RUN calcolato attraverso ripetute prove degli Spec stessi. La seguente tabella riporta i numeri di FFWD e RUN per ognuno degli Spec utilizzati, ed in più riporta nella terza colonna il numero degli accessi alla L2 per ogni milione di istruzioni.

SPECINT2000	Phase		L2 load acc/ Million instr.
	FFWD	RUN	
176.gcc	2,367B	300M	25.900
181.mcf	5B	200M	260.620
256.bzip2	744B	1B	9.300
300.twolf	511B	200M	22.500

Tabella 13: FFWD e RUN per i 4 SpecInt 2000 utilizzati.
Il FFWD è l'avanzamento rapido del test. Il RUN è il numero di istruzioni simulate.

Gli Spec sono eseguibili compilati ovviamente per l'architettura Alpha 21264 ed ognuno ha i propri input (input REF).
La configurazione dell'architettura presa in considerazione nelle seguenti simulazioni prevede le impostazioni standard del simulatore Simple Scalar, le principali configurazioni standard dell'architettura Alpha 21264 sono riportate di seguito

Parametro	Valore
Frequenza CPU	500.000.000 Hz
Dimensione della coda di Instruction Fetch	4 istruzioni
Dimensione della coda di commit	11 istruzioni
Dimensione del Reorder Buffer	80 entrate
Numero di Integer ALU	4
Numero di FP ALU	1
Branch predictor	Standard 21264
Intruction TLB e Data TLB	4KB, 128 vie, blocco a 32B, 1 ciclo di latenza
Instruction L1 Cache	64KB, 2 vie, blocco a 64B, 1 ciclo di latenza
Data L1 Cache	64KB, 2 vie, blocco a 64B, 3 cicli di latenza
Dimensione del MSHR (Miss Status Holder Register)	8 entrate per cache
Bus interno al processore	128bit, 1 ciclo di latenza
Bus della memoria (esterno)	128bit, 4 cicli di latenza
Memoria	SDRAM, 1 ciclo di Ras, 1 ciclo di Cas, Single Data Rate, moltiplicatore a 6

Tabella 14: principali parametri di configurazione dell'architettura Alpha 21264 usata nei tests .

In particolare, la gerarchia delle memorie è composta da un TLB dati e da un TLB istruzioni entrambi a 128 vie e di ampiezza 4KB; è composta da una L1 dati e da una L1 istruzioni ciascuna da 64KB, a 2 vie, con blocchi da 64B, 3 cicli di accesso per la parte dati, e a singolo ciclo di accesso per la parte istruzioni; è composta da un bus verso la memoria da 16B di ampiezza e 4 cicli di latenza; è composta infine da una memoria in tecnologia SDRAM con una latenza di 48 cicli per un singolo trasferimento.
Relativamente alla cache L2 la sua definizione avviene inserendo nel file di configurazione una o più stringhe.

Il simulatore TD-NUCA Sim-Alpha implementa parte delle tecniche riguardanti le caches TD-NUCA definite precedentemente ed ognuna di esse è impostabile come è mostrato di seguito.

La struttura dei banchi è definita dal parametro:

–nuca:t_banks_number 1 1 2 4

In questo esempio la cache è triangolare crescente con 1, 1, 2 e 4 banchi per ogni via. Notare che per attivare la modalità “triangolare” per le caches L2, ovvero per definire la presente opzione, deve essere disattivata la modalità “matriciale” impostando 1 come numero di righe ed impostando la somma totale del numero di banchi come numero di colonne (in questo esempio 8).

Attraverso i seguenti parametri:

–nuca:cachebanks_latencies 3 3 3 3

–nuca:interleaved true

–nuca:random_insertion false

–nuca:force_promotion false

–nuca:prom_banks_n 1

-nuca:way_power 11.217 11.217 11.190 11.583

sono impostate rispettivamente le latenze dei banchi delle quattro vie (nell'esempio ogni via ha banchi con tempi di risposta 3), la modalità di mapping verticale (interlacciata o meno), la tecnica per inserire i blocchi in modo random, la tecnica per forzare la promozione in caso di incompatibilità fra blocco da promuovere e blocco da retrocedere, il numero di vie da saltare in ogni promozione, ed i consumi di ogni via.

4.3 RISULTATI DELLE SIMULAZIONI

Di seguito sono riportati i grafici relativi alle simulazioni prese in esame; abbiamo messo in evidenza l' IPC (istruzioni per ciclo), il miss rate, ed il consumo dinamico totale:

