Arquitectura de Computadores

Notas de Clase

Elisardo Antelo Suárez

Tema 1: Procesadores Multinúcleo e Perspectivas de Escalamento

1- Introdución

Neste tema abordamos unha primeira aproximación aos microprocesadores multinúcleo, estudando a súa estrutura xeral, o impacto das tecnoloxías de fabricación, e un modelo simple para entender as prestacións que se poden acadar. No estudo do contexto actual da tecnoloxía de microprocesadores, vamos a explicar primeiro por qué o modelo seguido até fai poucos anos na industria de microprocesadores non era escalábel para seguir obtendo un incremento de prestacións ao mesmo ritmo que se viña facendo.

Os núcleos de procesamento convencionais baséanse nun modelo de programación secuencial (os programas son secuencias ordenadas de instrucións producidas polo compilador a partires dun programa escrito nunha linguaxe de alto nivel) executadas nunha arquitectura superescalar capaz de executar (fora de orde) un certo número de instrucións en paralelo por ciclo (ver Figura 1). Deste xeito é necesario extraer o paralelismo a nivel de instrución mediante técnicas hardware (dinámico) ou mediante o compilador (estático), ou mesturar estratexias estáticas e dinámicas xunto coa execución especulativa na presenza de saltos. Todas estas técnicas deben posibilitar a execución en paralelo de instrucións non dependentes e predicir os saltos no control de fluxo cunha elevada probabilidade de éxito. Os saltos no control de fluxo do programa e as dependencias entre instrucións constitúen as maiores limitacións para acadar unha elevada taxa de instrucións executadas por ciclo.

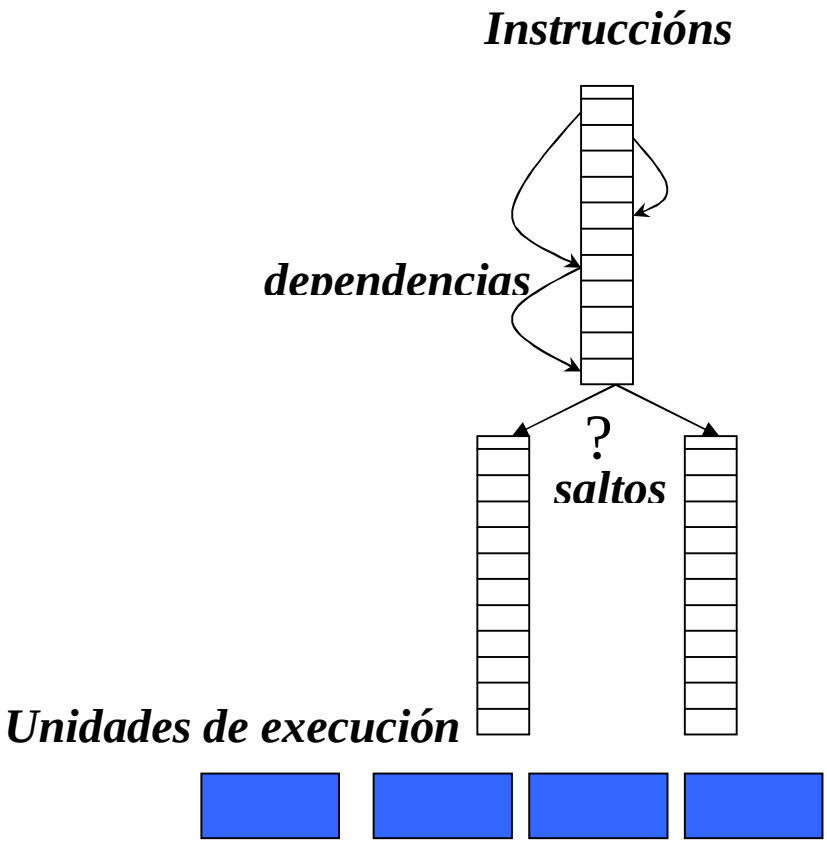


Figura 1: Modelo de execución secuencial.

Na Figura 2 amósase a arquitectura dun núcleo típico que podemos atopar nos microprocesadores actuais (o núcleo do exemplo é da familia de arquitecturas Sandy Bridge de Intel). A Figura 3 presenta a arquitectura dun microprocesador multinúcleo. De xeito breve podemos dicir que a arquitectura consta de 3 niveis de cache. As memorias caches son circuítos de almacenamento dentro do mesmo chip que as unidades de procesamento. Almacenan o código (instrucións) e datos que están sendo utilizados polo procesador (cando se fai unha instrucción ñLoadò ou ñStoreò, cárgase na memoria cache a palabra referenciada, se non o está xa, xunto coas palabras veciñas en memoria, o que se coñece como liña cache; polo tanto é un mecanismo automático de precarga de datos no chip). Deste xeito, permiten un acceso rápido as instrucións e datos para a súa execución. A memoria principal, externa ao chip do procesador, ten na actualidade uns tempos de acceso da orden de centos de ciclos de reloxo do procesador. Por contra os tempos de acceso para as caches internas son só duns poucos ciclos (1-30 ciclos).

Por outra banda, existen diferentes niveis de cache dentro do procesador. Os diferentes niveis permiten un acceso rápido (baixa latencia e alto ancho de banda) aos datos e instrucións por parte do procesador, e ao mesmo tempo permite ter unha elevada capacidade de memoria dentro do chip.

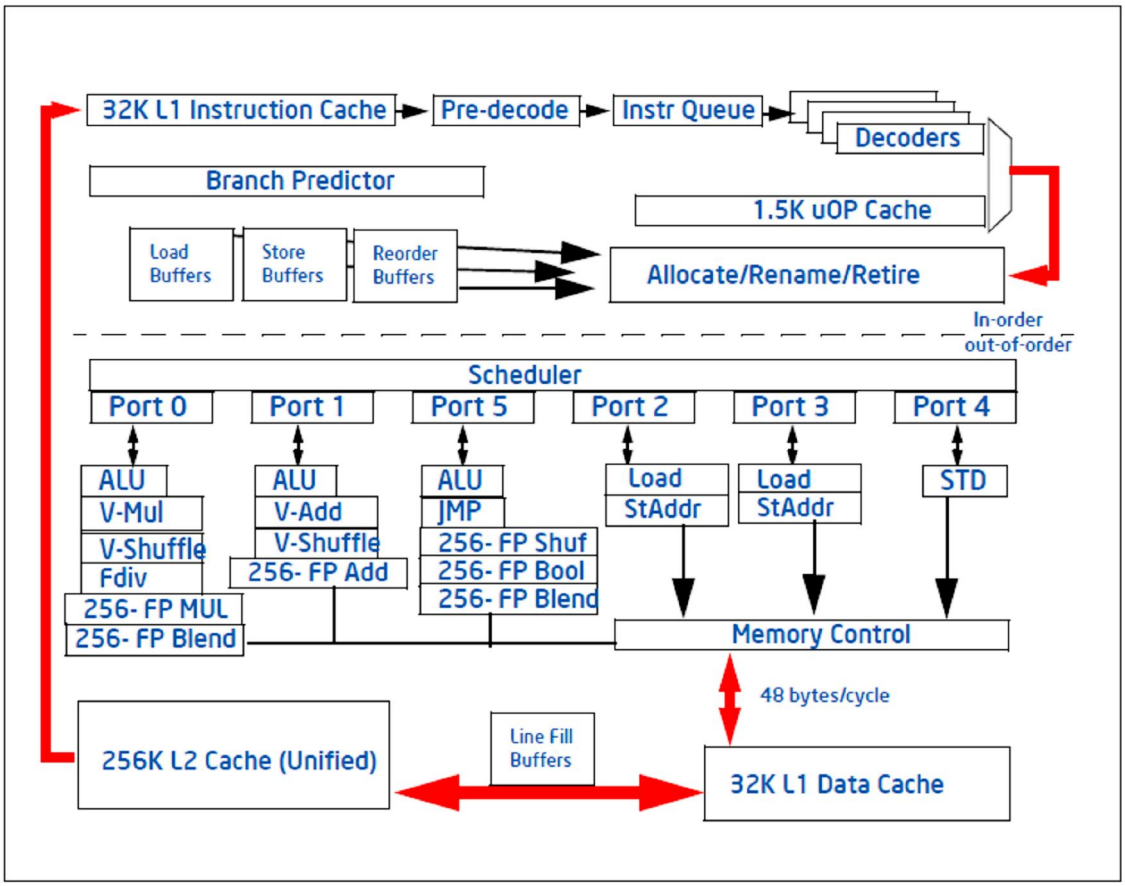


Figura 2: Arquitectura dun núcleo.

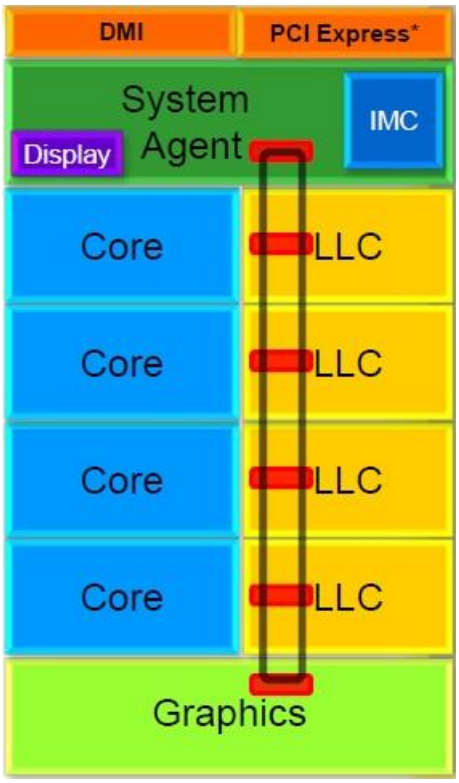


Figura 3: Arquitectura dun microprocesador multinúcleo.

O primeiro nivel, usualmente corresponde en realidade a dúas caches, unha para datos e outra para instrucións, cada unha con capacidade típica de 32KB e latencia de acceso mínimo de uns 4 ciclos por liña cache (con un valor típico de 64 bytes). Este nivel de cache é o que está conectado directamente ao procesador para executar as instrucións de lectura e escritura de datos (ñLoad/Storeò), e a lectura das instrucións, e polo tanto está optimizado para baixa latencia e alto ancho de banda. No exemplo da Figura 1, a cache de instrucións permite ler 16 bytes de instrucións por ciclo. No caso da cache de datos, permítense facer concorrentemente até dúas operación de lectura e unha de escritura por ciclo, transferindo 16 bytes en cada unha delas (taxa de transferencia de lectura de 32 bytes por ciclo, e de escritura de 16 bytes por ciclo, para un ancho de banda total de 48 bytes por ciclo). O segundo nivel de cache, corresponde a unha cache unificada (instrucións e datos), con valores típicos de 256KB cunha latencia mínima de 12 ciclos e un ancho de banda de 32 bytes/ciclo.

Os dous primeiros niveis de cache so privados para cada núcleo. O terceiro nivel de cache (LLC na Figura 3, por Last Level Cache) é compartido entre todos os núcleos e usualmente está divido en varios bancos para soportar múltiples accesos concorrentes por parte dos diferentes núcleos. Na Figura 3 observamos como unha interconexión interna con topoloxía de anel conecta os bancos das cache de último nivel cos diferentes núcleos e outros elementos do sistema.

O terceiro nivel de cache está optimizado para acadar unha alta capacidade de almacenamento, con valores típicos de 12MBytes (chega actualmente até uns 32MBytes nalgúns microprocesadores), con tempos de acceso da orde de 26-31 ciclos (depende do tamaño), e ancho de banda de 32 bytes/ciclo por cada núcleo. A latencia da memoria principal é de centos de ciclos. Polo tanto se unha instrución ou dato non está nun determinado nivel de cache, é necesario esperar ao acceso ao seguinte nivel. O peor caso é cando o procesador ten que esperar polo acceso á memoria principal, xa que implica unha espera de centos de ciclos. Os procesadores actuais son capaces de executar instrucións fora da orde orixinal do programa, para continuar facendo traballo cando existe un fallo cache nunha instrución ñLoadò ou ñStoreò. Non obstante, se non se atopan instrucións independentes do dato polo que se espera, o procesador entra en parada, esperando polo dato (ou instrución).

Cada núcleo tamén incorpora TLBs (Translation Lookaside Buffers) para acelerar a tradución dos enderezos de memoria virtual (que emite o procesador) a memoria física (que indica o emprazamento físico da páxina na memoria principal). Polo tanto, funciona como unha cache para a táboa de paxinación do sistema de memoria virtual. Por exemplo, para a familia de procesadores que se ilustra nas Figuras 2 e 3, existen dous niveles de TLB, un primeiro chamado DTLB (TLB de datos) e ITLB (TLB de instrucións) e un segundo nivel chamado STLB (TLB compartido para datos e instrucións). A DTLB ten 64 entradas para páxinas de 4KBytes, 32 entradas para páxinas de 2/4 MBytes, e 4 entradas para páxinas de 1 GByte. A ITLB ten 144 entradas. A STLB dispón de 512 entradas.

Un aspecto moi importante dos núcleos de procesamento é a súa capacidade para executar instrucións concorrentemente. Na Figura 2 podemos ver diferentes estruturas dedicadas á manter a ilusión dunha execución secuencial, facendo en realidade execución concorrente de instrucións. A parte superior do núcleo da Figura 2 obtén a secuencia de instrucións en orde da cache de instrucións (tal e como se indica no

programa). Estas instrucións son decodificadas para obter información de control e prepáranse para unha execución eficiente. Logo da execución de instrucións, estas son retiradas na orde do programa para manter a semántica secuencial. Unha parte importante na lectura e decodificación de instrucións é o da predición de saltos. Este hardware intenta adiviñar con elevada probabilidade cal será a seguinte instrución na secuencia do programa, é dicir adiviña se hai un elevada probabilidade de que a instrución que se está decodificando sexa de salto e se este salto se debe levar a cabo. Deste xeito pode facerse unha decodificación continua de instrucións, sen necesidade de esperar ao resultado final das instrucións de salto. Como é lóxico, en algunha ocasións o predictor de saltos falla, e entón e necesario facer as accións pertinentes para restaurar a correcta execución do programa.

Na parte inferior da Figura 2, estás a uniades funcionais nas que se executan as instrucións. Como vemos hai moitas unidades que poden operar en paralelo. Un elemento hardware (scheduler) está encargado de seleccionar instrucións independentes para lanzalas á execución fora de orde.

Un aspecto moi importante dun microprocesador é o seu ciclo de reloxo (ou a súa frecuencia de operación). Para entender as implicacións de ciclo de reloxo, é necesario falar da segmentación (ñpipeliningò en inglés) do microprocesador. Dedicaremos un tema completo ao estudo da segmentación dos núcleos, polo que de momento aquí introduciremos o concepto dun xeito básico para que nos permita entender o resto de contidos deste tema. O microprocesador é un circuíto combinacional moi complexo (con elevada profundidade lóxica). Nunha implementación ineficiente, as instrucións cargaríanse dende a memoria cache e executaríanse no bloque combinacional, de tal xeito que sería necesario esperar un tempo equivalente ao camiño crítico (o camiño de sinal con máis retardo) de todo o bloque combinacional para poder executar a seguinte instrución. Os microprocesadores segmentan o bloque combinacional, introducindo rexistros que manteñen illada electricamente unha etapa doutra. O paso dos sinais dunha etapa á seguinte está sincronizado por un reloxo. O ciclo de reloxo ten que ser superior ao retardo da etapa (peor caso do retardo combinacional entre rexistros, máis o retardo do propio rexistro) de maior retardo, ademais dunha penalización pola distribución do propio sinal de reloxo a todos os rexistros. Polo tanto, nun núcleo segmentado, podemos emitir instrucións ao ritmo marcado pola frecuencia de reloxo. Nun momento dado, estarán en execución diferentes instrucións, nas diferentes etapas nas que está segmentado o procesador. A Figura 3 ilustra este concepto (FO4 é unha medida de retardo).

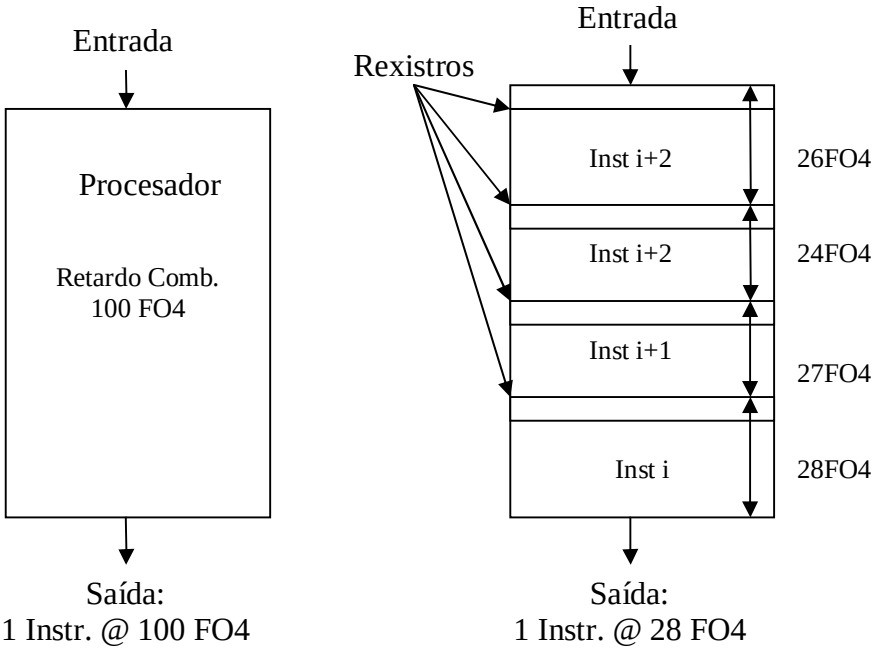


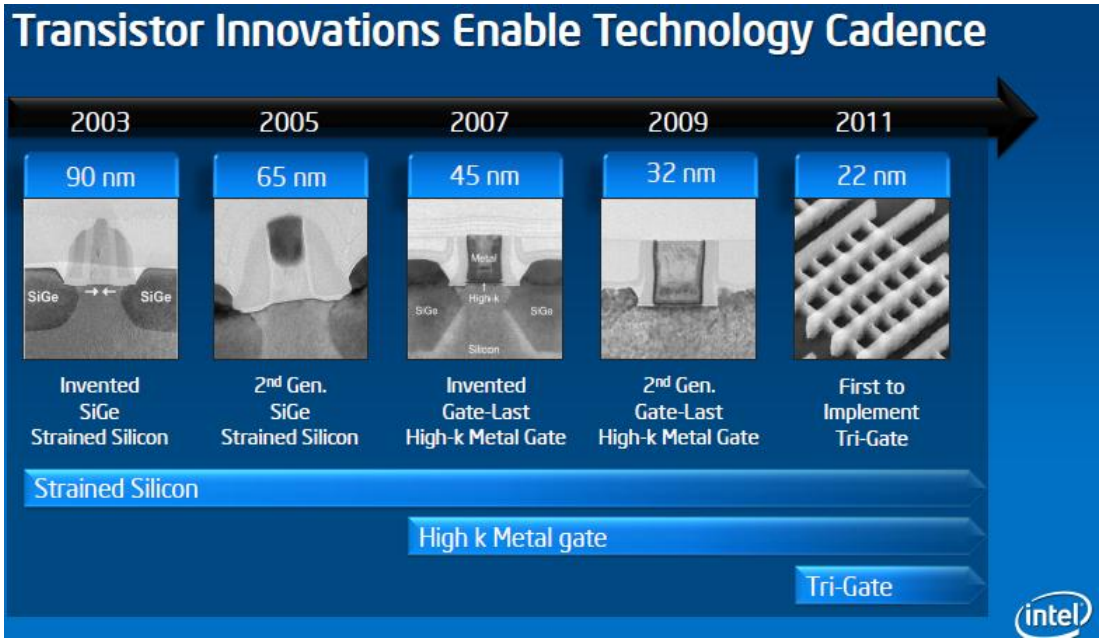
Figura 3: Segmentación

A profundidade de segmentación do procesador (ñpipeline depthò en inglés) está dada polo número de etapas nas que se dividiu o bloque combinacional (máis os ciclos de retardo de acceso ás memorias cache). Polo tanto, a profundidade de segmentación determina o número de ciclos que é necesario esperar dende que se emite unha instrución, até que o resultado da mesma está dispoñible. Canto máis pequeno é o ciclo de reloxo (maior frecuencia), maior é a profundidade de segmentación (máis etapas), e polo tanto será necesario esperar máis ciclos para ter o resultado dunha instrución emitida. Isto ten unha incidencia directa nas dependencias de datos entre as instrucións secuenciais do programa, e tamén sobre as instrucións de control de fluxo do programa (saltos). Se unha instrución ten que utilizar un dato doutra instrución que acaba de lanzarse, será necesario que esta instrución espere un número de ciclos determinado pola profundidade de segmentación. Se non é posible emitir unha instrución que non dependa deste resultado, o procesador deixará de emitir novas instrucións ao pipeline, e a produtividade baixa. O mesmo sucede coas instrucións de salto, no caso en que a predición do destino do salto fose incorrecta. Trataremos esta cuestión en maior profundidade nas seccións seguintes.