GCC

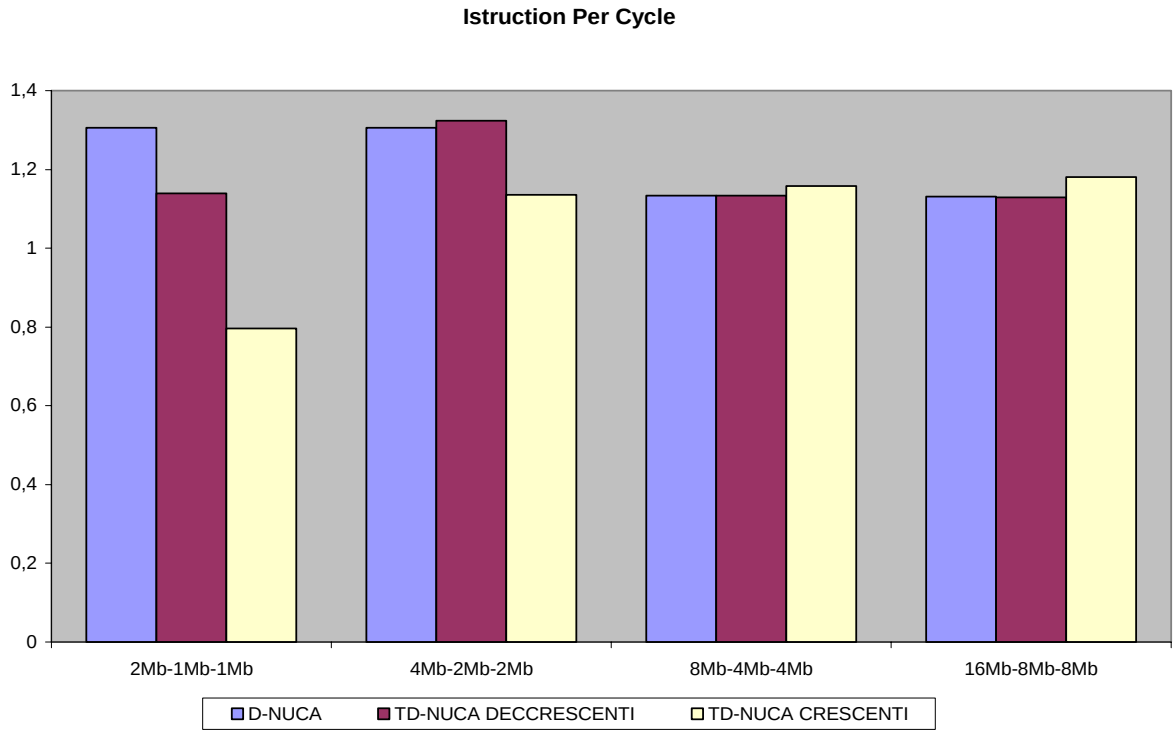

Grafico 28: IPC relative al Bebchmark 176.gcc a confronto.

In questo Benchmark si può notare come la TD-Nuca Decrescente regga bene il confronto con la D-Nuca di dimensione doppia, mentre la TD-Nuca Crescente, per dimensioni maggiori rende anche meglio delle altre due, situazione opposta si ha nei tagli più piccoli.

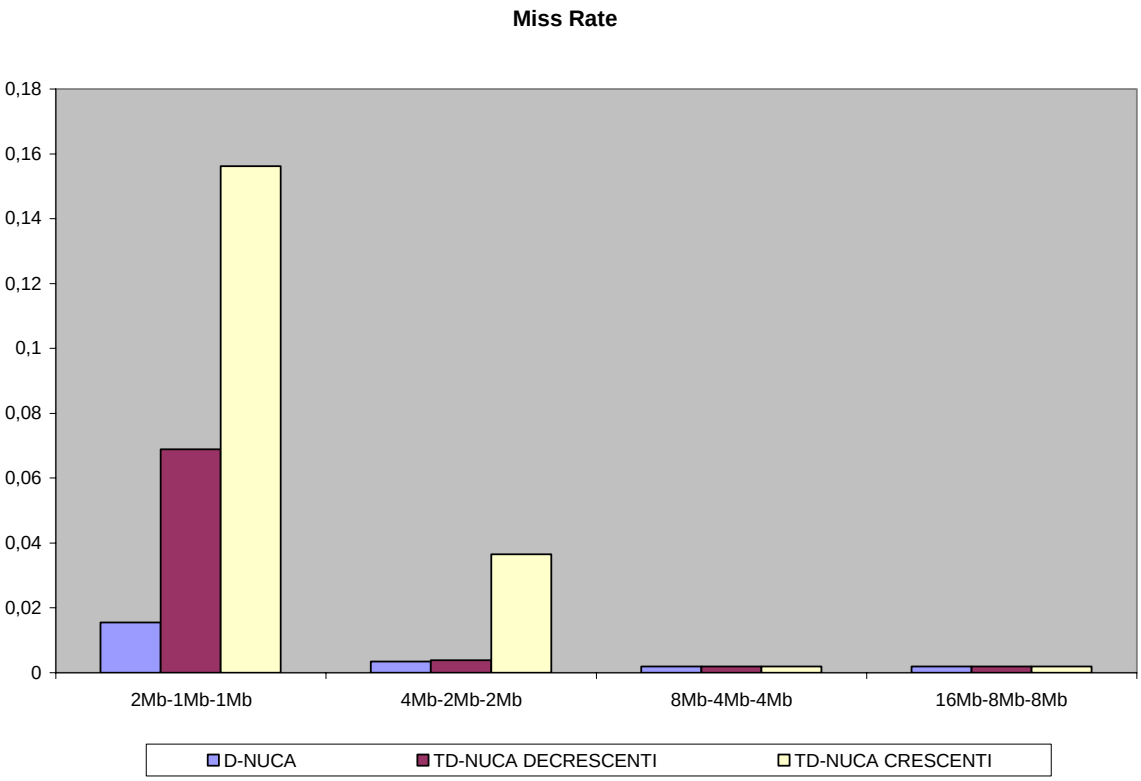

Grafico 29: Miss Rate relative al Bebchmark 176.gcc a confronto.

Come era prevedibile le TD-Nuca Crescenti presentano un tasso di miss molto più elevato rispetto alle concorrenti per le dimensioni più piccole, differenze che si appianano subito dopo aver superato il taglio da 2Mb.

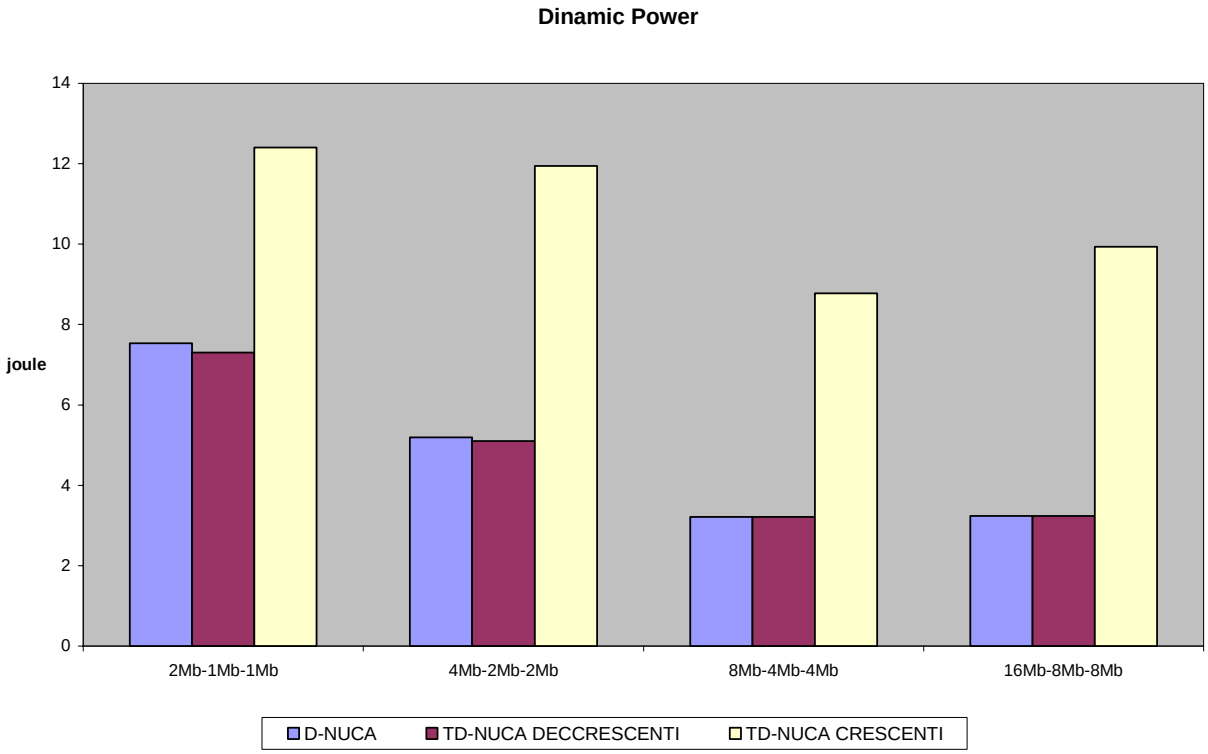

Grafico 30: Dinamic Power relativa al Bebchmark 176.gcc a confronto.

Il consumo finale per quanto riguarda le TD-Nuca Crescenti si attesta sempre su valori più elevati delle altre 2 configurazioni; mentre per quanto riguarda le TD-Nuca Decrescenti si mantengono costantemente su valori leggermente inferiori alle D-Nuca.

MCF

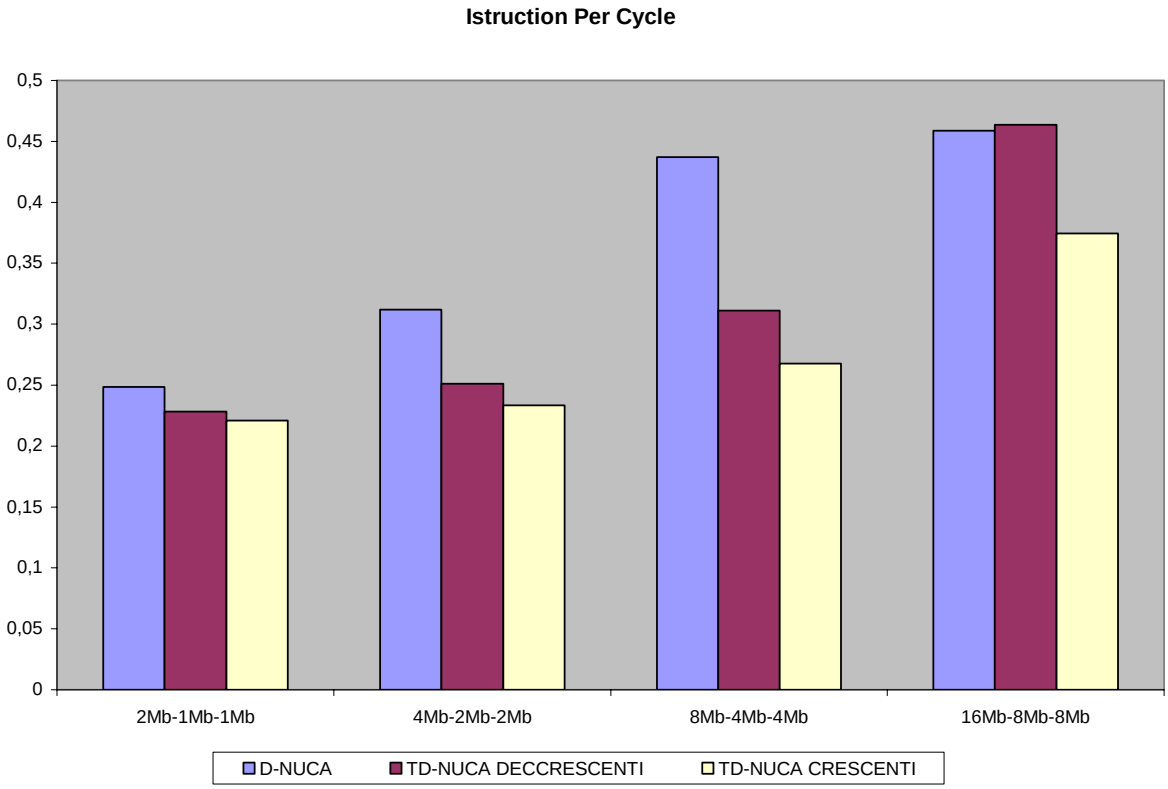

Grafico 31: IPC relative al Bebchmark 181.mcf a confronto.

In questo caso le D-Nuca si mantengono al di sopra delle concorrenti, fatta eccezione nel caso del taglio più grande, dove il primato ,seppur di poco, è delle TD-Nuca Decrescenti.