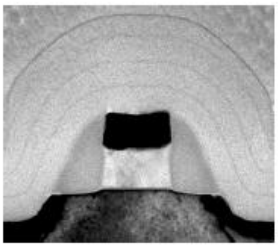
2- Tecnoloxías de Fabricación e Escalamento

Para entender as limitacións e características importantes dos procesadores actuais necesitamos coñecer os principios que rexen a súa tecnoloxía de fabricación. Os procesadores están constituídos por bloques hardware que se deseñan a nivel de porta. A súa vez, as portas lóxicas deséñanse en base a circuítos electrónicos que funcionan como interruptores: os transistores CMOS. Deste xeito podemos pensar que o microprocesador está constituído por transistores fabricados en silicio e liñas de cobre para interconectalos que se dispoñen en varios niveis sobre o silicio. Na Figura 4 podemos ver transistores CMOS para as diferentes tecnoloxías de fabricación que

utiliza Intel. Basicamente consiste en depositar capas de diferentes materiais sobre silicio, co problema engadido de que as dimensión nas que se traballa están xa no eido da nanotecnoloxía. Podemos ver tamén un esquema das conexións de cobre que van por encima dos transistores. Para poder conectar millóns de transistores é necesario incorporar varios niveis de conexión (8 na figura).



Transistor CMOS 45nm ***Capas de interconexión CMOS 45nm***



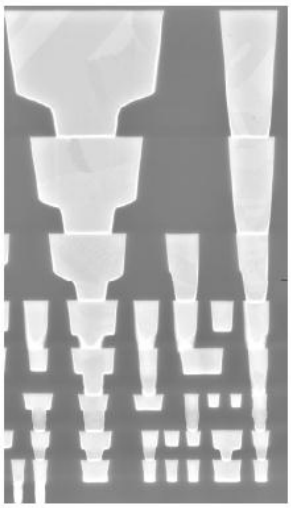


Figura 4: Tecnoloxía de fabricación CMOS.

Vamos a explicar agora como evoluciona (escala) a tecnoloxía de fabricación CMOS. Chamamos nodo tecnolóxico a un proceso caracterizado polo tamaño dos transistores e conexións que fabrica. Por exemplo, actualmente o nodo en produción masiva é o de 22

nanómetros, e este parámetro indica o ancho mínimo dunha conexión de cobre no nivel 1 e está directamente correlacionado coa superficie que ocupa o transistor.

[Escalamento das dimensións]. Dun nodo tecnolóxico ao seguinte (aproximadamente cada dous anos aparece un novo nodo) as dimensións lineais dos transistores e conexións escalan por un factor 0.7 (diremos que escala x0.7). Isto implica un escalamento ideal na área que ocupan os elementos de x0.5 (é dicir 0.7 x 0.7). Polo tanto un microprocesador cunha área de $2cm^2$ na tecnoloxía de 65 nm ocuparía idealmente unha área de $1cm^2$ na tecnoloxía de 45 nm (=65 x 0.7). Se dunha xeración a outra o microprocesador utiliza a mesma área, entón dispón idealmente do dobre de área para incluír máis elementos hardware que os que xa tiña (máis unidades funcionais, máis memoria cache, etc). Isto corresponde ao escalamento ñidealò. Non obstante, dentro dun microprocesador non todas as partes escalan do mesmo xeito, xa que depende moito da regularidade dos circuítos. Podemos esperar factores de escalamento de área moi próximos a x0.5 para arranxos de memoria cache, pero para o caso dos núcleos de procesamento os factores de escalamento poden ser de x0.6 ou x0.7.

A Figura 5 ilustra a evolución do número de transistores por chip ao longo do tempo. Esta evolución exponencial é o que se coñece como a ley de Moore (dobrar o número de transistores en cada nodo tecnolóxico).

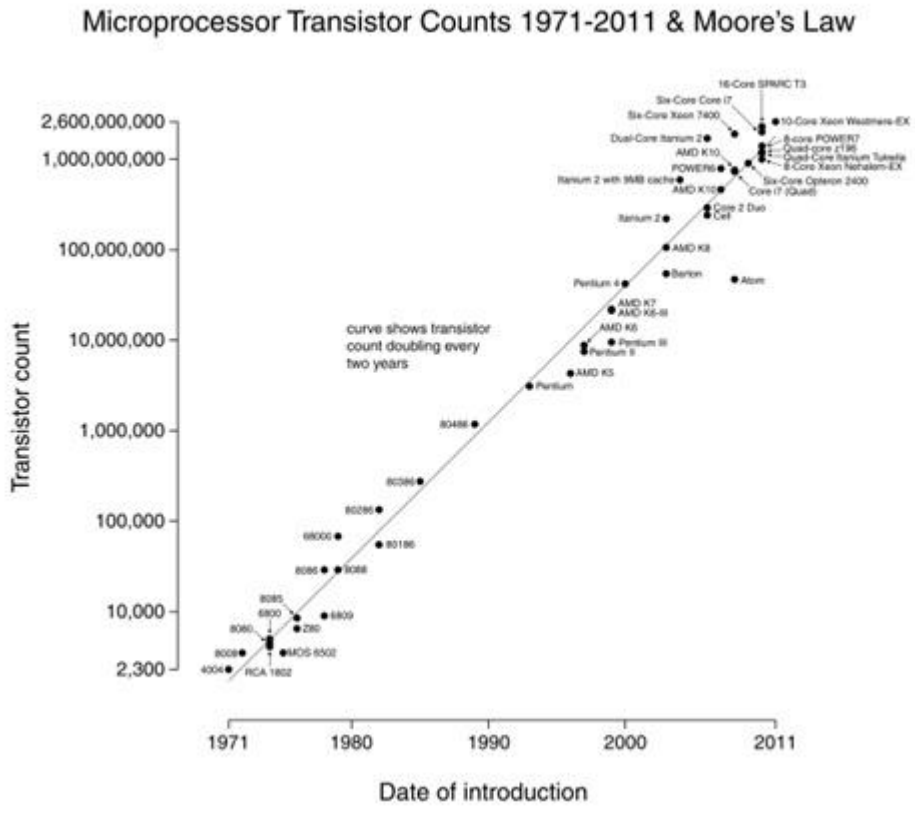


Figura 5: Ilustración da evolución do número de transistores por chip (ley de Moore).

[Escalamento da potencia]. A potencia disipada por un circuíto fabricado en CMOS é proporcional ao número de transistores activos (capacidade de carga activa), á

frecuencia de reloxo e ao cadrado da tensión de alimentación. O consumo de potencia escala mal en CMOS. Actualmente a potencia que disipa cada transistor escala por un factor x0.8 dunha xeración a outra, pero o número de transistores escala por x2. Ademais dunha xeración a outra as correntes de fuga (que non fan traballo lóxico, simplemente son perdas no chip) aumentan de xeito exponencial, e representan cada vez un porcentaxe máis significativo da potencia total (actualmente pode chegar a un 30%).

Debido as limitacións nas condicións de temperatura as que poden funcionar os circuítos existen limitacións no que atinxe á densidade de potencia disipada (watts por unidade de área do chip) e a presenza dos chamados ñhot spotsò, puntos no chip onde a densidade é moito máis elevada ca noutros (elevada actividade, por exemplo nas unidades para cálculo de direccións). Outra limitación importante é a da densidade de potencia a nivel de sistema (digamos nun sistema multiprocesador). Considérase que uns 10KWatts/m^2 é un límite razoable que permite utilizar sistemas de refrixeración económicos (ventiladores). Límites superiores fan que sexa necesario utilizar sistemas ñexóticosò de refrixeración que elevan considerablemente o costo, e que en xeral non están xustificados, excepto para o caso da supercomputación. En último termo unha métrica que nos pode ser moi útil para escoller un microprocesador é a de GFLOPS/Watt (10^9 operacións de punto flotante por segundo e por Watt consumido) para cargas de traballo con forte compoñente de cálculo numérico. A Figura 6 ilustra a eficiencia en GFLOPS/Watt e GFLOPS/mm^2 de área de chip para distintos procesadores de altas prestacións. Destacan o BlueGene/Q de IBM pola súa eficiencia enerxética, e os chips de computación orientada a gráficos (vectoriais) que presentan densidades de computación elevadas. Outra métrica, sobre todo para grandes sistemas multiprocesador, é a de GFLOPS/m^2, que indica a densidade física de computación que se pode acadar mantendo sistemas de refrixeración por ventilación.

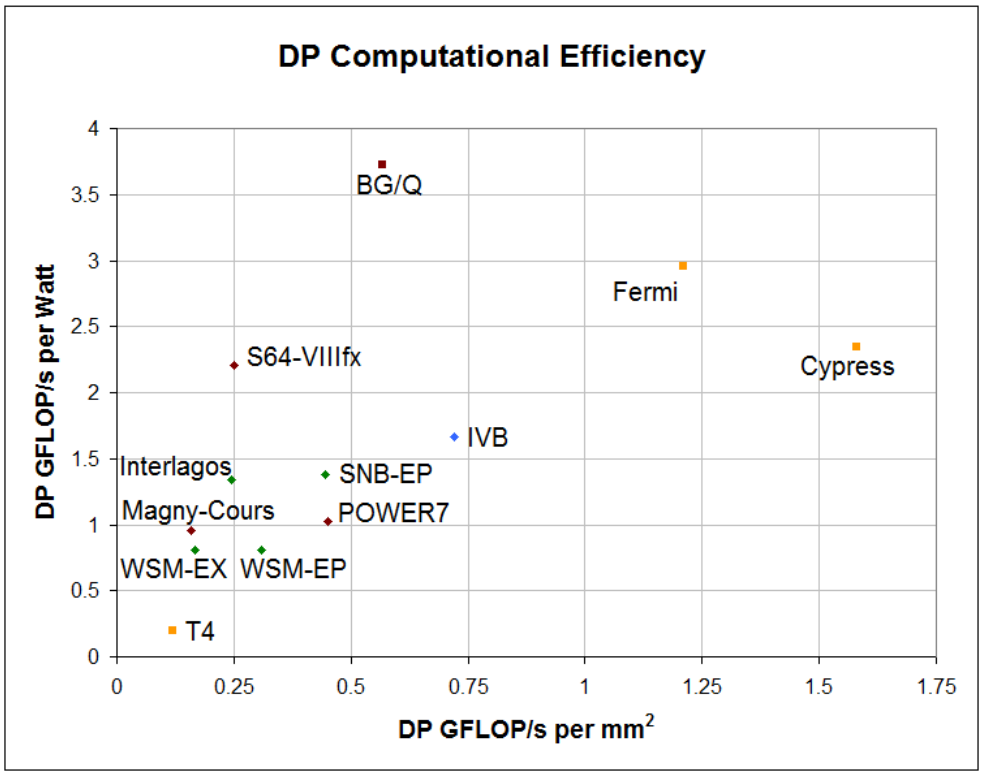


Figura 6: Eficiencia de microprocesadores de altas prestacións.

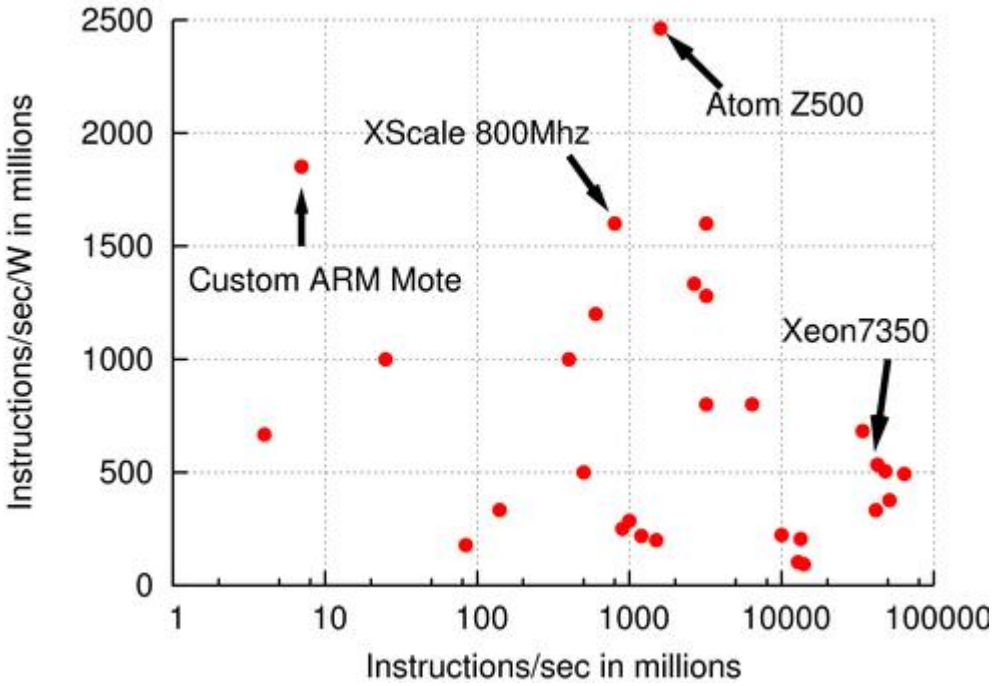


Figura 7: Microprocesadores nun espazo prestacións ï prestacións/W.

A Figura 7 amosa a eficiencia de microprocesadores de propósito xeral, incluíndo microprocesadores de moi baixas prestacións, pero esta vez en termos xenéricos das prestacións que acadan en número de instrucións executadas por segundo (aplicable polo tanto a cargas de traballo non dominadas pola computación numérica de punto flotante). A mellor posición corresponde á esquina superior dereita. A existencia de microprocesadores en zonas afastadas desta posición óptima está xustificada polo tipo de aplicación ao cal están destinados. Por exemplo os procesadores Xeon son utilizados para servidores de altas prestacións, no que o factor fundamental son as prestacións.

Non obstante, debe terse en conta que un sistema consta de máis elementos que o microprocesador. Por exemplo, engadindo un aumento de 0.1Watt fixo a cada microprocesador, transforma o espazo ilustrado na Figura 6. A Figura 8 amosa o resultado. Como vemos o procesador ñARM Moteò, que tiña una posición interesante en canto a prestacións/Watt, sufre unha baixada importante xa que este procesador só consume 0.5 Watt. Ao engadir 0.1 Watt, prodúcese unha baixada moi significativa na ratio prestacións/Watt.

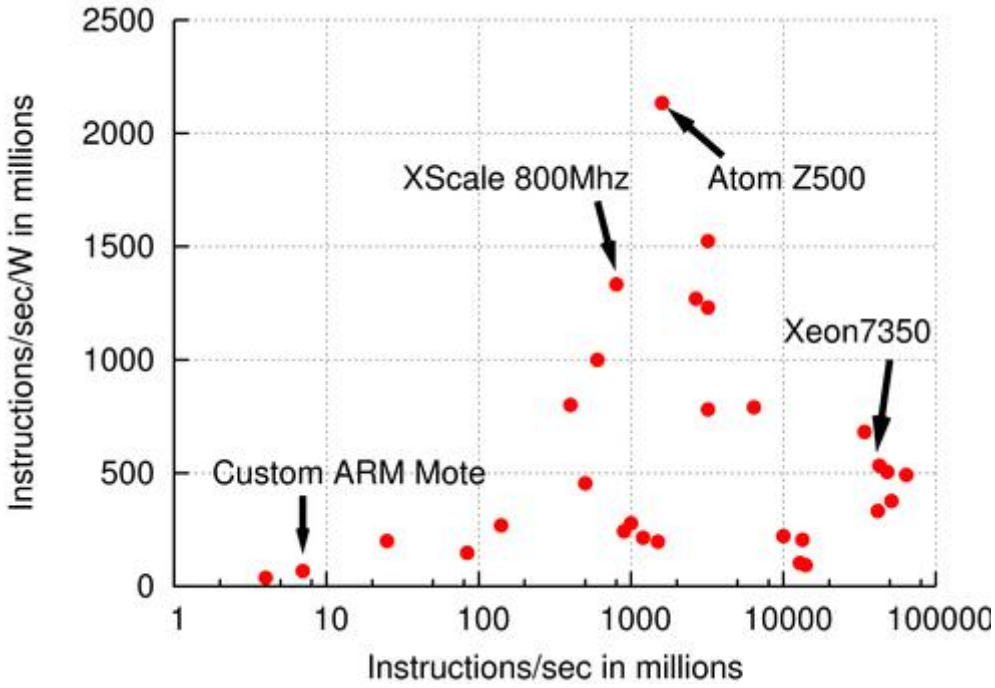


Figura 8: Efecto da suma dun termo fixo de potencia de 0.1 W.

Na Figura 9 podemos ver un detalle da evolución da TDP (Thermal Design Power: disipación de potencia para cal está deseñado o sistema de refrixeración) para distintas familias de microprocesadores. Como vemos, na última década chegouse a un límite (100-150 Watt) na disipación de potencia debido a cuestións de tipo técnico e económico. Manter este límite en cada nova xeración de microprocesadores require unha forte investigación en como optimizar o consumo de potencia (deshabilitando unidades non utilizadas, escalando a voltaxe de unidades non críticas na computación, innovacións a nivel de transistor, innovacións no deseño a nivel de circuíto, etc).

Na Figura 10 amósase como a distribución de temperatura non é uniforme dentro do chip, facendo que nalgúns puntos (ñhot spotsò) se supere a máxima temperatura recomendábel. Isto da lugar, se a situación persiste o tempo suficiente, a un comportamento pouco fiable ou ao deterioro do microprocesador. A potencia constitúe a gran limitación para o aumento de prestacións dos microprocesadores actuais e futuros.

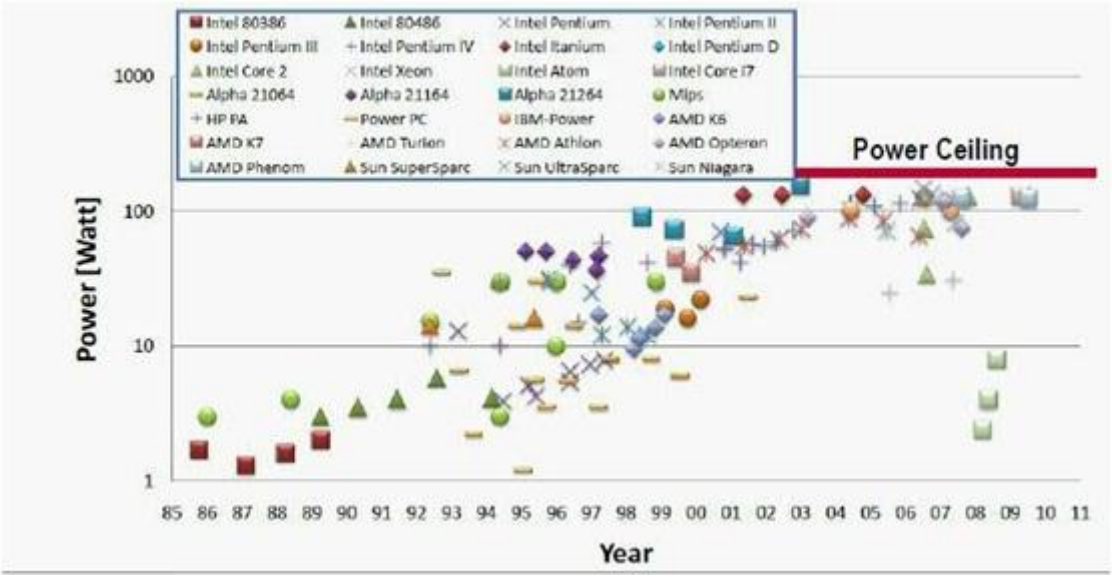


Figura 9: Evolución do consumo de potencia (TDP) para diferentes microprocesadores.

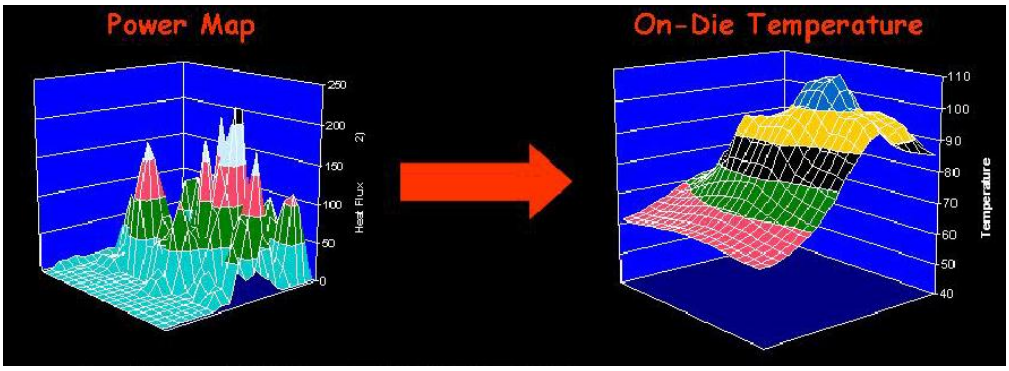


Figura 10: Distribución de temperatura dentro do chip (presenza de ñhot spotsò).