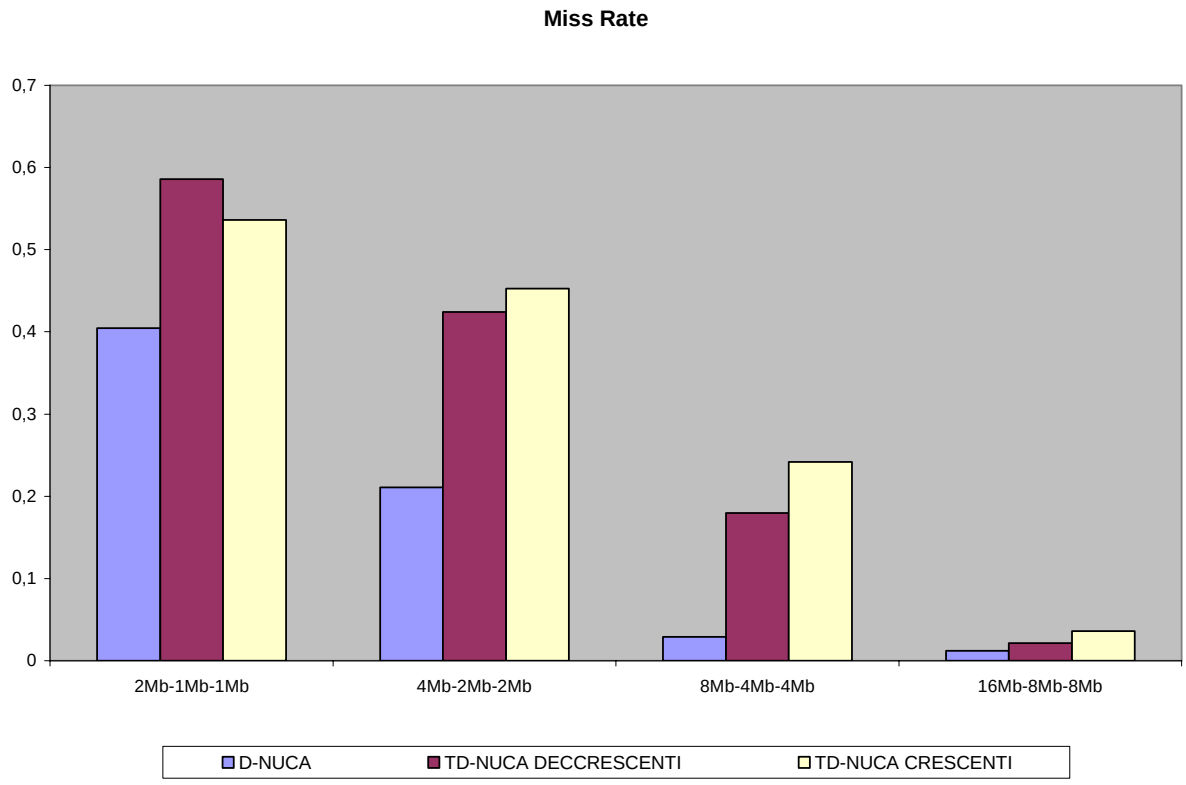

Grafico 32: Miss Rate relative al Bebchmark 181.mcf a confronto.

Anche in questo caso le D-Nuca si attestano su valori di Miss Rate più bassi, che si appianano però sui tagli più grandi.

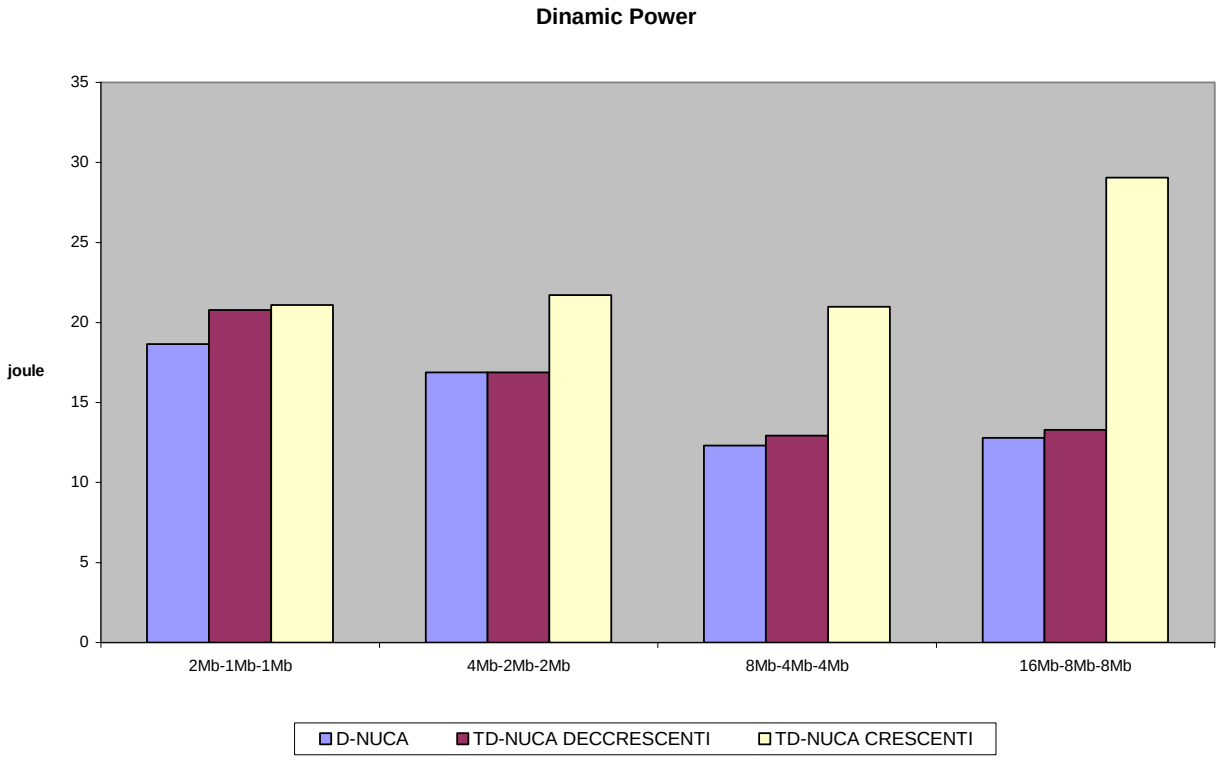

Grafico 33: Dinamic Power relativa al Bebchmark 181.mcf a confronto.

Per quanto riguarda i consumi infine, le TD-Nuca Crescenti si attestano in ogni caso su valori superiori alle altre, e nel caso del taglio più grande questo valore è addirittura più del doppio delle altre, che invece si comportano in modo simile.