[Escalamento do retardo dos circuítos]. O retardo dos circuítos CMOS escala de xeito lineal coa tecnoloxía (por agora). Polo tanto redúcese nun factor x0.7 dun nodo tecnolóxico a outro. No canto de expresar os retardos dos circuítos de xeito absoluto (en picosegundos por exemplo), resulta máis interesante expresalo en termos dunha unidade de retardo chamada FO4 (fanout-of-four). Un FO4 é o retardo que ten un inversor cunha carga de catro inversores. Tense verificado que a expresión do retardo dos circuítos normalizados ao retardo de un FO4 é practicamente independente do nodo tecnolóxico. Por exemplo, podemos dicir que un sumador rápido ten un retardo equivalente a 10 FO4. Iso significa que o seu retardo real equivale a 10 veces o retardo dun inversor cargado con catro inversores.

Para converter os retardos en FO4 a retardos absolutos é necesario coñecer cal é valor dun FO4 en función do nodo tecnolóxico. Experimentalmente determinouse que 1 x FO4=0.2 x T (nm) (picosegundos), onde T é a lonxitude que identifica o nodo tecnolóxico. Polo tanto para a tecnoloxía CMOS de 45nm o retardo de 1FO4=0.2 x 45=9ps. Isto corresponde as condicións de retardos típicos. Nas peores condicións de

operación (temperatura no rango máximo e variacións da tensión de alimentación e parámetros de fabricación que aumentan o retardos dos transistores), este retardo aumenta aproximadamente por un factor x1.4.

Supoñamos que un microprocesador ten un ciclo de reloxo de 13 FO4. A frecuencia de reloxo que corresponde a este ciclo na tecnoloxía de 45nm é 1/(13 x 9 ps)=8.5GHz. O mesmo procesador tería unha frecuencia de reloxo de 1/(13 x 13 ps) =5.9 GHz para unha tecnoloxía de 65nm.

No gráfico da Figura 11 represéntanse os diferentes microprocesadores de Intel ao longo do tempo. Así por exemplo o Intel Pentium II de 1997 tiña unha frecuencia de 300Mhz nunha tecnoloxía de 350nm. Isto implica un ciclo de reloxo de 3.3 (ns/ciclo) / 70 (ps/FO4)=47 FO4. Sen embargo o Pentium 4 con HyperThreading de 2002 a 3GHz nunha tecnoloxía de 130 nm tiña un ciclo de reloxo de 0.33 (ns/ciclo) /26 (ps/FO4)=12.5 FO4. Isto implica que o pipeline era moito máis profundo (máis etapas de pipeline) neste último caso.

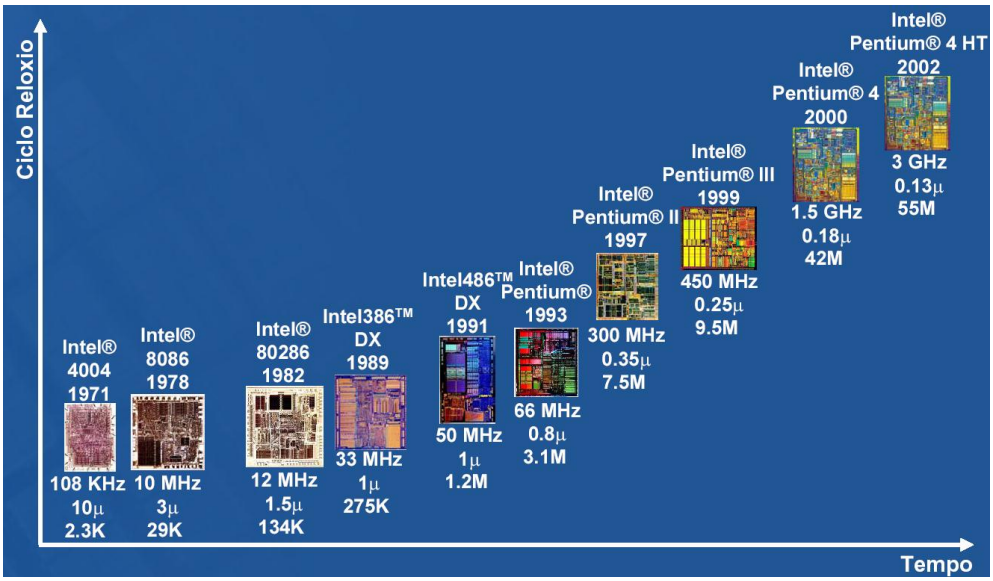


Figura 11: Evolución da frecuencia de reloxo de microprocesadores Intel.

Os circuítos que compoñen os microprocesadores seguen dúas regras xerais que relacionan o seu retardo, área e potencia e o nodo tecnolóxico. Por un lado tense verificado que para un determinado circuíto cúmprese a relación Latencia^3 x Potencia=cte(L). Isto implica que se se pretende diminuír a latencia á metade (factor x1/2), podemos esperar que o circuíto consuma x8 veces máis potencia. Por outra banda a cte diminúe co nodo tecnolóxico.

En relación con isto, é interesante mencionar a relación entre a voltaxe e a frecuencia (inverso do ciclo de reloxo, proporcional ao retardo dos circuítos). Así, de xeito aproximado, para as tecnoloxías actuais, podemos dicir que se escalamos (baixamos) a voltaxe dun circuíto por un factor Kv, a frecuencia deberá escalar por un factor:

Kf=1.45 Kv ï 0.40

Por exemplo, escalando a voltaxe por un factor x0.6, obriga a escalar a frecuencia por un factor aproximado de x0.5. O efecto do escalamento da voltaxe no consumo de potencia está dado polo produto Kf x Kv^2 , xa que o consumo de potencia ten dependencia cuadrática coa voltaxe e lineal coa frecuencia. Para o exemplo anterior, a potencia escalaría por un factor aproximado de x0.2. Os procesadores actuais explotan esta propiedade de xeito masivo. Por un lado, para rebaixar o consumo de potencia nos circuítos que non están nos camiños críticos, a por outra, mediante o escalado dinámico da voltaxe ao núcleos de procesamento cando non demandan computación intensiva. O escalamento dinámico da voltaxe será masivamente utilizado nas futuras xeracións de microprocesadores para poder seguir escalando en capacidade de computación mantendo uns incrementos de consumo de potencia moderados.

Existe unha relación semellante entre a latencia dun circuíto e a área que ocupa: Area x $Latencia^n$ = cte(L) con n entre 1 e 2. Por exemplo para n=2, se pretendemos reducir nun factor x1/2 a latencia dun circuíto, podemos esperar aumento de área por un factor x4. A cte diminúe co proceso tecnolóxico.

Como resumo desta sección mostramos as Figuras 12 e 13. A Figura 12 ilustra o efecto da tecnoloxía nas prestacións e no consumo de potencia. Neste gráfica preséntanse datos para diferentes servidores en base a procesadores Intel Xeon, fabricados en diferentes nodos tecnolóxicos (é dicir, dun nodo a outro cambiou a tecnoloxía de fabricación, pero tamén a arquitectura do procesador para optimizala á nova tecnoloxía). O eixo x corresponde as prestacións do sistema, e o eixo y corresponde á potencia que consume. Cada sistema é escalable en prestacións, variando o número de núcleos/procesadores activos, frecuencia de reloxo, etc. Como vemos dado un nivel de prestacións, o novo nodo tecnolóxico permite reducir o consumo de potencia dun xeito considerable. Por outra banda, cada nodo tecnolóxico permite acadar un importante incremento no nivel de prestacións consumindo a mesma potencia.

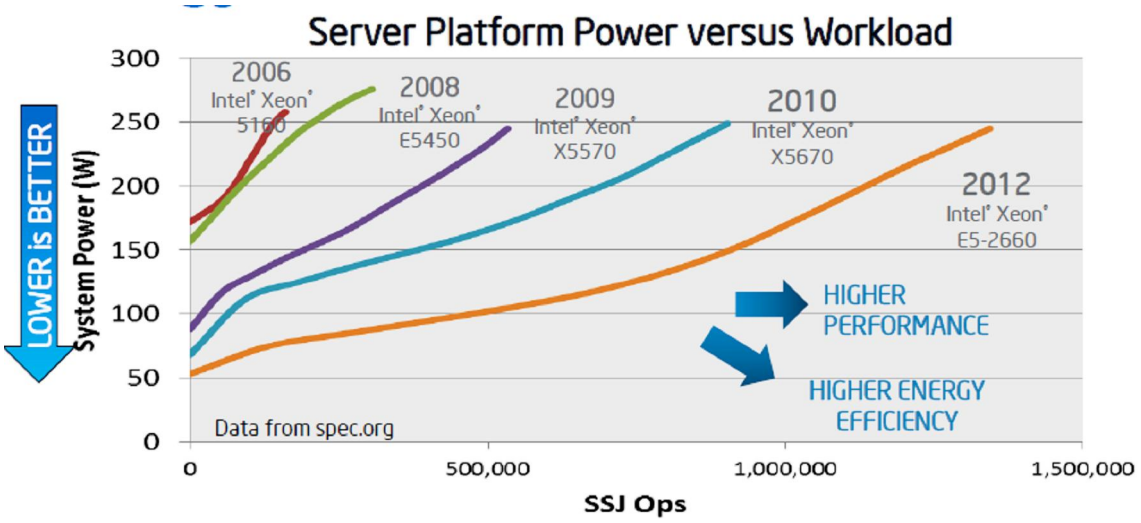


Figura 12: Evolución tecnolóxica de servidores Intel Xeon: exemplo do efecto da tecnoloxía.

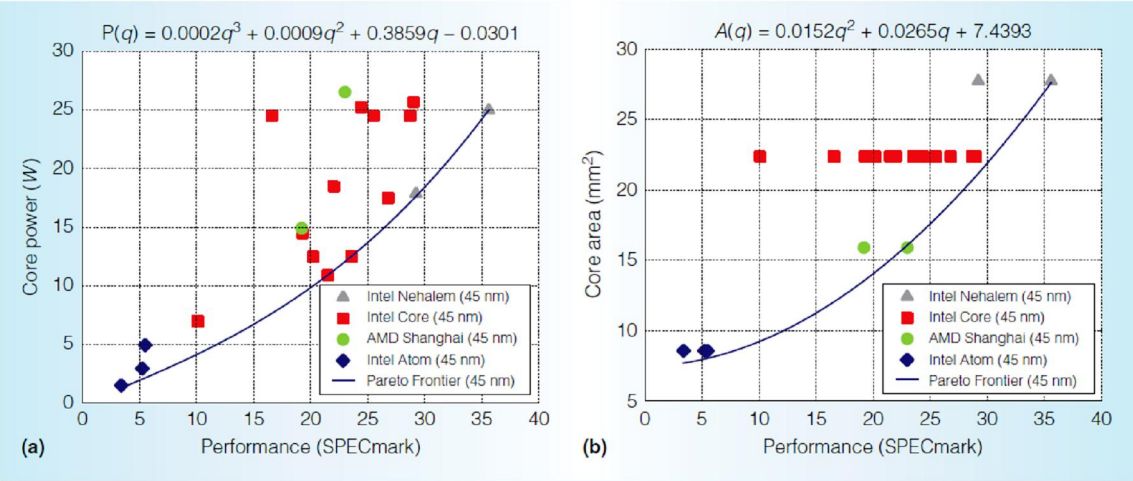


Figura 13: Relación Potencia-Prestacións e Área-Prestacións para núcleos en tecnoloxía de 45 nm.

A Figura 13 ilustra a dependeza entre prestacións e potencia, e prestacións e área. A Figura da esquerda amosa a potencia por núcleo de procesamento vs as prestacións por núcleo medidas en unidades SPECmark (a figura de mérito que acada o procesador ao correr os benchmarks SPEC), para a tecnoloxía de 45 nm. Como vemos, os puntos óptimos caen nunha curva con forma cúbica. A Figura da dereita amosa a área por núcleo vs as prestacións por núcleo segundo as unidades SPECmark. Neste caso, os puntos óptimos caen nunha curva con forma cuadrática.

3- Escalamento Hardware dos Microprocesadores

Vexamos agora a evolución (escalamento) das características tecnolóxicas dun procesador para computación de altas prestacións (Intel Itanium). Na táboa da Figura 14 vemos catro implementacións diferentes do procesador en varios procesos tecnolóxicos (180nm no 2001, 130 nm no 2003, 90 nm no 2005 e 65nm no 2008). Na táboa amósanse os factores de escalamento, dunha xeración a outra, de parámetros fundamentais para as prestacións do procesador, como son a frecuencia de reloxo, o tamaño de memoria cache de último nivel ou o ancho de banda de comunicación coa memoria principal (BWN: taxa de transferencia nominal da interconexión coa memoria). Ademais podemos ver a área e o número de transistores de cada implementación, e o número de núcleos de procesamento.

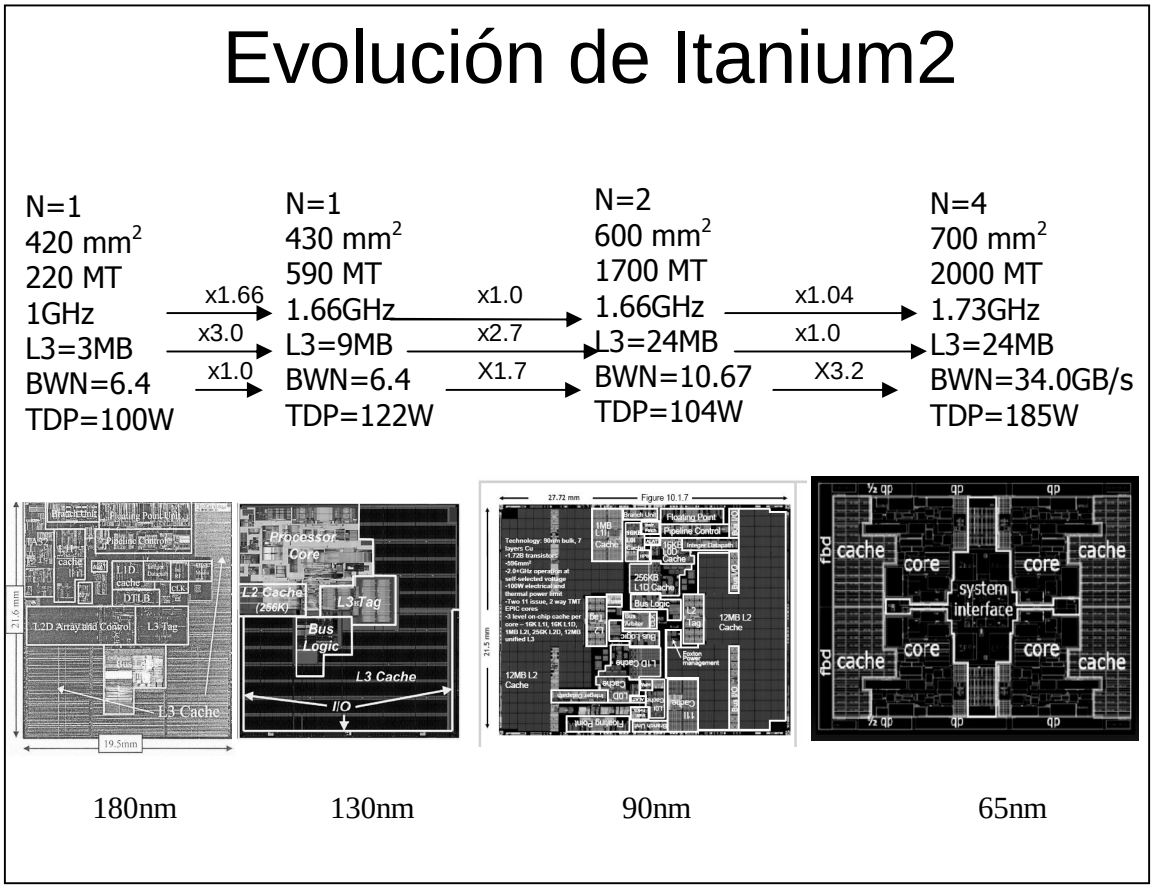


Figura 14: Evolución da familia de microprocesadores Intel Itanium 2.

Podemos inferir o número de transistores dedicados ao procesador e á memoria cache (e tamén as súas áreas correspondentes). Da foto do chip para o nodo de 130nm con 9MB deducimos que o 61% da área está ocupada pola cache de nivel 3, e que o procesador e os pads de entrada/saída e a lóxica de interface co bus ocupan o 39% restante (25% o procesador, 5% a lóxica de interface co bus e 9% os pads de entrada/saída). Polo tanto a distribución de área e a seguinte: 109mm^2 (procesador) + 21 mm^2 (interface bus) + 37 mm^2 (I/O) + 263 mm^2 (9MB cache). Isto implica uns 29.2 mm^2/MB de cache de nivel 3. Para o nodo de 180nm en principio a área dos elementos sería o dobre. Sen embargo é probábel que a área de interface co exterior non escale, e que só escale a memoria cache e o procesador. Facendo esta suposición a área esperada para o nodo de 180 nm é a seguinte: 109 x 2 mm^2 (procesador) + 29.2 x 2 x 3 mm^2 (cache de 3MBytes) + 21 mm^2 (interface bus) + 37 mm^2 (I/O) = 219 mm^2 (procesador) + 175 mm^2 (cache) + 21 mm^2 (interface bus) + 37 mm^2 (I/O) = 452 mm^2 (non lonxe dos 420 que da Intel como dato).

Verbo do número de transistores, na transición de 180nm a 130nm o procesador aumenta en 6MB a cache de nivel 3, que corresponde a un aumento de 370 millóns de transistores. Polo tanto inferimos que se requiren uns 61 millóns de transistores por MB de cache de nivel 3. Isto implica que o procesador (e o nivel 1 e 2 de cache) e a interface co bus e I/O requiren 590-549=41 millóns de transistores. Efectivamente isto é coherente coa implementación no nodo de 180nm: 41 (procesador+interface+I/O) + 183 (3MB cache) =224 millóns (moi aproximado).

Resulta interesante destacar as diferenzas de densidade de transistores da memoria cache en comparación coa densidade para a lóxica do procesador. Para a cache temos unha densidade de (millóns que ocupan 9MB dividido polo que ocupa en área) 549 millóns/263 mm^2= 2.1 millóns de transistores/mm^2. Para o procesador+interface+I/O temos 41 millóns/167 mm^2= 250 mil transistores/mm^2. Esta enorme diferenza é debida á estrutura totalmente regular da memoria cache, fronte a estrutura aleatoria da lóxica do procesador, que fai que a densidade sexa moi inferior.

Máis en xeral, e no que atinxe á área, a Figura 15 ilustra a distribución da área do chip para microprocesadores para servidores ou grandes sistemas. A distribución englóbase en tres compoñentes: núcleos, cache é sistema. A área de sistema refírese á área dedicada aos controladores de memoria integrados, á interface coa memoria, e ao hardware necesario para configurar sistemas multiprocesador. Os valores medios aproxímanse a un reparto equitativo entre os tres tipos de compoñentes (34% para núcleos, 30% para cache, e 36% para hardware de sistema).

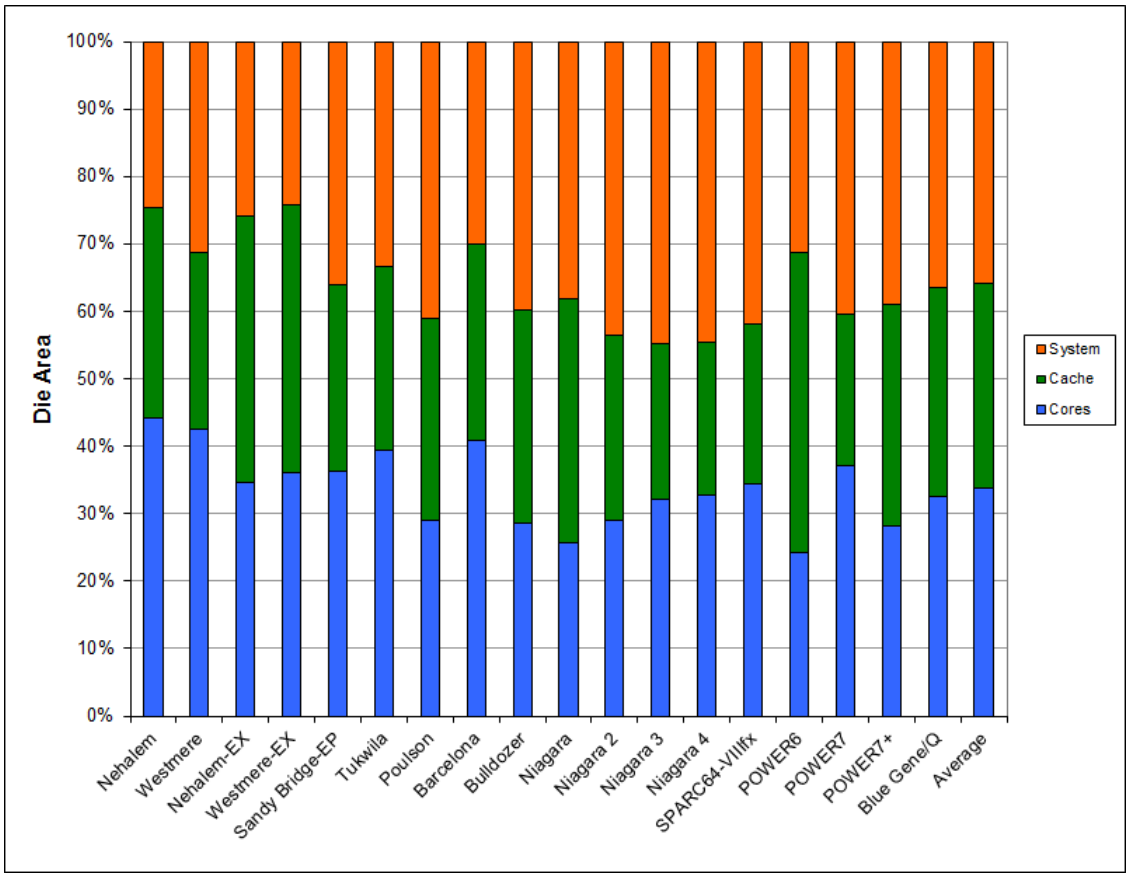


Figura 15: Distribución de área en microprocesadores para servidores (system indica área adicada ao controlador de memoria integrado, á interface coa memoria, e a hardware adicional para configurar sistemas multiprocesador).

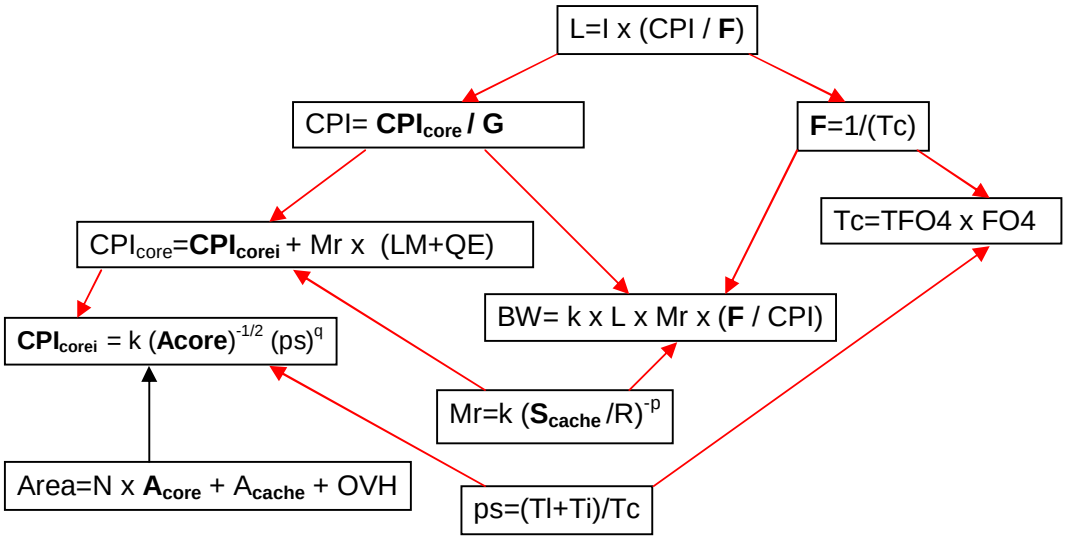
4- Escalamento de Prestacións en Microprocesadores

Pasamos agora a discutir os elementos fundamentais que permiten explicar o escalamento en prestacións dos microprocesadores. Un dos aspectos que explicaremos logo de expoñer esta sección serán as causas do esgotamento das prestacións dos microprocesadores superescalares (capaces de executar varias instrucións por ciclo) convencionais cun só núcleo.

Na Figura 16 amosamos as principais ecuacións que definen, cun modelo de primeira orde, as prestacións dos microprocesadores. En xeral as ecuacións contemplan a posibilidade de contar con varios núcleos de procesamento (dado polo parámetro N), e que as aplicacións están divididas en fíos (threads) para poder procesalas en paralelo nos diferentes núcleos. Ademais contémplase a posibilidade de que cada núcleo soporte a execución a tempo compartido de varios fíos (multithreading hardware, descrito polo parámetro T). Os parámetros máis significativos son os seguintes:

1- L: Latencia dun programa con I instrucións.

2- F: Frecuencia de reloxio do procesador, que é igual ao inverso do tempo de ciclo do pipeline do procesador (Tc). O tempo de ciclo do procesador podemos expresalo como o produto de dous termos: o número de unidades FO4 por ciclo polo retardo dun FO4 (que escala linealmente co parámetro que define a tecnoloxía).
3- CPI: número medio de ciclos por instrución executada. Basicamente, logo da execución dun programa correspondería ao cociente entre o número de ciclos totais consumidos e o número de instrucións executadas. CPI_{core} corresponde ao CPI individual de cada núcleo de procesamento do microprocesador. CPI_{corei} corresponde ao CPI dun núcleo, supoñendo o caso ideal no que non se producen fallos de cache de último nivel (é como se a penalización por acceso a memoria principal fose nula).
4- G: é o factor de ganancia en velocidade por utilizar N núcleos de procesamento para executar o programa, en comparación coa velocidade de execución cun só núcleo. Estritamente, é cociente entre os ciclos por instrución que acada un só núcleo e os ciclos por instrución que acada o procesador completo con N núcleos (CPI_{core} /CPI).
5- Mr: taxa de fallos de cache de último nivel por instrucción (por exemplo, unha taxa 0.001, significa que en media por cada 1000 instrucións executadas, prodúcese un fallo da cache de último nivel. Este fallo da cache implica un custe medio de LM ciclos cando o sistema de memoria non está saturado con peticións. A este custe engadimos unha penalización de QE ciclos debido a efectos de cola cando hai un número elevado de peticións.



I: número de instrucións que executa o programa.
CPI: ciclos por instrucción do procesador.
CPI_{core}: CPI para cada núcleo de procesamento.
CPI_{corei}: CPI para cada núcleo supoñendo que non hai fallos de cache de último nivel.
F: Frecuancia de procesador.
N: número de núcleos de procesamento.
G: factor de ganancia en velocidade por utilizar N núcleos de procesamento.
Mr: taxa de fallos por instrucción da cache de último nivel.
LM: latencia media por penalización de acceso á memoria principal (fora do chip) en número de ciclos do procesador.
QE: penalización en ciclos por acceso á memoria principal debido a efectos de cola cando hai moitas peticións de acceso.
BWM: ancho de banda máximo sostido dispoñible para comunicación coa memoria.
BW: ancho de banda requirido polo sistema nun momento dado.
L: tamaño da liña cache.
S_{cache}: tamaño da cache de último nivel.
R: factor efectivo polo que se reduce o tamaño de cache total de último nivel visto por cada núcleo, debido á execución concorrente de múltiples fíos.
ps: número de etapas do pipeline (profundidade do pipeline).
Tc: tempo de ciclo do procesador en unidades de tempo absoluto.
TFO4: ciclo do procesador normalizado ao retardo dun FO4.
Tl: suma do retardo das portas lóxicas no camiño crítico dun núcleo.
Ti: suma do retardo das interconexións no camiño crítico dun núcleo. Está dominado polas conexións longas.
Area, A_{core}, A_{cache}: áreas do procesadores, área dun núcleo e área ocupada pola cache de último nivel respectivamente.
OVH: contribución a área por outros elementos distintos dos núcleos e cache. Por exemplo: sistema de interconexión interno ao chip (rede de interconexión), controladores de memoria integrados, interfaces coa rede integrados, etc.
p,q,k: constantes dependentes de moitos factores.

Figura 16: Ecuacións de rendemento para microprocesadores.

Respecto das ecuacións, cabe salientar o seguinte:

[**Latencia**]: a latencia estímase polo produto do CPI, o número de instrucións (con isto obtense o número total de ciclos) e o inverso da frecuencia (tempo efectivo que require cada ciclo). Debe terse en conta que F/CPI corresponde ao número de instruccións executadas por segundo.

[**CPI**]: estímase como o produto do CPI_{core} e o inverso do factor de ganancia en velocidade G por utilizar N núcleos de procesamento. Isto deriva directamente da definición do factor de ganancia en velocidade.

[**CPI_{core}**]: ten dúas compoñentes, unha debido á latencia propia do procesamento das instrucións no núcleo de procesamento (**CPI_{corei}**), e outra debida á latencia de espera por datos da memoria principal cando se produce un fallo na cache de último nivel. Os núcleos de procesamento actuais soportan máis dun fío (thread) hardware para enmascarar parte da latencia de memoria. Cando un fío queda parado por un fallo da cache ou pola resolución dun salto, ou por unha dependencia doutro resultado no pipeline, pasa a executarse o seguinte fío, co que se solapa a execución coa latencia de resolución deste conflitos. A Figura 17 ilustra un exemplo do efecto da execución ñmultifíoò hardware. Cun só fío, a contribución das paradas do procesador é de 3.79 ciclos por instrución (por cada instrución que se executa en media, é necesario esperar 3.79 ciclos debidos a fallos de cache, e dependencias no pipeline), resultando un CPI total de 4.79 (0.21 instrucións por ciclo). Na parte inferior da Figura 17, observamos a execución con catros fíos. Nese caso cando un dos fíos para, entra a executarse o fío seguinte. A presenza de varios fíos fai que se produzan algúns conflitos no pipeline, xa que algún deles no poderá avanzar por mor da execución doutro nese momento. Aínda así, observamos que o tempo de espera por instrución no procesador reduciuse a 1.56 ciclos, resultando un CPI de 1.39 (0.72 instrucións por ciclo).

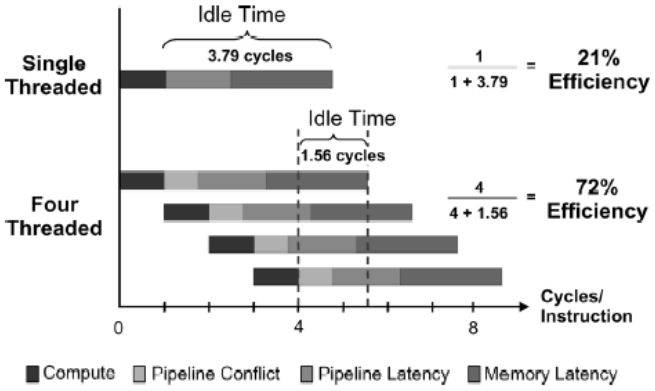


Figura 17: Redución do efecto da latencia de memoria mediante ñmultithreadingò.

A introdución de múltiples fíos no é dende logo a costo cero. Como xa mencionamos, cantos máis fíos se poidan executar, maior é a probabilidade de que cada fío teña que estar parado esperando aos recursos utilizados por outro no pipeline. Ademais implica aumentar o costo hardware, xa que algunhas das estruturas do pipeline son dedicadas para cada fío (por exemplo o conxunto de rexistros). Por outra banda, se os fíos non

comparten datos, o número de fallos das caches aumentará, xa que os datos e instrucións duns fíos interfiren cos outros no uso das estruturas de memoria. Unha última cuestión a ter en conta é que os fíos que intercalan a execución deben ser independentes, o cal non sempre é fácil de atopar, e depende moito do tipo de carga de traballo, e do deseño (programación) da aplicación.

Finalmente mencionar que incluímos na ecuación o termo QE (que se suma á penalización por acceso á memoria principal), para ter en conta o posible efecto da limitación de ancho de banda na latencia do programa. Cando o ancho de banda requirido (BW) se achega ao ancho de banda máximo sostible (BWM), este termo toma valores significativos debido aos efectos de cola, co que aumenta de xeito efectivo a latencia de acceso á memoria. Neste caso ao haber moitos requirimentos de ancho de banda, algúns fíos quedarán bloqueados xa que o ancho de banda é un recurso limitado.

A latencia media por acceso á memoria principal (LM) non se corresponde coa latencia total do sistema de memoria principal. Isto é debido a que o sistema permite ter varias transaccións de memoria activas de xeito concorrente e procesándoas coma se fose un pipeline. Por exemplo, para un procesador actual a latencia total de memoria pode ser duns 60 ns (co controlador de memoria integrado no chip do procesador). Un valor típico para o límite do número transaccións de memoria concorrentes por parte dun núcleo de procesamento é de 10. Polo tanto a latencia media por acceso, LM, será de 60/10=6ns. En termos do número de ciclos do procesador, é necesario multiplicar este valor absoluto de tempo pola frecuencia. Por exemplo para un procesador cunha frecuencia de 3GHz, o valor de LM en ciclos é de 6 ns x 3 GHz =18 ciclos (observar que a latencia total de memoria é de 60ns x 3 GHz=180 ciclos).

[Mr]: a taxa de fallos por instrución na cache de último nivel está directamente correlacionada co tamaño desta e tamén con número de fíos que se executan. Tense comprobado que Mr depende do inverso da potencia p do tamaño de cache de último nivel, con valores de p entre 0.3 e 0.7 (depende da carga de traballo e incluso do propio tamaño da cache). Non obstante debe terse en conta que debido á execución concorrente de varios fíos, o tamaño efectivo que pode utilizar cada núcleo está reducido por un factor R. Este factor R está comprendido entre 1 e N x T (número de núcleos multiplicado polo número de fíos hardware que soporta cada núcleo). O caso R=1 corresponde á situación na que todos os fíos comparten todos os datos e non teñen datos privados. O caso R=N x T corresponde á situación na que se executan N x T fíos que non comparten ningún dato. Obviamente o máis probable é un valor intermedio no que os fíos comparten parte dos datos e teñen tamén datos privados. En xeral tomaremos R=N como valor de compromiso.

[**BW**]: o ancho de banda requirido estímase en primeira orde como o produto do número de fallos por segundo da cache de último nivel e multiplicada polo costo en bytes dun fallo (tipicamente o tamaño en bytes dunha liña cache L; engadimos o factor k para ter en conta as escrituras en memoria cando a liña cache a substituír foi modificada). O número de fallos de cache por segundo estímase coma o produto da taxa de fallos por instrución (Mr) polo número de instrucións por segundo (dado pola frecuencia dividida polo CPI).

[$\mathbf{CPI_{corei}}$]: Este é o CPI dun núcleo supoñendo que non hai fallos de acceso á cache de último nivel que obrigan a buscar os datos en memoria principal. Polo tanto este

parámetro inclúe as penalizacións típicas por execución en pipeline (dependencias de datos e saltos), e por fallos de acceso ás caches de nivel interno (normalmente nivel 1 e 2). Supoñemos que este parámetro aumenta coa profundidade do pipeline (ps) e diminúe coa área dedicada a cada núcleo (Acore). A forma exacta da dependencia destes dous parámetros non é sinxela. Para a dependencia coa área do núcleo tense verificado unha relación proporcional ao inverso da raíz cadrada. A Figura 18 ilustra a dependencia de CPI_{corei} con ps e Acore.

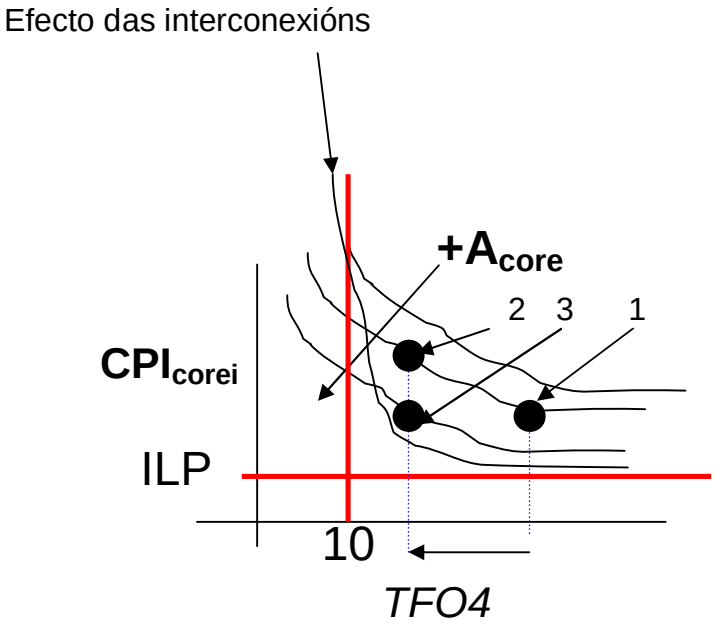


Figura 18: Efecto do tempo de ciclo e a área no $CPI_{corei.}$

No eixo *x* da Figura aparece o tempo de ciclo, medido en número de FO4. A profundidade do pipeline é inversamente proporcional ao tempo de ciclo, polo que ao movernos á esquerda no eixo *x* estamos aumentando o número de etapas do pipeline, e o CPI_{corei} comeza a aumentar, dun xeito especialmente importante en puntos preto a ciclos de 10 FO4. Isto é debido a que ao ser o pipeline máis profundo, teremos máis tempos de espera no pipeline por mor dos saltos e dependencias de resultados, que farán máis complicado atopar instrucións que se poidan executar inmediatamente. Aumentando a área pode reducirse este efecto. Por exemplo, pódense engadir unidades de predición de saltos máis sofisticadas, ou pode utilizarse especulación. Como exemplo, na Figura amósase que a redución de ciclo dos puntos 1 ao 2 produce un aumento de CPI do núcleo. Para manter o CPI orixinal teríase que aumentar a área, que correspondería ao punto 3. As diferentes curvas corresponden a diferentes áreas do núcleo. Por outra banda, como se observa na gráfica, a partir de certo valor, o aumento do tempo de ciclo en FO4 (diminución do número de etapas do pipeline) non implica unha redución do CPI. Isto é debido a que nese caso o CPI do núcleo está limitado polo paralelismo a nivel de instrución (ILP), que está derivado do modelo de programación secuencial. Finalmente mencionar que o espazo de deseño está limitado por un lado polo ILP, que marca o límite inferior do CPI do núcleo, e un ciclo mínimo duns 10 FO4 (por razóns de implementación, non se considera fiable ter reloxos con menos retardo), que potencialmente marca o máximo CPI do núcleo para unha área dada.

4.1 Exemplo de Aplicación do Modelo de Escalamento

Podemos aplicar este modelo para estimar os factores de escalamento que aparecen na Figura 14 entre as diferentes xeracións do procesador Itanium 2. A modo de exemplo ilustraremos o uso das ecuacións para o escalamento do Itanium 2 de 180 nm a 130 nm. O resto das transicións serán propostas como exercicio.

No que atinxe ao escalamento de microprocesadores, un obxectivo bastante común é o de escalar a latencia por un factor x0.5 (é dicir, que o mesmo programa corra dúas veces máis rápido no novo procesador). Na Figura 19 amósanse os posibles factores de escalamento utilizados para o procesador fabricado en 130nm, a partir do procesador fabricado en 180 nm. Para o factor p que condiciona a relación entre Mr e o tamaño da cache de último nivel utilizamos o valor p=0.65 que é típico para as cargas de traballo dos procesadores Itanium 2.

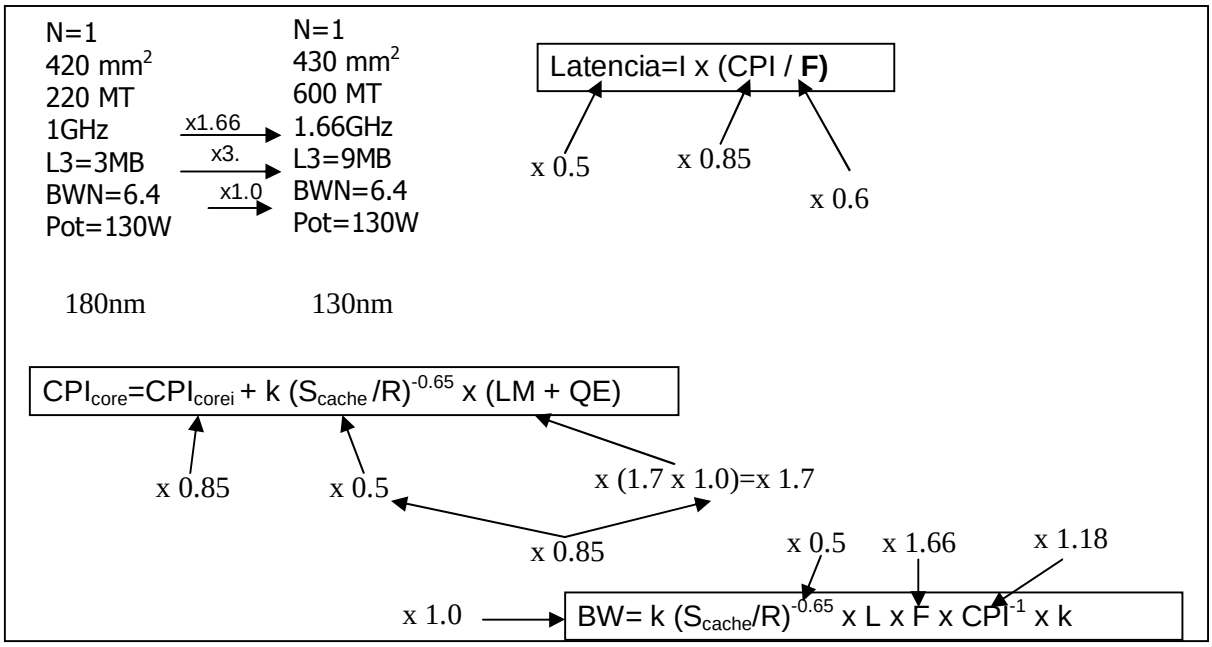


Figura 19: Escalamento do Itanium 2 de 180nm a 130nm.