BZIP

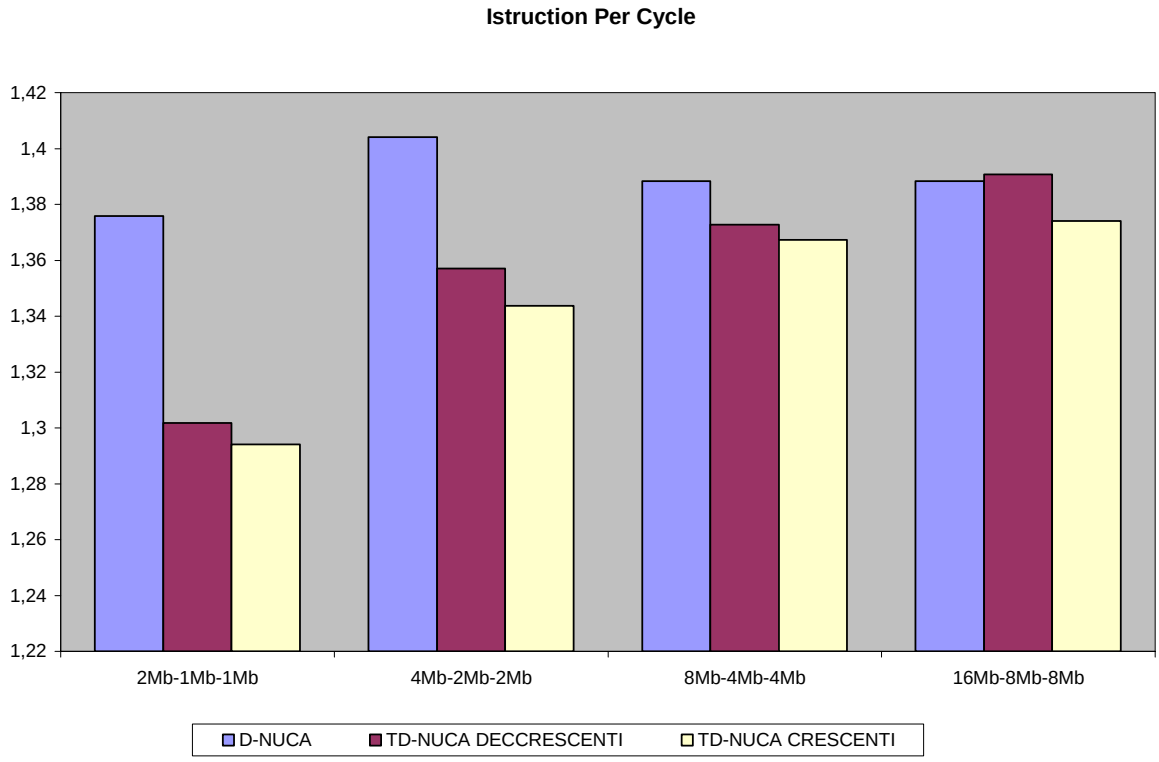

Grafico 34: IPC relative al Bebchmark 256.bzip2 a confronto.

Anche per questo benchmark come per il precedente, le D-Nuca si comportano meglio delle altre fino al taglio più grande dove vengono superate dalle TD-Nuca Decrescenti.

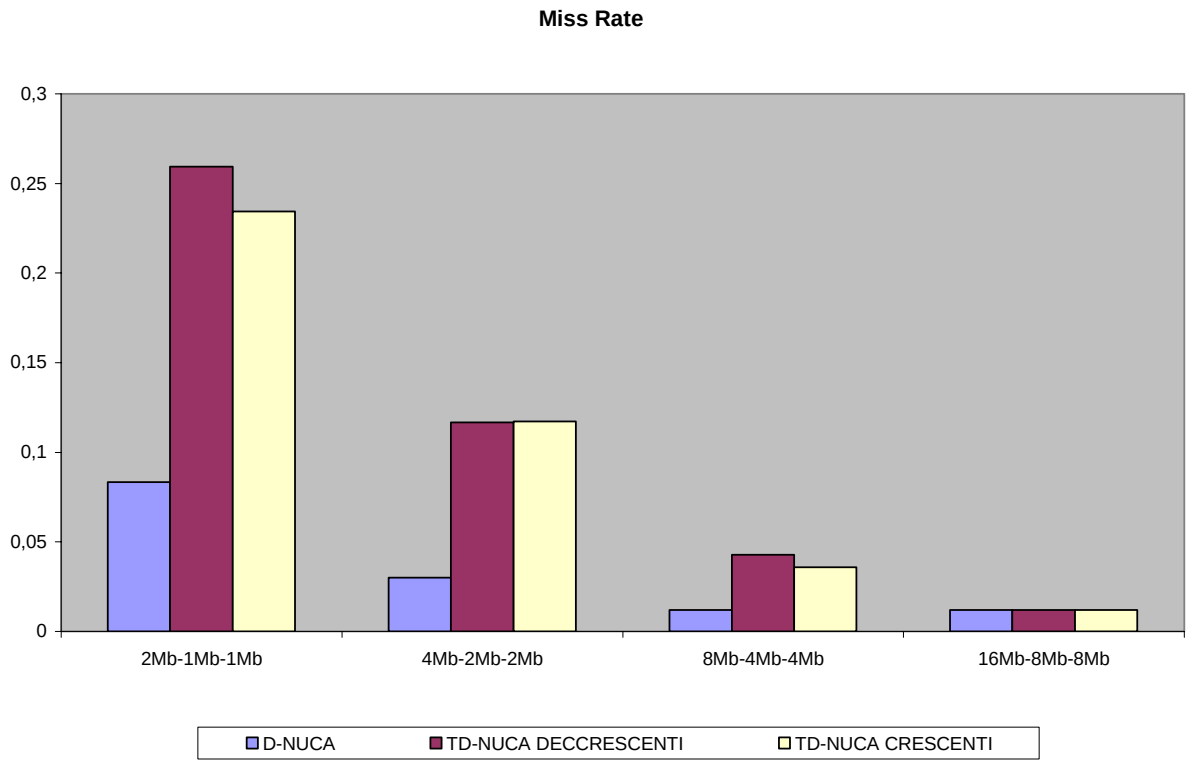

Grafico 35: Miss Rate relative al Bebchmark 256.bzip2 a confronto.