Un dato coñecido é que nesta transición, a frecuencia escalou por un factor x1.7. Polo tanto para acadar unha redución en latencia por x0.5, é necesario que o CPI escale por 0.85=1.7 x 0.5. Posto que o número de núcleos é un en ambas xeracións (G=1), implica que o CPI_{core} debe escalar por x0.85. Unha posibilidade é pensar que ambos termos de CPI_{core} escalan por ese mesmo factor (isto equivale a supoñer que ambos termos contribúen na mesma porcentaxe ao CPI; existen dende outras posibilidades, como supoñer que a carga de traballo está dominada pola computación na que o compoñente debida ao núcleo de procesamento constitúe o 70% e a compoñente debida ao acceso de memoria o 30%; de forma semellante podemos supoñer unha carga de traballo dominada polo acceso á memoria, cunha distribución nos pesos do 30% e 70% respectivamente). Polo tanto consideramos que se escalou o CPI_{corei} por un factor x0.85 (isto é posible con certas melloras no pipeline e no compilador para aumentar o paralelismo a nivel de instrución). Respecto do outro termo, dicir que podemos supoñer

que a latencia absoluta (en ns) da memoria principal non mellora con respecto á xeración anterior (factor de escalamento x1.0). Sen embargo, ao escalar a frecuencia de reloxo do procesador, a latencia en número de ciclos da memoria aumenta por un factor x1.7. Isto implica globalmente un incremento no número de ciclos de latencia da memoria de 1.7. Para acadar neste termo o factor x0.85, é necesario que a número de fallos por instrución na cache de último nivel escale por un factor x0.5. Isto require aumentar o tamaño da cache de último nivel nun factor $x2.90=1/(0.5)^{1/0.65}$, no que supoñemos p=0.65, típico para as cargas de traballo dos Itanium 2. Como podemos apreciar nos datos facilitados, neste cambio de tecnoloxía, a memoria cache escalou por un factor x3.0. Como vemos na Figura 14, con estes factores de escalamento, o ancho de banda requirido (BW) non escala. Isto coincide co non escalado do ancho de banda nominal da interconexión á memoria. Polo tanto estas ecuacións permiten obter en primeira orde os factores de escalamento dunha xeración de microprocesadores á seguinte, facendo suposicións razoables e polo tanto poden resultar útiles para entender as tendencias futuras (dentro das limitacións obvias dun modelo tan sinxelo).

4.2 CPI vs. Mr

Resulta moi instructivo facer unha representación gráfica do CPI fronte a taxa de fallos cache de último nivel (Mr). Para facer esta representación deben terse en conta dúas cotas inforiores do CPI:

- Cota inferior cando Mr tende a cero e o paralelismo é máximo: neste caso CPI_{core} tende a CPI_{corei} , e G (ganancia en velocidade por utilizar N núcleos) tende a N. Polo tanto para este caso temos que $CPI > CPI_{corei} / N$.

- Conta inferior cando Mr é moi grande: neste caso pode chegarse ao punto de saturación do ancho de banda do sistema, é dicir cando o ancho de banda demandado BW tende ao ancho de banda máximo sostible (BWM). Polo tanto unha cota inferior para este caso é CPI > k x L x Mr x F / BWM.

A Figura 20 ilustra a gráfica de CPI vs Mr, na que se incorporaron as dúas cotas inferiores que acabamos de mencionar. Cando Mr é elevado, entón teremos moitos requirimentos á memoria externa é polo tanto os efectos de cola serán importantes, é dicir, o parámetro QE será moi significativo. Neste caso é o ancho de banda do sistema o que limita as prestacións. Notar que a pendente da cota inferior do CPI para Mr grande está determinada polo inverso do ancho de banda máximo sostible. Polo que canto máis grande sexa o ancho de banda do sistema, menor pendente terá a cota inferior do CPI. Dicir que o ancho de banda máximo sostible (BWM) está directamente correlacionado co ancho de banda nominal da interconexión de memoria (BWN), de tal xeito que BWM é típicamente un 20-30% inferior.

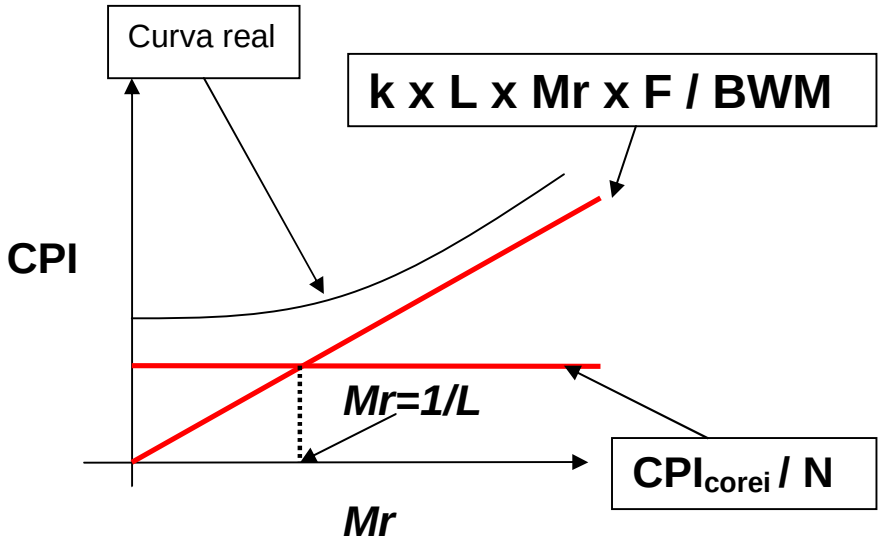


Figura 20: Representación do CPI vs Mr.

O factor L x Mr é moi importante no deseño de microprocesadores. Este factor ten unidades de bytes/instrución (=bytes/fallo x fallos/instrución), e o seu inverso coñécese coma intensidade operacional (instrucións/byte). Nas cargas de traballo cunha forte compoñente de computación en punto flotante, para simplificar, no canto de instrucións utilizan Flops (instrucións de punto flotante). É dicir, nese caso, consideran a cota inferior do CPI como o inverso do número máximo de instrucións de punto flotante que se poden executar por ciclo. Unha regra tradicional no deseño de microprocesadores indica que se debe dimensionar un ancho de banda para unha intensidade operacional de 1 Flop/byte (2 Flop/byte para arquitecturas recentes xa que soporta a instrución FMA que realiza a operación A x B + C, é dicir dúas operacións de punto flotante por instrución) tomada para o punto no que ambas cotas inferiores do CPI coinciden (cóbado da curva). A Figura 21 amosa o número de bytes/Flop para as arquitecturas de Intel nos últimos anos. Como vemos o valor está próximo a 0.5, xa que nestas arquitecturas utilízase a instrución FMA.

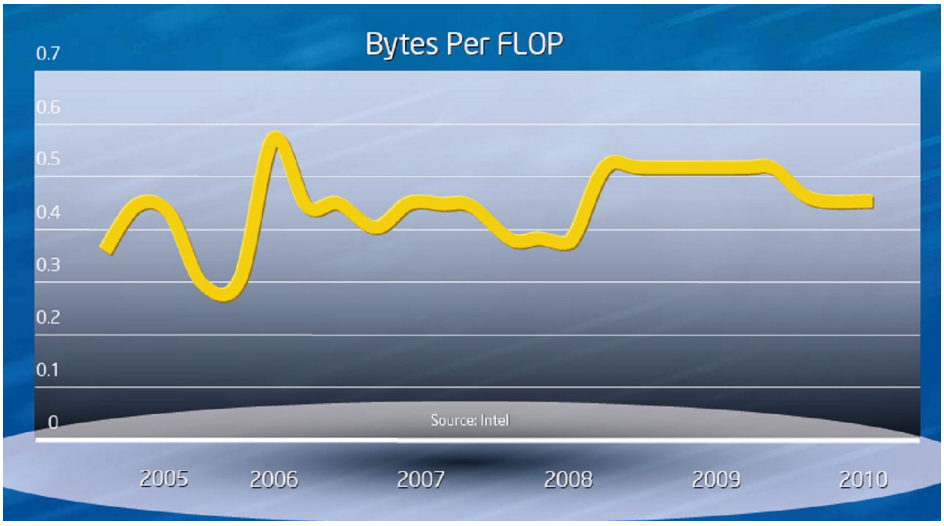


Figura 21: Bytes por Flop para as arquitecturas de Intel nos últimos cinco anos.

Polo tanto, igualando as dúas cotas (facémolo en térmos do CPI no canto de Flops) e facendo 1/(L x Mr) =1 (intensidade operacional igual a 1) obtemos o seguinte valor para o ancho de banda:

$$BWM = k \times N \times F / CPI_{corei}$$

Isto equivale a dimensionar o ancho de banda para o caso no que a taxa de fallos da cache de último nivel é de Mr=1/L, e supoñendo paralelismo total (G=N). Obviamente, cada programa que se execute no microprocesador terá os seus valores característicos de Mr e G, o que dará lugar a diferentes valores de CPI segundo a carga de traballo. Esta regra indica que o ancho de banda debe escalar coa frecuencia, o número de núcleos e o número de instrucións por ciclo (inverso do CPI) máximo que poden acadar os núcleos. A industria non sempre sigue esta regra, xa que redeseñar a interconexión co sistema de memoria en cada xeración resulta moi custoso en termos de deseño, verificación e compatibilidade co resto de elementos do sistema. Unha práctica habitual é manter a mesma interface coa memoria durante máis dunha xeración. Na Figura 22 podemos observar o escalado histórico do ancho de banda de memoria en microprocesadores de Intel. Como vemos, nalgunhas transicións o ancho de banda non escala, e noutras prodúcese un salto moi significativo. O efecto global equivale a un ritmo de escalar x2 o ancho de banda cada dous anos.

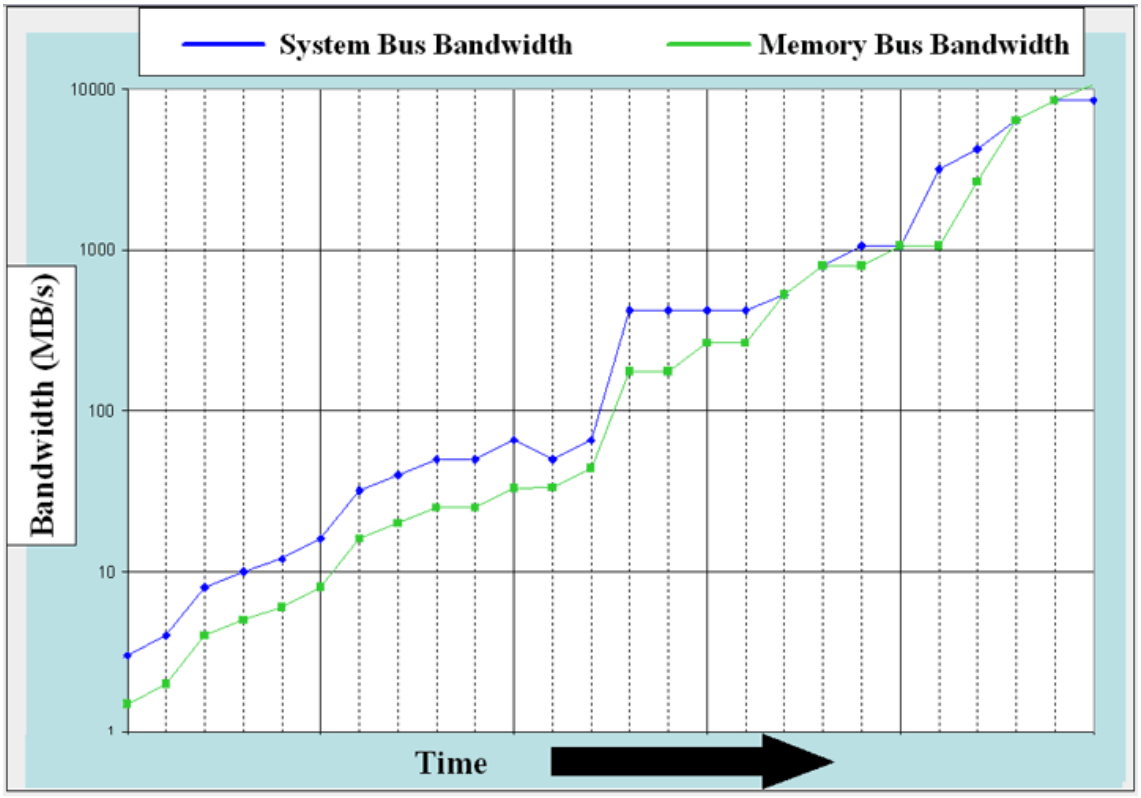


Figura 22: Tendencia histórica no escalamento do ancho de banda de memoria nos procesadores de Intel.

5- Final dos Procesadores cun só Núcleo

Até o 2005 o modelo predominante de microprocesador foi o que contiña un só núcleo de procesamento superescalar. Este modelo escalaba por un factor x0.5 a latencia dunha xeración de microprocesadores á seguinte con diferentes estratexias de escalamento. Unha destas estratexias, a cal se axustan bastante ben os microprocesadores de Intel amósase na Figura 23.

Escalado para Latencia x 0.5: era unicore
N=1
G=1
F x 2 (FO4 x 0.7 →novo proceso; TFO4 x 0.7 →
pipeline mais profundo)

CPI=x 1 con F x2 → LM x 2 (#ciclos) x 0.85 (mellora da latencia de memoria)

→Scache x 2.3 -> Acache x 1.15=2.3 x 0.5

BWM → F x2 / CPI x1

Acore x 1.4 ?? (CPI=cte)

Achip x 1.2-1.4

Pot x2 (Frec) x 1.4 (Achip) x 0.85^2 (Vdd)=x2

Figura 23: Escalamento na era de microprocesadores cun so núcleo.

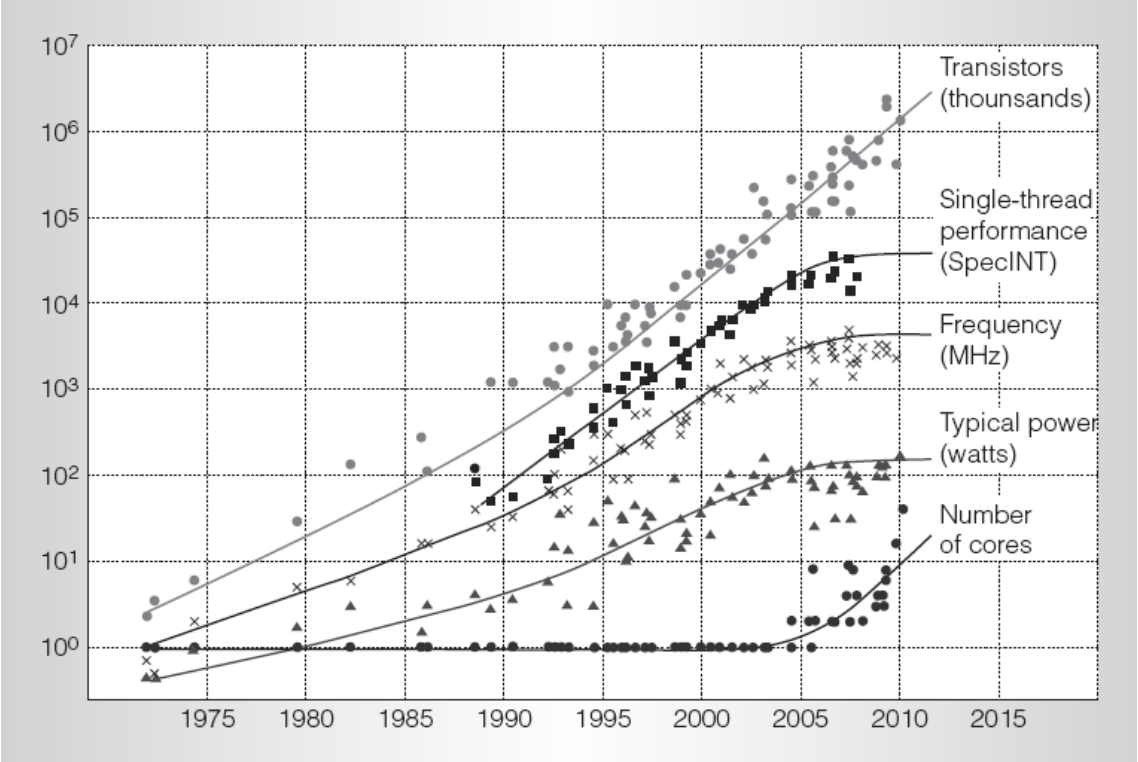


Figura 24: Evolución dos microprocesadores dende 1970 até 2010.

A Figura 24 ilustra a evolución dos microprocesadores dende 1970 até 2010. Vemos como a gráfica confirma este modelo de escalamento, polo menos dende principios dos 90 até o 2005 (ver o escalamento da frecuencia, as prestacións para un só fio, e a potencia).

O modelo de escalamento utilizado consistiu en dobrar a frecuencia dunha xeración a outra mantendo o CPI constante. Para dobrar a frecuencia, aproveitouse o escalamento na velocidade dos circuítos (por un factor 1/0.7) e o aumento de etapas do pipeline, reducindo o número de retardos FO4 por ciclo de reloxio (escalando TFO4 por x0.7). Respecto da parte de retardo por esperas dos datos de memoria principal, supoñemos un factor de escalamento de x0.85 da latencia absoluta de memoria (a latencia de memoria escalou pouco). Ao aumentar a frecuencia por un factor x2.0, o número de ciclos para acceso á memoria aumenta nun valor x1.7=2.0 x 0.85. Para manter o CPI=1, é necesario aumentar o tamaño de memoria cache para compensar o factor x1.7 de incremento de número de ciclos de latencia de memoria. Isto conséguese aumentando a memoria cache por un factor x2.3 (supoñendo p=0.65). Outra cuestión a destacar é que o CPI_{corei} non debe reducirse malia ter aumentado a profundidade do pipeline. Isto resulta complicado, e require aumentar o costo hardware (supoñemos que se require un aumenta dun factor 1.4 da área do núcleo). A área dedicada a cache aumenta por un factor 1.15=2.3 (factor de aumento do tamaño de memoria cache) x 0.5 (factor de redución de área por usar un proceso novo). Como en esa época a maior parte da área estaba ocupada polo núcleo de procesamento, a área do procesador aumentaba por un factor entre x1.2-x1.4.

Este modelo de escalamento pode variar se analizamos instancias concretas ao longo do tempo, sen embargo podemos dicir que esta é a tendencia xeral, sobre todo nas últimas

xeracións de microprocesadores con un só núcleo. Na Figura 25 amósase a evolución do parámetro TFO4 (número de retardos de FO4 por etapa do pipeline) ao longo do tempo para diferentes familias de microprocesadores. Cunha liña marcamos a tendencia de microprocesadores Intel de alta frecuencia (tipo Pentium). Como vemos o escalamento do parámetro TFO4 é evidente.

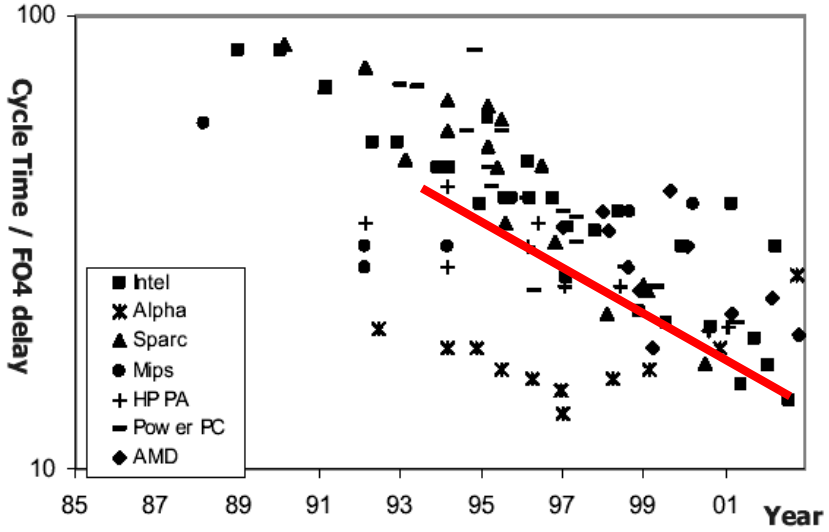


Figura 25: Evolución do parámetro TFO4 ao longo do tempo para diferentes familias de microprocesadores con un só núcleo.

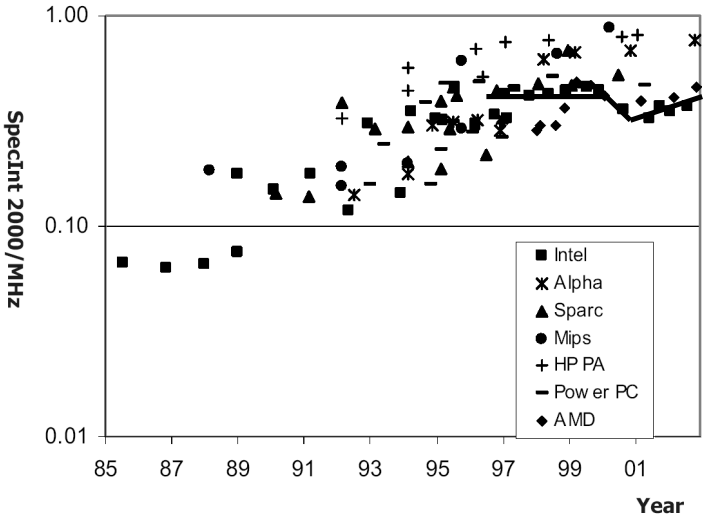


Figura 26: Evolución da ratio SpecInt/Mhz (proporcional ao inverso do CPI) ao longo do tempo para diferentes familias de microprocesadores.

Na Figura 26 amósase a evolución do parámetro SpecInt/Mhz (SpectInt é unha medida das prestacións dun microprocesador en relación a un sistema de referencia e para un

conxunto de programas de proba). Este parámetro é inversamente proporcional ao CPI. Como vemos para os microprocesadores de Intel de alta frecuencia, a tendencia (marcada cunha liña na figura) foi a de intentar manter constante o CPI. Vemos incluso que nas últimas xeracións o CPI descendeu, debido á profundidade tan elevada do pipeline.

Por qué se esgotou este modelo de escalamento?.

Vexamos as razóns polas que este modelo de escalar as prestacións con arquitecturas convencionais non atopa saída. A Figura 18 ilustra o espazo de posibilidades para escalar. Por un lado, logo de ver os datos da Figura 25, xa se ten chegado preto do límite práctico do parámetro TFO4 (8-10 FO4). Polo tanto esa fonte de escalamento desaparece. A frecuencia só podería escalar por x1/0.7 (escalamento do valor FO4 coa tecnoloxía).

Sen embargo aparecen diferentes circunstancias que fan que todo sexa moito peor. O principal problema son as interconexións. Dunha xeración a outra a área dos compoñentes redúcese a metade, e as conexións dentro destes compoñentes escalan linealmente. Sen embargo debido a utilización de caches de maior tamaño e outras estruturas, aparecen conexións que non reducen a súa lonxitude. Na Figura 27 amósase o retardo de dúas conexións de 5mm (lonxitude cte) co escalamento da tecnoloxía. Vemos como o retardo medido en número de retardos FO4 aumenta de xeito exponencial. Isto implica que cada vez serán necesarios máis ciclos de reloxo para transmitir un sinal por dita conexión. Isto leva a que aumente a profundidade do pipeline, segmentando as propias conexións longas con rexistros, coas implicacións negativas que iso supón no CPI (este efecto está ilustrado na Figura 18).

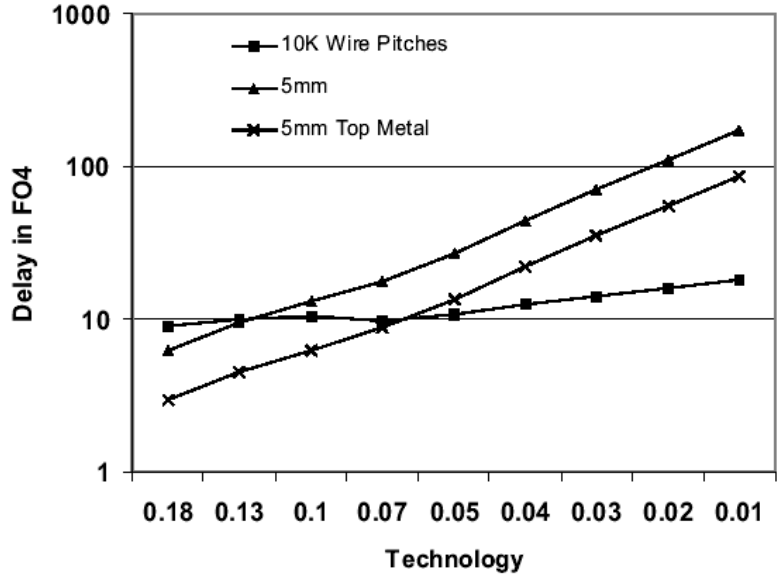


Figura 27: Evolución coa tecnoloxía do retardo de interconexións longas (en FO4).

Polo tanto, neste límite do parámetro TFO4 preto de 10, só se escala x0.7 o tempo de ciclo debido ao escalamento da tecnoloxía. Pero o aumento da profundidade do pipeline debido as conexións longas que non escalan degrada o CPI, polo que son necesarias

máis e mellores estruturas para manter o CPI, por exemplo unha mellor predición de saltos, etc. Sen embargo a introdución desta complexidade hardware fai que aparezan conexións máis longas e aumente a profundidade do pipeline, pechando un círculo sen saída posible para obter o escalamento desexado.

O certo e que o CPI degrádase (aumenta) e como resultado o escalamento en velocidade xa non chega nin a x1.5. Por outra banda o propio modelo de programación limita o mínimo CPI que se pode acadar (límite ILP na Figura 18). Polo tanto, aumentar o parámetro TFO4 para reducir a profundidade do pipeline, e intentar diminuír o CPI sempre estará limitado polo modelo de programación, e non é unha estratexia viable.

As melloras nos compiladores para aumentar o CPI, prebusca de datos e instrucións para diminuír a taxa de fallos, execución multifío hardware para ocultar as paradas do procesador, etc, poden atrasar en menor ou maior grao estes efectos, pero o modelo de escalamento do microprocesador cun só núcleo quedou esgotado.

6- Arquitecturas alternativas segundo o tipo de carga de traballo

Como xa mencionamos anteriormente a alternativa que actualmente está a seguir a industria é a de incorporar varios núcleos de procesamento nun chip (os chamados en inglés ñmulticoreò). Para resolver un determinado problema as aplicacións organízanse en fíos e execútanse nos diferentes procesadores do chip en paralelo. Ademais cada procesador pode conmutar entre un determinado número de fíos, mellorando a produtividade cando o thread activo queda bloqueado. Noutras implementacións faise unha execución multifío simultáneo, é dicir o procesador ten activos máis dun fío, e as instrucións dos diferentes fíos mestúranse para acadar unha maior produtividade.

[ñLatency Computingò]. O tipo de carga de traballo dos procesadores determina en grande medida a súa arquitectura. Por exemplo, o tipo de procesador Itanium Multicore (polo momento con catro núcleos como máximo) caracterízase por núcleos complexos, que soportan dous fíos hardware, e unha elevada cantidade de memoria cache. Isto está determinado polo tipo de cargas de traballo para o que está deseñado: aplicacións de cómputo intensivo de propósito xeral con paralelismo limitado (isto é o que se coñece en inglés como ñlatency computingò). Neste tipo de cargas de traballo non é doado atopar moitos threads que poidan executarse en paralelo, polo que é importante que a latencia de execución de cada fío sexa baixa (por iso os núcleos son complexos e só soportan dous fíos hardware).

Na Figura 28, amósase outro exemplo de procesador orientado a este tipo de cargas de traballo, o procesador IBM Power 7+. Este procesador está fabricado en tecnoloxía de 32nm (ocupa uns 570mm^2), consta de 8 núcleos, con caches privadas nos niveis 1 e 2, e unha cache de nivel 3 compartida entre todos os núcleos (80Mbytes de cache de nivel 3 utilizando eDRAM ï DRAM integrable no chip do microprocesador- que é 3-4x veces máis densa que a SRAM convencional). Cada núcleo soporta 4 fíos hardware, polo que o número total de fíos soportado é de 32. O procesador incorpora dous controladores de memoria dentro do propio chip. Cada núcleo conta con 12 unidades de execución (4 unidades de punto flotante). O procesador acada até 5.0GHz de frecuencia de reloxo.

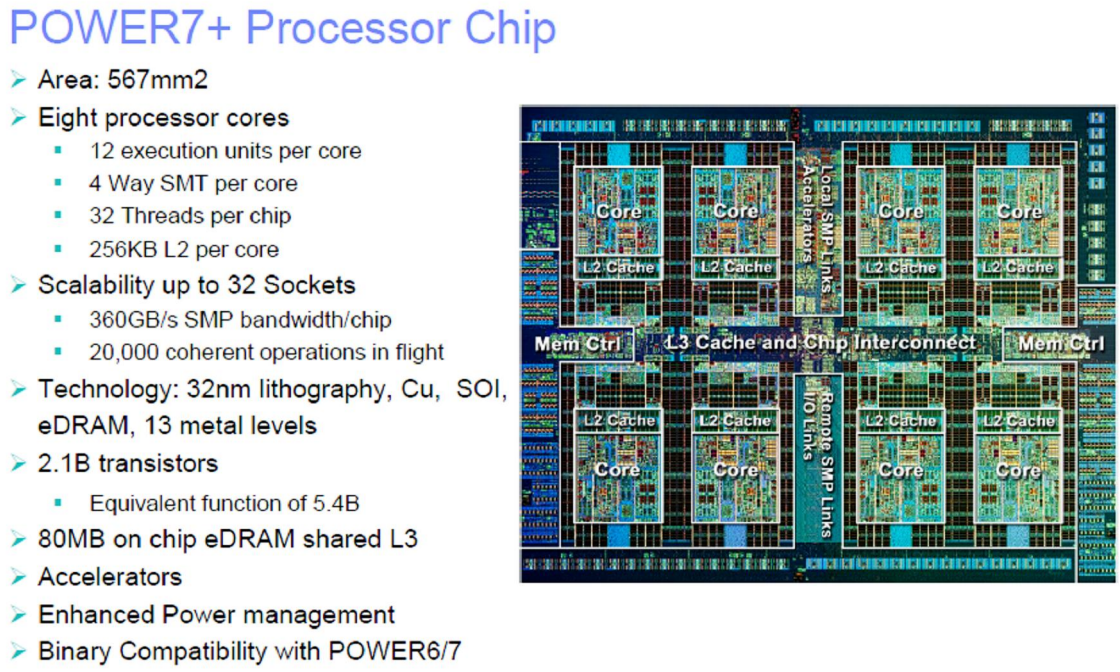


Figura 28: Características do procesador IBM Power 7+.

[ñThroughput Computingò]. Outro tipo de cargas de traballo están máis orientadas a servizos de transaccións independentes para grandes servidores (un exemplo sería un complexo sistema de reservas). Neste caso a carga de traballo consta dun elevado número de fíos, en moitos casos independentes (transaccións de diferentes clientes), e que fan pouca reutilización de datos (non son tan dependentes do tamaño das caches). Neste caso a latencia por fío e menos relevante que a taxa de fíos que se completa por unidade de tempo (isto é o que se coñece en inglés como ñthroughput computingò). O procesador Rock, de Sun Microsystems (agora Oracle) constitúe un exemplo de arquitectura que responde a este tipo de cargas de traballo.

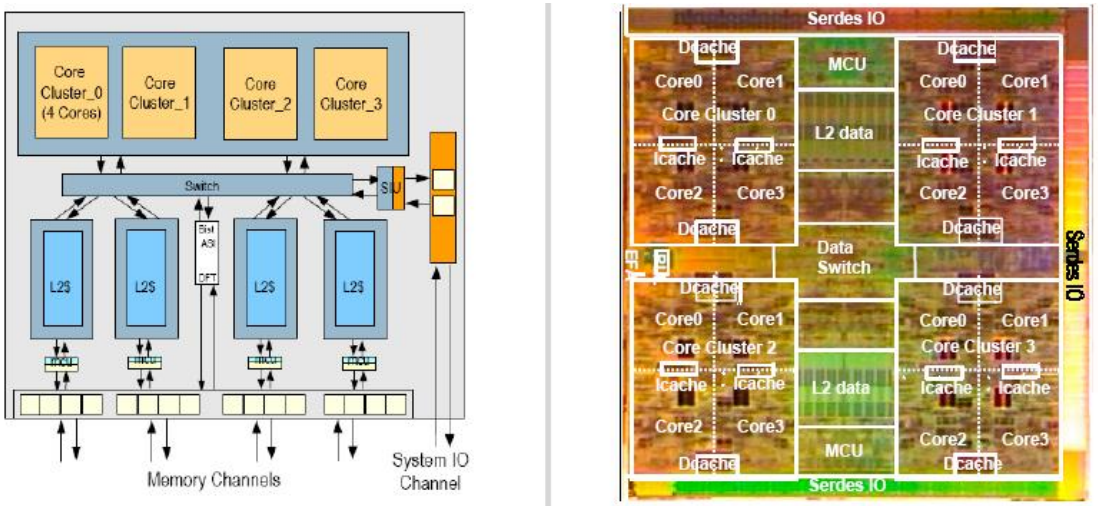


Figura 29: Arquitectura do procesador Rock de Sun Microsystems.

Na Figura 29 amósase a arquitectura do procesador Rock. En total consta de 16 núcleos de procesamento sinxelos (cada un ocupa uns 14 mm^2 nun proceso de 65 nm, fronte aos 27 mm^2 dos núcleos da familia Intel Itanium escalado ao mesmo proceso), onde cada núcleo soporta catro fíos hardware (en total o chip soporta 64 fíos). A cache de último nivel é de 2MB (compartida por todos os núcleos), como vemos moi inferior aos 24MB do Itanium 2 en 65nm. O chip ocupa case 400 mm^2, lonxe dos 700 mm^2 que require o Itanium 2 en 65nm. Polo tanto vemos como o tipo de carga de traballo determina en gran medida a arquitectura do procesador.

[Paralelismo de datos]. Outro tipo de aplicacións están estruturadas en tarefas relativamente simples (ñkernelsò) sobre un elevado número de datos, cun elevado potencial de procesamento paralelo. Un exemplo claro son as aplicacións gráficas, realidade virtual e multimedia. Neste caso é doado atopar fíos que poden operar en paralelo e a limitación está no procesamento dos datos (tanto na realización das operacións aritméticas coma na provisión dos datos ás unidades de procesamento).

Unha primeira aproximación aos procesadores que explotan o paralelismo de datos constitúena os **procesadores vectoriais**. Esta idea non é nova (os supercomputadores Cray utilizaban nos anos 70 esta forma de procesamento como elemento principal, e o EarthSimulator, un supercomputador instalado en Xapón que ten sido o supercomputador máis rápido do mundo fai uns poucos anos, utiliza uns 5000 procesadores vectoriais), non obstante, cos cambios tecnolóxicos e as novas xeracións de compiladores poden ser unha alternativa de moito interese.

Os procesadores vectoriais incorporan instrucións sobre vectores tal e como se mostra na Figura 30 para o caso dunha suma. Isto implica que unha soa instrución da lugar a un elevado número de computacións. Polo tanto increméntase o paralelismo e por conseguinte a velocidade.

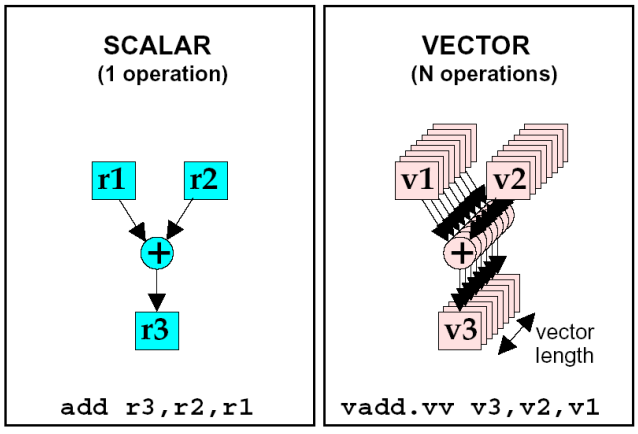


Figura 30: Procesamento vectorial vs procesamento escalar.

A programación de procesadores vectoriais xa é un campo ben coñecido en supercomputación (por exemplo HPF, High-Performance Fortran baséase neste tipo de paradigma). Para a programación utilízanse directivas vectoriais no código de alto

nivel. Os compiladores vectorizan o código detectando operacións vectoriais (análese de dependencias entre iteración nos bucles), transforman bucles para que sexan vectorizabeis, etc.

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
      Y[i] += M[i][j] * V[j]
```

Figura 31: Código vectorizable.

No exemplo de código da Figura 31, temos dous bucles aniñados (en i e j). É posíbel vectorizar o código de dous xeitos. Podemos vectorizar o bucle interno en j, facendo N multiplicacións vectoriais cada unha do vector M[i] polo vector V. O vector resultante é reducido mediante unha operación escalar de suma das súas compoñentes (N-1 sumas escalares). O acceso á memoria dos elementos da matriz é por filas (vectores con elementos contiguos).

Outra alternativa sería vectorizar primeiro o lazo i (externo). Neste caso faríanse N multiplicacións vectoriais fa fila da matriz M[j] polo elemento j do vector V. Resultarían N vectores (cada vector sería a columna dunha matriz). O cálculo remata cunha segunda vectorización, o lazo j, con N-1 sumas vectoriais (suma das columnas). O acceso aos vectores é descontinuo xa que se accede a M por columnas.

```
for (i=0; i<N; i++) {
  if (A[i]<threshold) C[i]=A[i];
  else C[i]=B[i];
}
```