Come prevedibile il miss rate delle D-Nuca risulta più basso delle altre fino al taglio più grande, dove il gap viene annullato.

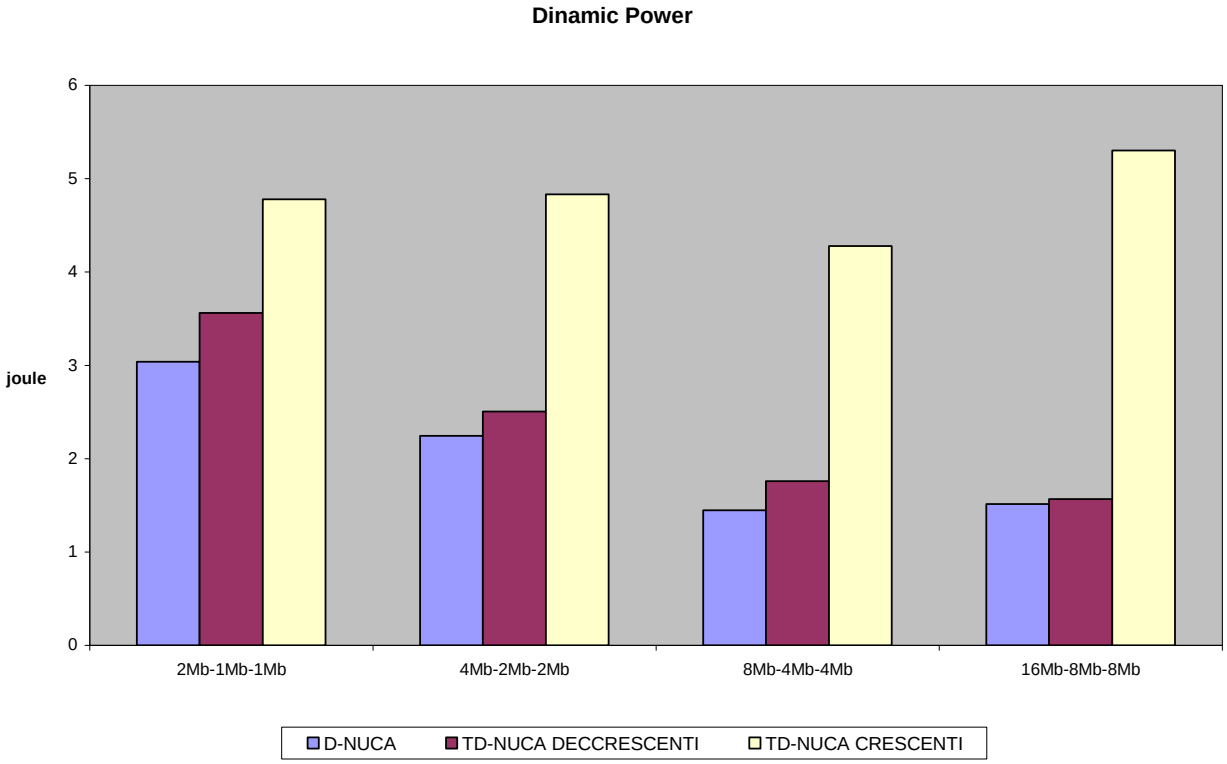

Grafico 36: Dinamic Power relativa al Bebchmark 256.bzip2 a confronto.

I consumi, anche in questo caso, come in precedenza, sono più alti per le TD-Nuca Crescenti ed oscillano attorno a valori simili per le altre due configurazioni.

TWOLF

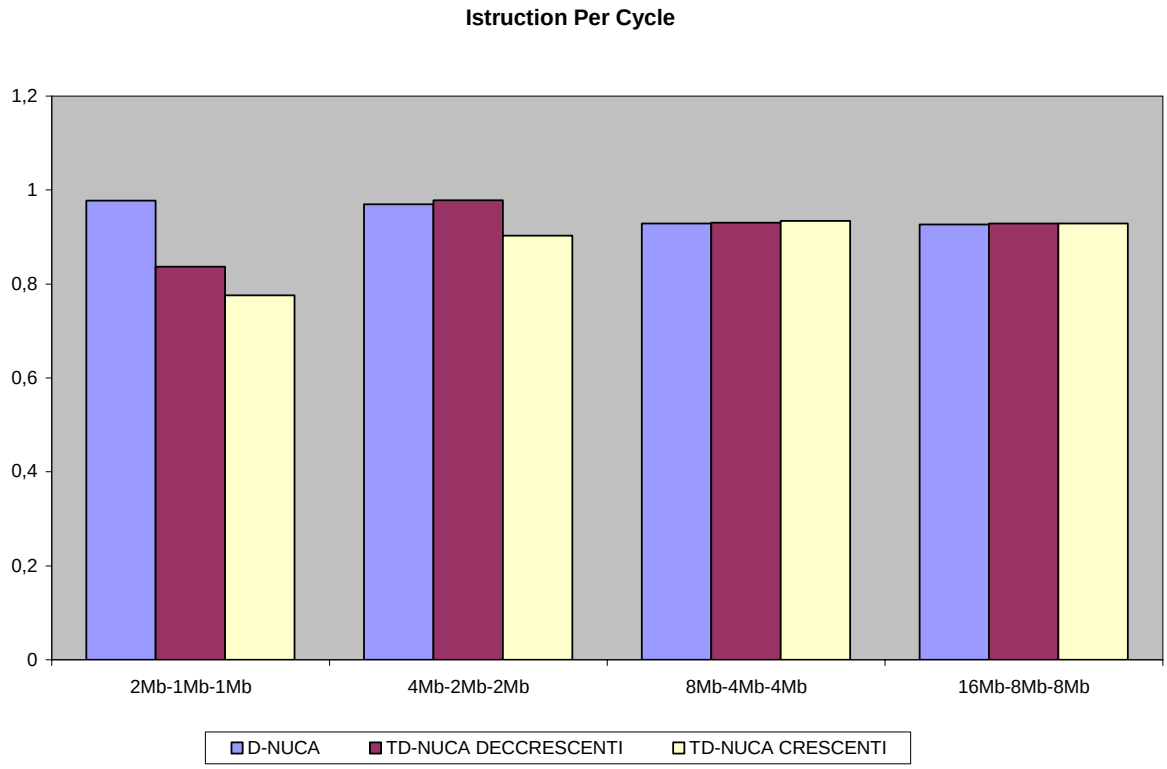

Grafico 37: IPC relative al Bebchmark 300.twolf a confronto.

Per questo ultimo benchmark l'IPC si attesta su valori pressoché identici nei tagli superiori, nei tagli più piccoli c'è l'alternanza del miglior risultato fra D-Nuca e TD-Nuca Decrescenti.

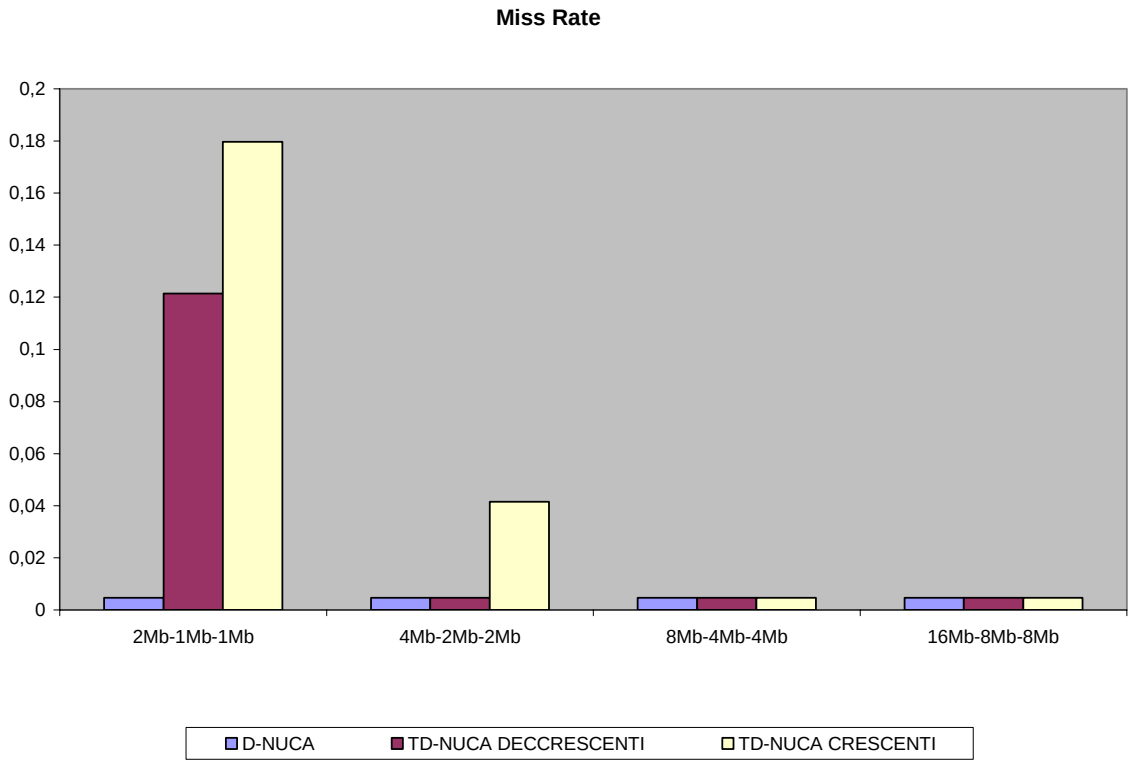

Grafico 38: Miss Rate relative al Bebchmark 300.twolf a confronto.

Dal grafico di questo benchmark si evince chiaramente che una cache TD-NUCA da 1 Mb non può reggere il confronto con l'equivalente D-NUCA da 2Mb.

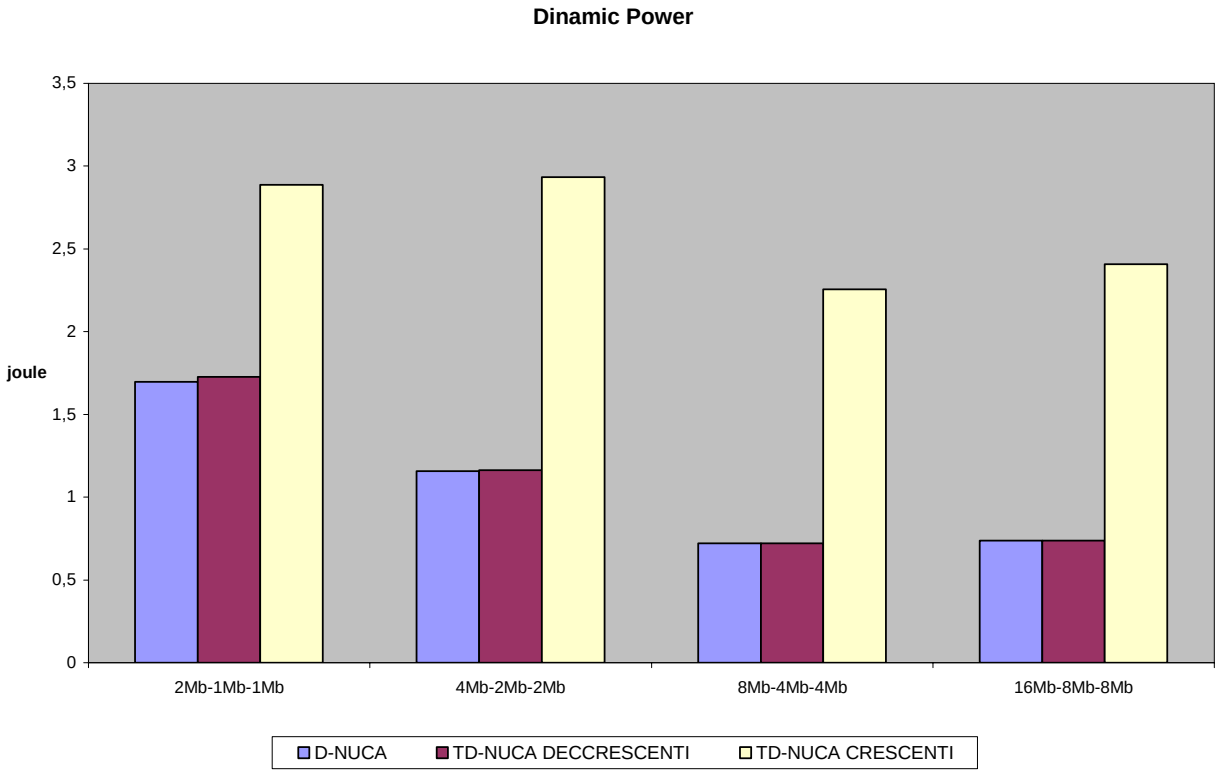

Grafico 39: Dinamic Power relativa al Bebchmark 300.twolf a confronto.