```
vld       a, a_address     # Vector load from array A
vld       b, b_address     # Vector load from array B
vcmp.lt   t, a, threshold  # Vector comparison; element i in flag
                           # register t is set if A[i]<threshold
vneg      f, t             # Negate the result of the comparison and
                           # store it in flag register f
vst       a, c_address, t  # Vector store under mask t to array C
vst       b, c_address, f  # Vector store under mask f to array C
```

Figura 32: Vectorización de código.

Na Figura 32 amósase a vectorización dun código que enche un vector con elementos de dous vectores dependendo de se o valor de cada compoñente do vector A é inferior ou non a un determinado valor. Este tipo de bucles con condicións pódense vectorizar ca

axuda dos ñflags registersò. Estes rexistros almacenan información que permite enmascarar operacións vectoriais sobre un determinado compoñente do vector.

No código adxunto, cárganse os vectores A e B nos rexistros vectoriais a e b. Faise unha comparación vectorial (elemento a elemento) co valor ñthresholdò, e o resultado almacénase no rexistro vectorial t (o elemento i de t estará a 1 se A[i]<threshold). Na seguinte operación négase o contido do rexistro t (o que está a 1 a pasa a cero, e o que está a cero pasa a 1) e almacénase no rexistro f. Finalmente faise un almacenamento condicional de vectores, almacénase o vector a na dirección de memoria de C pero ca máscara t (só se almacenarán as posicións de a que conteñan un 1 no vector máscara t). O mesmo co vector b.

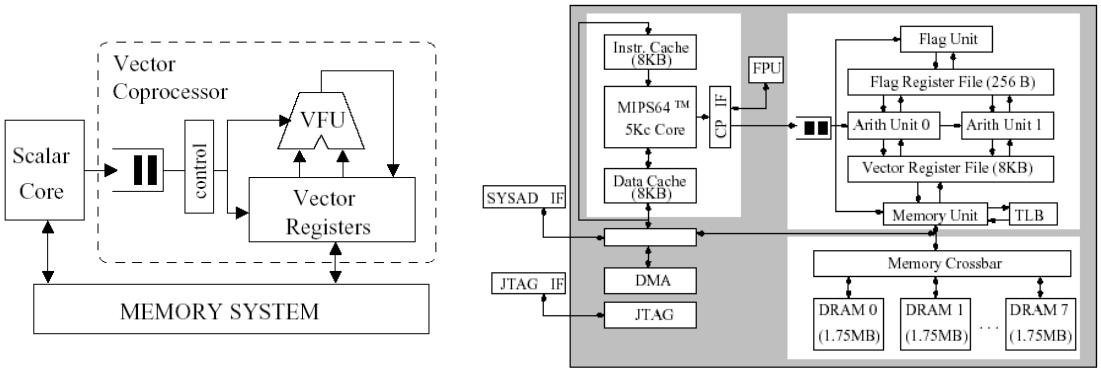


Figura 33: Arquitectura dun procesador vectorial.

Na Figura 33 vemos un exemplo de arquitectura dun microprocesador vectorial. Consta dun procesador escalar convencional que executa o sistema operativo e a aplicación. En canto se atopan directivas vectoriais (as de maior carga computacional), estas envíanse ao coprocesador vectorial. Este coprocesador consta de bancos de rexistros vectoriais para almacenar os vectores orixe, vectores obtidos como resultados intermedios e vectores finais; unidades aritméticas segmentadas, e unidades de load/store vectorial para transferir vectores entre o banco de rexistros vectorial e a memoria principal.

Os procesadores vectoriais necesitan un elevado ancho de banda entre a memoria e o banco de rexistros. A velocidade a que se procesan os vectores está dada por Nu (número de unidade aritméticas) x Frec (frecuencia) (xa que se os vectores son longos, as unidades funcionais estarán a máxima produción). Isto implica que o ancho de banda total de datos no caso extremo debe ser moi elevado: Nu x Frec x nº_bytes_datos/operación_aritmética.

Por exemplo, supoñamos dúas unidades aritméticas (Nu=2) operando a unha frecuencia de 1GHz e implementando as dúas as operacións vectoriais D=A+B*C, onde A, B e C e D son vectores longos con elementos en dobre precisión (8 bytes). Para facer esta operación sen paradas das unidades aritméticas requírese un ancho de banda coas unidades de 2 x 1GHz x (4 x 8) = 64 GB/sec. A memoria debería aportar unha certa fracción deste ancho de banda, dependendo do tamaño dos bancos rexistros, caches que se incorporen e reutilización dos vectores.

Este modelo de procesamento permite reducir drasticamente o hardware necesario para atopar paralelismo a nivel de instrucción, e introducir así máis unidades de procesamento de datos. Isto permite acadar CPIs moi reducidos, sobre todo para vectores de tamaño longo. Un aspecto importante das arquitecturas vectoriais é que reducen o efecto da longa latencia da memoria. Nas arquitecturas superescalares convencionais a diferenza entre a latencia entre emisión de instrucións e a latencia da memoria é moi elevada. Sen embargo, nos procesadores vectoriais a latencia de emisión multiplícase pola lonxitude dos vectores (unha instrucción vectorial implica moitas instrucións escalares) e redúcese moito a diferenza ca latencia da memoria, o cal fai que sexa menos probábel ter paradas dos pipelines por causa da latencia de memoria.

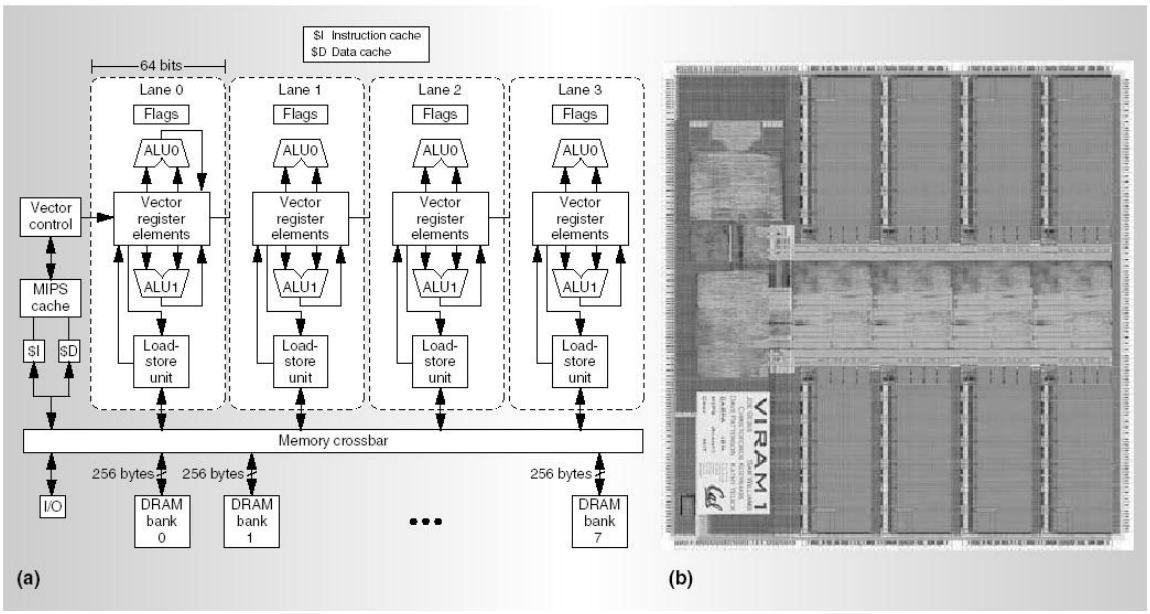


Figura 34: Arquitectura dun microprocesador vectorial coa memoria principal integrada no mesmo chip.

Na Figura 34 vemos unha implementación dun microprocesador vectorial (grupo de Berkeley, USA). Consta de catro liñas de procesamento, contando cada unha con dúas unidades aritméticas segmentadas, rexistros para vectores (32 vectores de 32 elementos de 8 bytes cada un), rexistros para máscaras (ñflagsò), e unha unidade para carga/almacenamento de vectores dende a memoria principal.

A característica fundamental desta implementación é que a memoria principal está dentro do chip, e non ten memoria cache. As tecnoloxías de implementación das memoria DRAM e dos procesadores so lixeiramente diferentes xa que buscan obxectivos distintos. Neste caso utilizouse un proceso de fabricación para memorias que permite integrar elementos de procesamento. A desvantaxe é que os elementos de procesamento son máis lentos que no caso dun proceso típico para implementar microprocesadores. Sen embargo integrando memoria e unidades de procesamento no mesmo chip, conséguese un elevado ancho de banda coa memoria e baixa latencia de acceso, xa que todo está dentro do chip.

Polo tanto os procesadores vectoriais deste tipo presentan vantaxes evidentes, pero están condicionados a que nas aplicacións que se queren resolver se poida utilizar o modelo de programación vectorial. No eido das simulacións científicas e computación gráfica isto parece bastante xustificado xa que as operacións sobre grandes vectores son moi frecuentes.

Os **microprocesadores baseados en ñstreamingò** estenden os conceptos dos microprocesadores vectoriais. No canto de vectores, a información está organizada en strings (vectores de estruturas). No canto de instrucións vectoriais simples (a mesma operación sobre cada elemento do vector), utilízase un ñkernelò (conxunto de instrucións) que actúa sobre cada un dos elementos do string (independentemente).

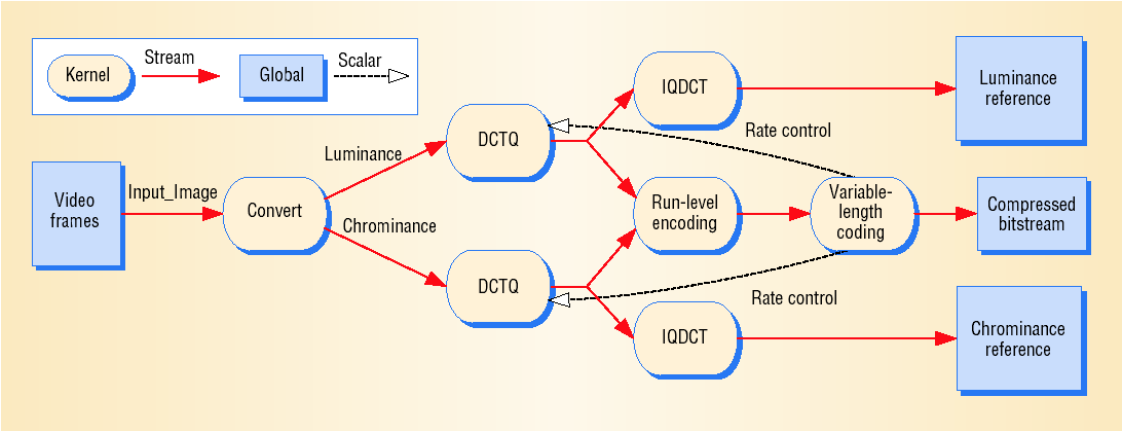


Figura 35: Modelo de computación baseado en ñstreamsò.

Na Figura 35 podemos ver un exemplo do concepto de aplicación da computación streaming. Este exemplo corresponde á parte dunha aplicación que realiza compresión MPEG2. Un stream de datos de entrada (unha imaxe, onde cada elemento do stream son as compoñentes rgb dos pixels da imaxe) é procesado por un kernel para producir dous streams de saída (luminancia e crominancia). Estes streams son procesados por dous ñkernlelsò producindo máis streams intermedios e que son procesados por sucesivos ñkernelsò.

Este modelo de programación presenta todas as vantaxes en canto a paralelismo que ten o modelo vectorial, pero presenta moitas máis oportunidades para conseguir paralelismo xa que os datos son vectores de estruturas complexas e os kernels poden facer operacións complexas sobre os elementos dos streams.

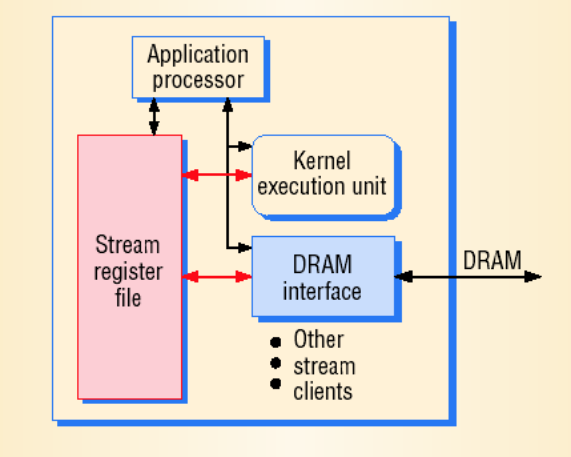


Figura 36: Arquitectura de alto nivel dun procesador de streams.

Na Figura 36 vemos unha implementación de alto nivel dun procesador para streaming. Consta dun procesador superescalar convencional para executar o sistema operativo e aplicación principal. As instrucións tipo streaming son enviadas ao coprocesador. O coprocesador consta dun banco de rexistros para streams, e varios clientes para streams: unidades de execución de kernels, interface ca memoria principal, etc.

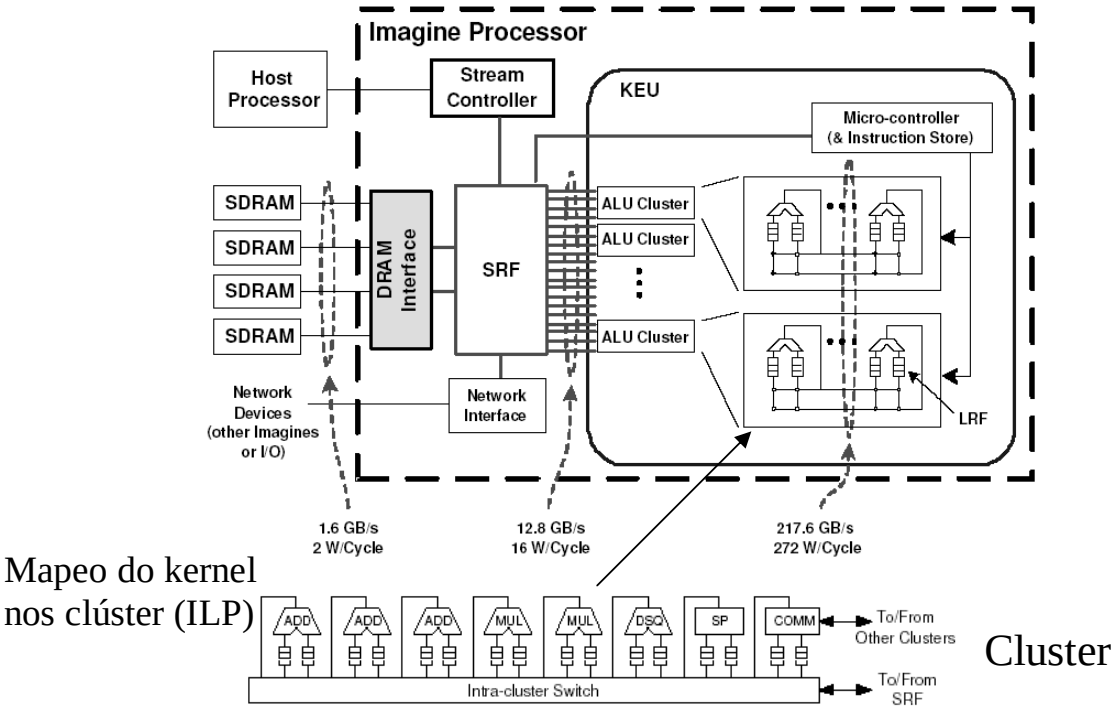


Figura 37: Arquitectura detallada dun procesador streaming.

Na Figura 37 vemos máis en detalle a estrutura do procesador. A unidade para execución do kernel consta dun conxunto de clúster de unidades aritméticas. Cada clúster consta dun certo número de unidades aritméticas e rexistros para resultados

intermedios. Nesta implementación todos os clúster realizan o mesmo kernel, polo que todos comparten un mesmo microcontrolador para o control das instrucións a executar. Os streams repártense entre os clústers e estes mapean o kernel buscando o máximo paralelismo a nivel de instrucción na execución de dito kernel.

É importante ver a xerarquía de anchos de banda presente no procesador. Dentro dos clúster o ancho de banda é de 272 palabras/ciclo, pasa a 16 palabras/ciclo para a comunicación entre o banco de rexistros para streams e os clúster, e 2 palabras/ciclo entre o banco de rexistros para streams e a memoria principal. A comunicación entre o banco de rexistros e os clúster só se leva a cabo despois de executar un kernel. A comunicación entre a memoria principal e o banco de rexistros de streams só se leva a cabo ao final da computación streaming ou se o banco de rexistros non pode acoller todos os streams involucrados na computación. Deste xeito proporcionase o ancho de banda axeitado a cada parte do procesador.

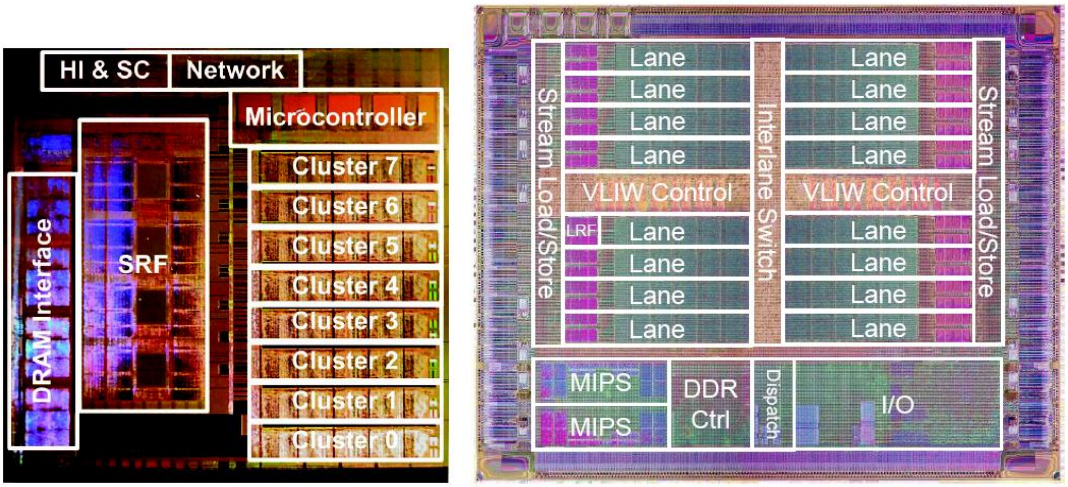


Figura 38: Procesador Imagine (esquerda) deseñado na Universidade de Stanford, e o procesador Storm-1 (dereita) comercializado pola empresa Stream Processors Inc.

Na Figura 38 vemos un chip que implementa un procesador streaming (Imagine) con 8 clúster de computación. Resulta interesante comparar a estrutura do chip ca dun procesador máis convencional (por exemplo Itanium). Vemos como neste caso as unidades de procesamento ocupan unha parte moi relevante do chip. Ademais non existe memoria cache. Se co modelo de programación se consegue manter as unidades aritméticas cun alto grado de utilización, o resultado é un procesador extremadamente rápido e eficiente en termos da métrica GFLOPs/Watt. Na Figura 34 tamén se amosa unha versión comercial deste procesador, neste caso con 16 clústers e dous procesadores para o control da aplicación. Resulta interesante destacar que a carga dende e escritura en memoria dos streams é responsabilidade do programador. Isto complica a programación, xa que o programador ten a responsabilidade explícita de indicar os movementos de datos entre o banco de rexistros de streams e a memoria. As **tarxetas gráficas**, no seu camiño de ampliar o eido de computación no que poden operar, contan

na actualidade con moitas unidades deste tipo, ademáis do hardware especializado para taréfas de baixo nivel de procesamento gráfico. A Figura 39 ilustra un procesador gráfico de Nvidia que conta con 16 réplicas dun procesador streaming semellante ao da Figura 38 (a arquitectura específica amósase na Figura 40).

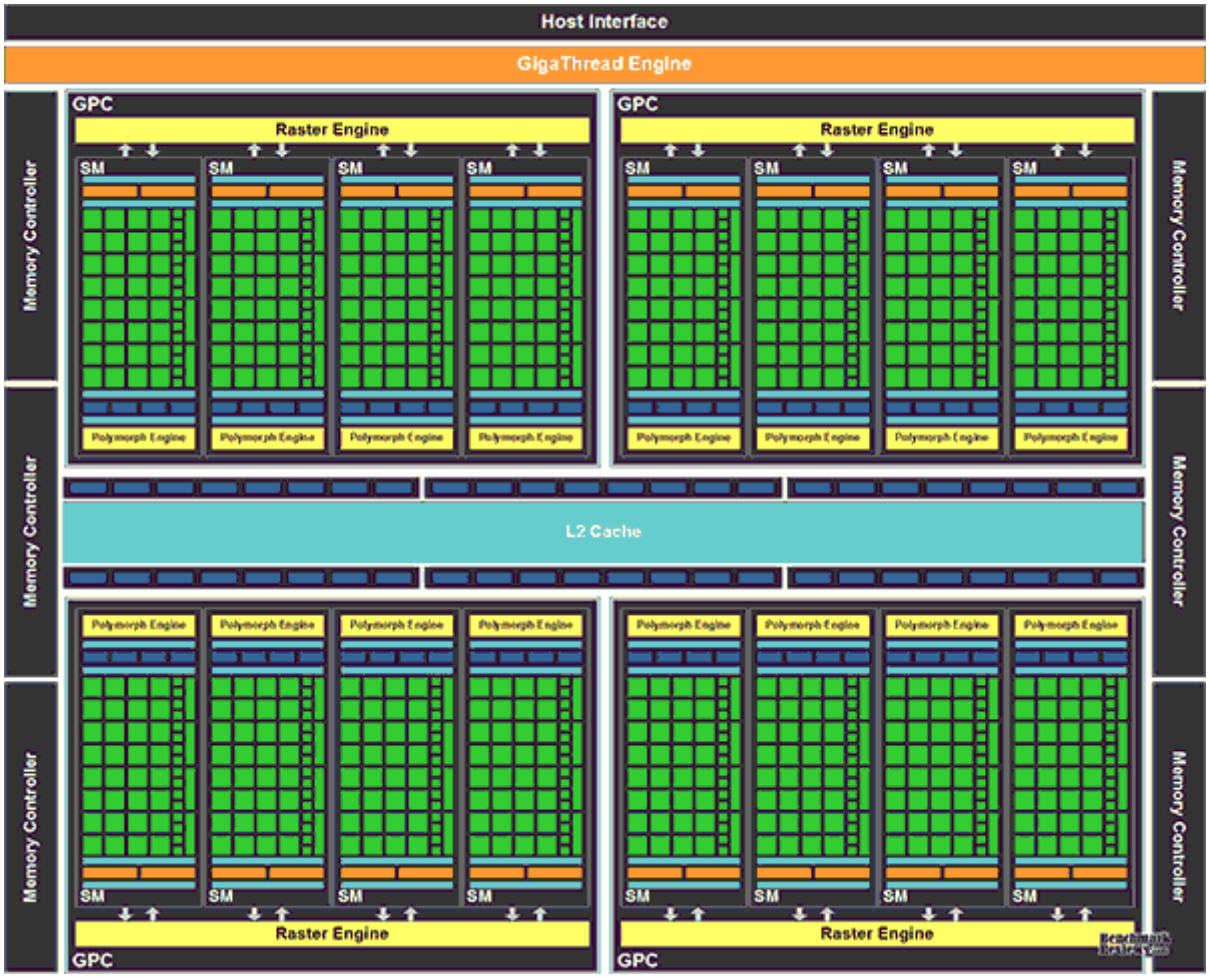


Figura 39: Arquitectura do procesador gráfico Fermi de Nvidia.

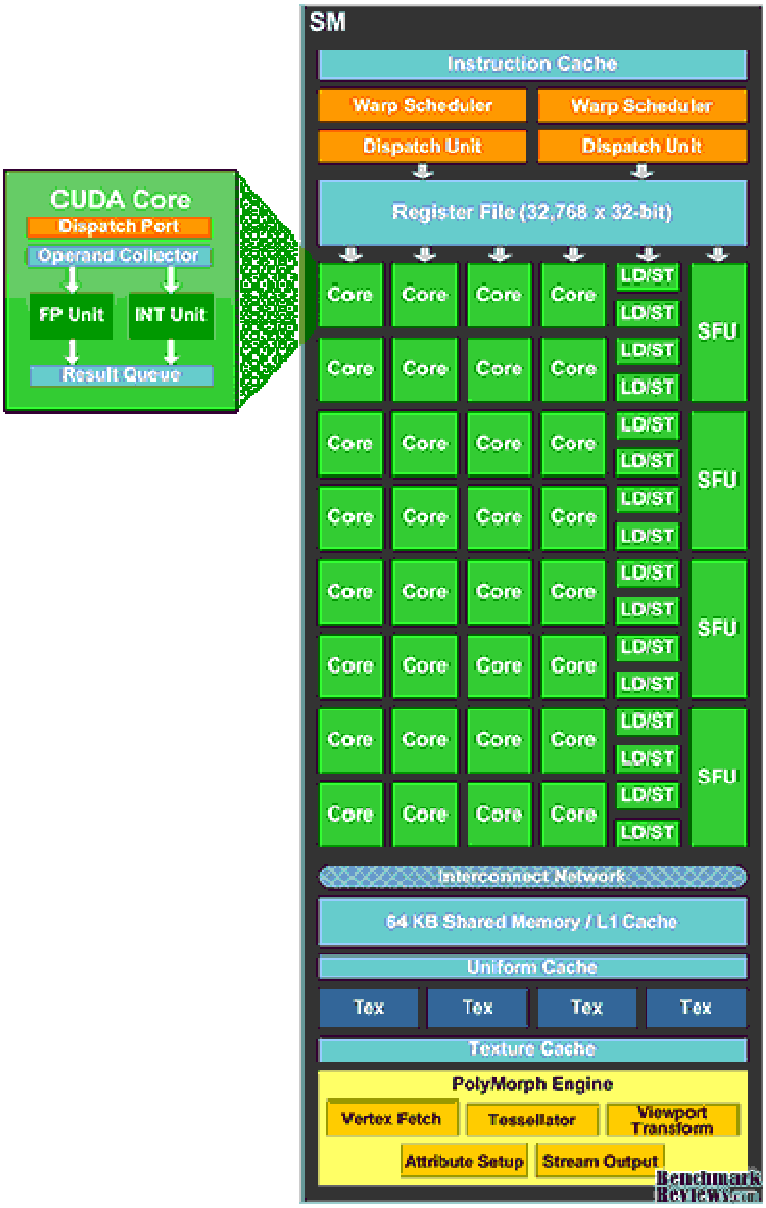


Figura 40: Vista detallada dun dos procesadores streaming do procesador gráfico da Figura 35.

Estes chips estanse a utilizar en computación de altas prestacións (supercomputación) e acadan capacidades de procesamento duns 0.7 Tflops (dobre precisión). A súa programación realízase a través de extensións coma CUDA ou OpenCL, que permiten ter acceso á programación da tarxeta gráfica. Non obstante, en xeral, a programación destes dispositivos é máis complexa que cos entornos convencionais de programación.

Na Figura 41, amósase a arquitectura do procesador Intel Xeon Phi. Este procesador está orientado ao procesamento de altas prestacións (supercomputación) para aplicacións altamente paralelas Constan dunha serie de núcleos (na versión actual até uns 60 núcleos con unha frecuencia de 1GHz) con caches privadas de nivel 1, interconectados por unha rede en anel bidireccional de 512 bits de datos por dirección, e un nivel de cache compartida entre todos.

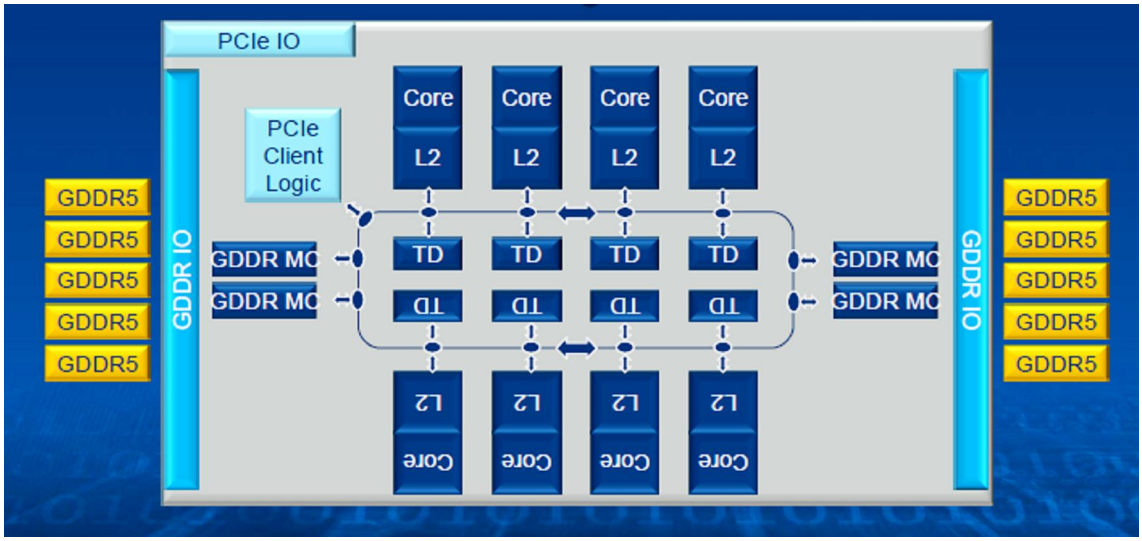


Figura 41: Arquitectura do procesador Intel Xeon Phi.

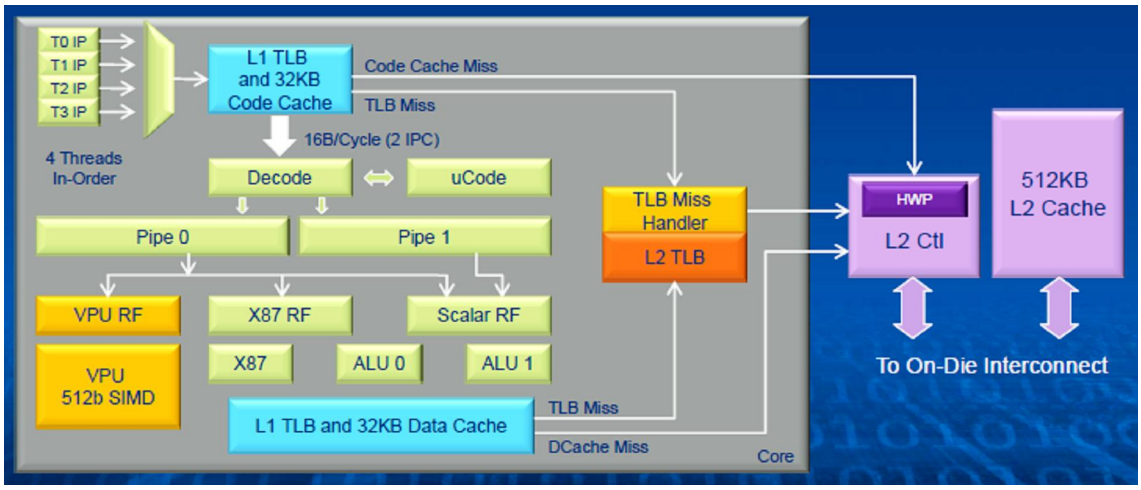


Figura 42: Arquitectura de cada un dos núcleos do procesador Intel Xeon Phi.

A Figura 42 amosa a arquitectura de cada un dos núcleos do procesador. Como vemos, consta dunha unidade escalar (procesador convencional sinxelo tipo Pentium III), e dunha unidade vectorial (de 8/16 vías para datos en dobre/simple precisión, é dicir pode operar en cada ciclo sobre 8/16 compoñentes dun mesmo vector, facendo sobre cada compoñente a operación D=AxB+C). Con isto acádase unha boa flexibilidade de programación (procesador de propósito xeral) e unha alta capacidade de computación (unidade vectorial). A potencia de cálculo actual é de 1 TeraFlop (en dobre precisión) con un consumo de 225Watts.

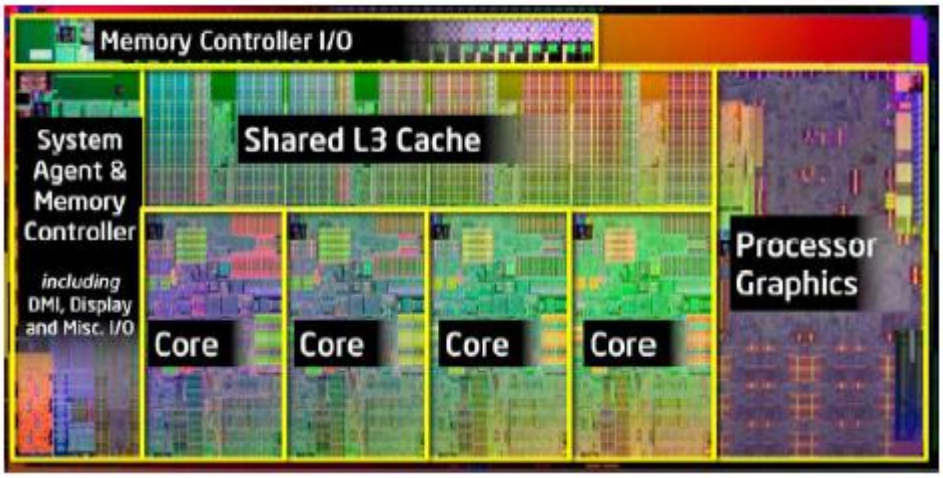


Figura 43: Procesador Sandy Bridge de Intel, con 4 núcleos e procesador gráfico integrado.

A tendencia actual para todo tipo de segmentos de mercado é a de incluír nos procesadores núcleos especializados en computación visual (para as cargas de traballo relacionadas non só con gráficos tradicionais).

Os principais fabricantes de microprocesadores teñen xa no mercado este tipo de produtos: Sandy Bridge e Ivy Bridge de Intel, ou Fusión de AMD. A Figura 43 ilustra este tipo de procesador.

Ademais, nos núcleos de procesamento xeral incorporan a posibilidade de procesamento vectorial. Actualmente é posible realizar operacións en paralelo sobre vectores de 256 bits (por exemplo vectores de 8 operandos en simple precisión, ou 4 operandos en dobre precisión). Especificamente Intel foi incorporando ao longo dos anos instrucións para procesamento vectorial: primeiro as extensións MMX, logo as SSE e agora as AVX. A Figura 44 ilustra os tipos de datos vectoriais que soportan as diferentes extensións.

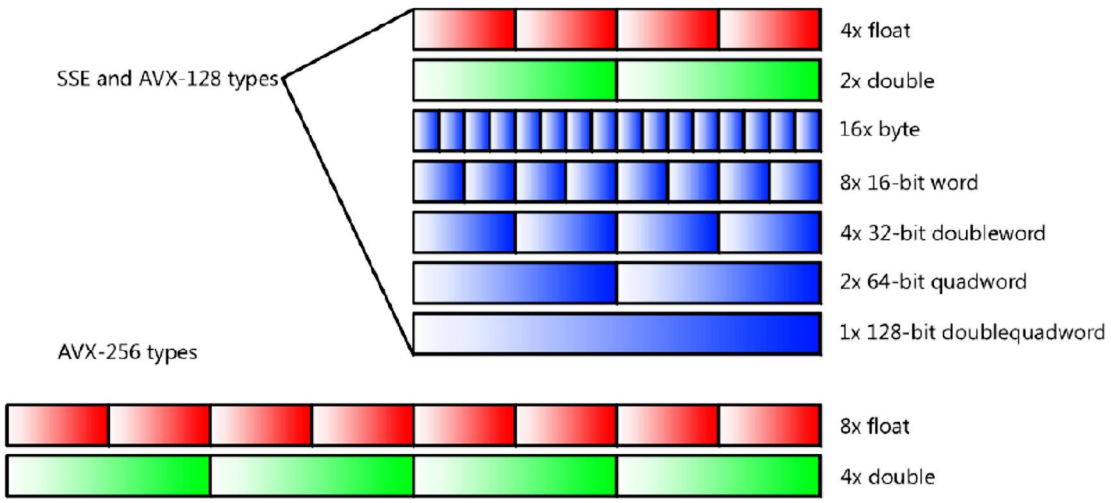


Figura 44: Tipos de datos vectoriais soportados polas diferentes xeracións de extensións vectoriais de procesadores de Intel.

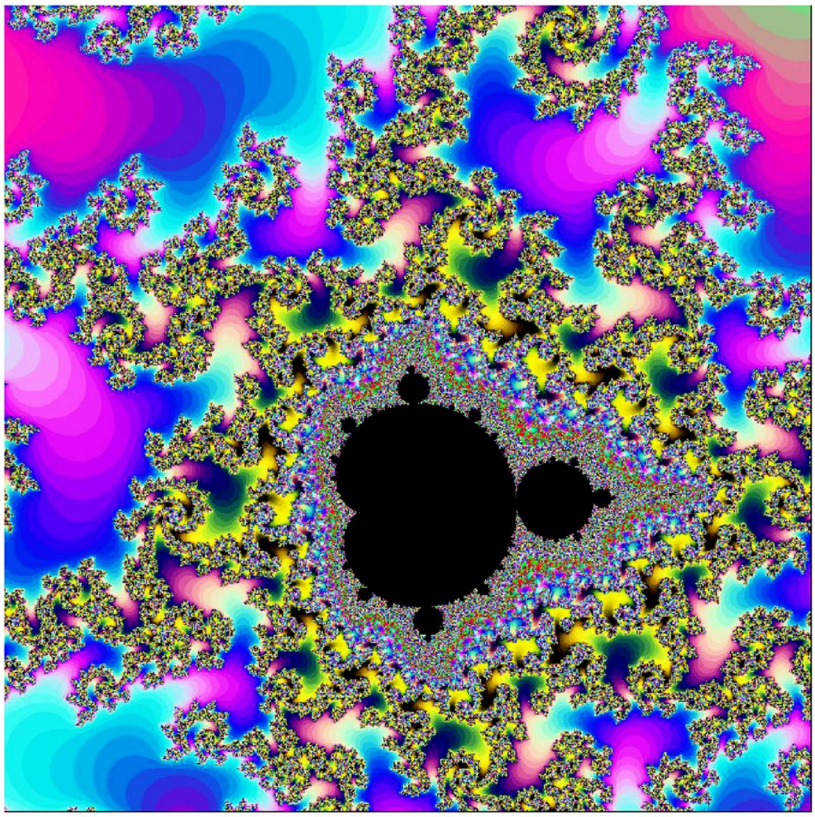


Figura 45: Fractal de Mandelbrot.

Como ilustración das posibilidades das extensións vectoriais, a Figura 45 amosa un exemplo de fractal de Mandelbrot, xerado co seguinte código C++ (escalar)