Come al solito la TD-NUCA Crescente tende a consumare più delle altre due, che invece si comportano allo stesso modo.

Capitolo 5

CONCLUSIONI

I benchmark selezionati dalla suite SpecInt 2000 per le simulazioni sono: 176.*gcc*, *181.mcf*, *256.bzip2*, *300.twolf*.
Sono state confrontate caches TD-NUCA (Crescenti e Decrescenti) con caches D-NUCA di dimensioni doppie.
I dati che emergono dalle simulazioni permettono di affermare che:

- l'utilizzo di cache D-NUCA di forma triangolare Crescente non è affatto conveniente, sia in termini di consumi che di prestazioni in generale, neanche ponendo a confronto cache D-NUCA con TD-NUCA di pari dimensioni

- le TD-NUCA Decrescenti in ogni benchmark si sono fermate su valori simili a quelli delle D-NUCA; in tre dei quattro si sono attestate su valori in assoluto migliori, non solo in termini di consumo ma anche di prestazioni generali

Va inoltre sottolineato il fatto che, avendo confrontato D-NUCA di dimensioni doppie rispetto alle TD-NUCA, le prime avranno una maggiore occupazione di superficie di silicio (circa il doppio), quindi ci sarà un maggiore consumo di potenza statico.
Ancora, poiché in questo studio è stata testata esclusivamente la politica di promozione e rimpiazzamento ad una via, è deducibile che esistano altre possibili combinazioni di algoritmi di mapping che possano ottimizzare la gestione di questa famiglia di memorie caches.
Analizzando l'attuale scenario nel mondo dei microprocessori, posso in tutta tranquillità affermare che ulteriori studi atti ad affinare questa tecnica progettuale per le cache sarebbero da prendere in considerazione.

RIFERIMENTI

[1] R. Desikan, D. Burger, S. Keckler, T. Austin, “Sim-Alpha: a Validated, Execution-Driven Alpha 21264 Simulator”. Technical Report TR-01-23, Department of Computer Sciences, University of Texas at Austin, 2001.

[2] C. Kim, D. Burger, S. Keckler, “An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches” *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 211-222, October, 2002..

[3] International Technology Roadmap for Semiconductor, 2001.

[4] J. L. Cruz, A. Gonzàlez, M. Valero, N. Topham, “Multiple-Banked Register File Architectures”, Proc. of 27th. Ann. Int. Symposium on Computer Architecture (ISCA 2000)
Vancouver (Canada), June 12-14, 2000, pages 316-325.

[5] A. Sakanaka, T. Sato, “A Leakage-Energy-Reduction Technique for High-Associativity Caches in Embedded Systems”, *Workshop on Memory Access Decoupled Architectures and Related Issues* (MEDEA) held in conjunction with *12th International Conference on Parallel Architectures and Compilation Techniques* (PACT), pp.51-56, September 2003.

[6] C. Kim, D. Burger, S. Keckler, “An Adaptive Cache Structure for Future High-Performance Systems”, *IBM Austin Center for Advanced Studies Workshop*, February, 2002.

[7] D. Burger, T. Austin, “The Simple Scalar Tool Set, Version 2.0” Technical Report 1342, University of Wisconsin-Madison, CS Department, June 1997**.**

[8] Compaq Computer Corporation, “21264/EV68CB and 21264/EV68DC Hardware Reference Manual”, pp 39-40 and 87-94, june 2001.

[9] Compaq Computer Corporation, “Alpha 21264 Microprocessor Hardware Reference Manual”.

[10] Simone Grechi, “ Progettazione logica e definizione di protocolli e politiche per le caches D-Nuca Triangolari”, tesi di laurea in Ingegneria Informatica Università di Pisa anno accademico 2002/2003.

[11] "SIA Annual Report 2003".

[12] Premkishore Shivakumar, Norman P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model", *Western Research Lab (WRL) Research Report 2001/2.*

[13] C. Cowell, C. A. Moritz, W. Burlesson, "Improved Modelling and Data Migration for Dynamic Non-Uniform Cache Access", In WDD2 2003 organized in conjunction with the International Symposium on Computer Architecture, ISCA 2003.

[14] R. Desikan, D. Burger, S. Keckler, "Measuring Experimental Error in Microprocessor Simulation", *28th International Symposium on Computer Architecture* (ISCA), pp. 266-277, June 2001.

[15] J. M. Parcerisa, A. Gonzàlez, "Reducing Wire Delay Penalty through Value Prediction", Proc. of the 33rd International Symposium on Microarchitecture (MICRO-33)
Monterey, California,United States, December 2000, pp. 317-326.

[16] H. Hanson, S. W. Keckler, D. Burger, "Static Energy Reduction Techniques in Microprocessor Caches" TR-01-18, Department of Computer Sciences, The University of Texas at Austin, June, 2001.

[17] H. Hanson, M.S. Hrishikesh, V. Agarwal, S. W. Keckler, D. Burger, "Static Energy Reduction Techniques for Microprocessor Caches" *International Conference on Computer Design* (ICCD), pp. 276-283, September, 2001.

[18] D. Burger, A. Kagi, M.S. Hrishikesh, "Memory Hierarchy Extension to the SimpleScalar Tool Set" The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-99-25.