```
1.  // simple code to compute Mandelbrot in C++
2.  #include <complex>
3.  void MandelbrotCPU(float x1, float y1, float x2, float y2,
4.                     int width, int height, int maxIters, unsigned short * ima
    ge)
5.  {
6.      float dx = (x2-x1)/width, dy = (y2-y1)/height;
7.      for (int j = 0; j < height; ++j)
8.          for (int i = 0; i < width; ++i)
9.          {
10.             complex<float> c (x1+dx*i, y1+dy*j), z(0,0);
11.             int count = -1;
12.             while ((++count < maxIters) && (norm(z) < 4.0))
13.                 z = z*z+c;
14.             *image++ = count;
15.         }
16. }
```

Esta é unha aplicación na que para cada coordenada dunha imaxe calcúlase unha iteración con números complexos. Logo faise corresponder o número de iteracións

realizado á unha determinada cor, e xérase a imaxe resultante. Dicir que os cálculos para cada pixel da imaxe son independentes, e polo tanto é un código doado de vectorizar. O seguinte código ilustra a utilización das extensións vectoriais para este exemplo:

```
1.  float dx = (x2-x1)/width;
2.  float dy = (y2-y1)/height;
3.  // round up width to next multiple of 8
4.  int roundedWidth = (width+7) & ~7UL;
5.
6.  float constants[] = {dx, dy, x1, y1, 1.0f, 4.0f};
7.  __m256 ymm0 = _mm256_broadcast_ss(constants);   // all dx
8.  __m256 ymm1 = _mm256_broadcast_ss(constants+1); // all dy
9.  __m256 ymm2 = _mm256_broadcast_ss(constants+2); // all x1
10. __m256 ymm3 = _mm256_broadcast_ss(constants+3); // all y1
11. __m256 ymm4 = _mm256_broadcast_ss(constants+4); // all 1's (iter increments)

12. __m256 ymm5 = _mm256_broadcast_ss(constants+5); // all 4's (comparisons)
13.
14. float incr[8]={0.0f,1.0f,2.0f,3.0f,4.0f,5.0f,6.0f,7.0f}; // used to reset th
    e i position when j increases
15. __m256 ymm6 = _mm256_xor_ps(ymm0,ymm0); // zero out j counter (ymm0 is just
    a dummy)
16.
17. for (int j = 0; j < height; j+=1)
18. {
19.     __m256 ymm7  = _mm256_load_ps(incr);  // i counter set to 0,1,2,..,7
20.     for (int i = 0; i < roundedWidth; i+=8)
21.     {
22.         __m256 ymm8 = _mm256_mul_ps(ymm7, ymm0);  // x0 = (i+k)*dx
23.         ymm8 = _mm256_add_ps(ymm8, ymm2);         // x0 = x1+(i+k)*dx
24.           m256 ymm9 =  mm256 mul ps(ymm6, ymm1);  // y0 = j*dy
25.         ymm9 = _mm256_add_ps(ymm9, ymm3);         // y0 = y1+j*dy
26.         __m256 ymm10 = _mm256_xor_ps(ymm0,ymm0);  // zero out iteration coun
    ter
27.         __m256 ymm11 = ymm10, ymm12 = ymm10;        // set initial xi=0, yi=
    0
28.
29.         unsigned int test = 0;
30.         int iter = 0;
31.         do
32.         {
33.             __m256 ymm13 = _mm256_mul_ps(ymm11,ymm11); // xi*xi
34.             __m256 ymm14 = _mm256_mul_ps(ymm12,ymm12); // yi*yi
35.             __m256 ymm15 = _mm256_add_ps(ymm13,ymm14); // xi*xi+yi*yi
36.
37.             // xi*xi+yi*yi < 4 in each slot
38.             ymm15 = _mm256_cmp_ps(ymm15,ymm5, _CMP_LT_OQ);
39.             // now ymm15 has all 1s in the non overflowed locations
40.             test = _mm256_movemask_ps(ymm15)&255;      // lower 8 bits are c
    omparisons
41.             ymm15 = _mm256_and_ps(ymm15,ymm4);
42.             // get 1.0f or 0.0f in each field as counters
43.             // counters for each pixel iteration
44.             ymm10 = _mm256_add_ps(ymm10,ymm15);
45.
46.             ymm15 = _mm256_mul_ps(ymm11,ymm12);        // xi*yi
47.             ymm11 = _mm256_sub_ps(ymm13,ymm14);        // xi*xi-yi*yi
48.             ymm11 = _mm256_add_ps(ymm11,ymm8);         // xi <- xi*xi-
    yi*yi+x0 done!
49.             ymm12 = _mm256_add_ps(ymm15,ymm15);        // 2*xi*yi
50.             ymm12 = _mm256_add_ps(ymm12,ymm9);         // yi <-
     2*xi*yi+y0
51.
52.             ++iter;
53.         } while ((test != 0) && (iter < maxIters));
```

```
54.
55.         // convert iterations to output values
56.         __m256i ymm10i = _mm256_cvtps_epi32(ymm10);
57.
58.         // write only where needed
59.         int top = (i+7) < width? 8: width&7;
60.         for (int k = 0; k < top; ++k)
61.             image[i+k+j*width] = ymm10i.m256i_i16[2*k];
62.
63.         // next i position - increment each slot by 8
64.         ymm7 = _mm256_add_ps(ymm7, ymm5);
65.         ymm7 = _mm256_add_ps(ymm7, ymm5);
66.     }
67.     ymm6 = _mm256_add_ps(ymm6,ymm4); // increment j counter
68. }
```

Unha das desvantaxes das extensións vectoriais é a complexidade de programación. Comparando ambas versións, resulta obvio que a complexidade de programación vese incrementada dun xeito substancial. Non obstante, o incremento de complexidade paga a pena cando é necesario aumentar a velocidade de procesamento. A Figura 46 ilustra a ganancia en velocidade que se pode acadar coas diferentes versións das extensións vectoriais (SSE e AVX), tomando como referencia a implementación en C++ utilizando floats (como referencia inclúense os datos para a implementación utilizando o tipo de datos complex de C++) e para diferentes tamaños do problema. Como vemos coas extensións AVX e tamaños do problema grandes, pódese acadar unha ganancia en velocidade preto de 8 (xa que se fai procesamento vectorial de 8 floats).

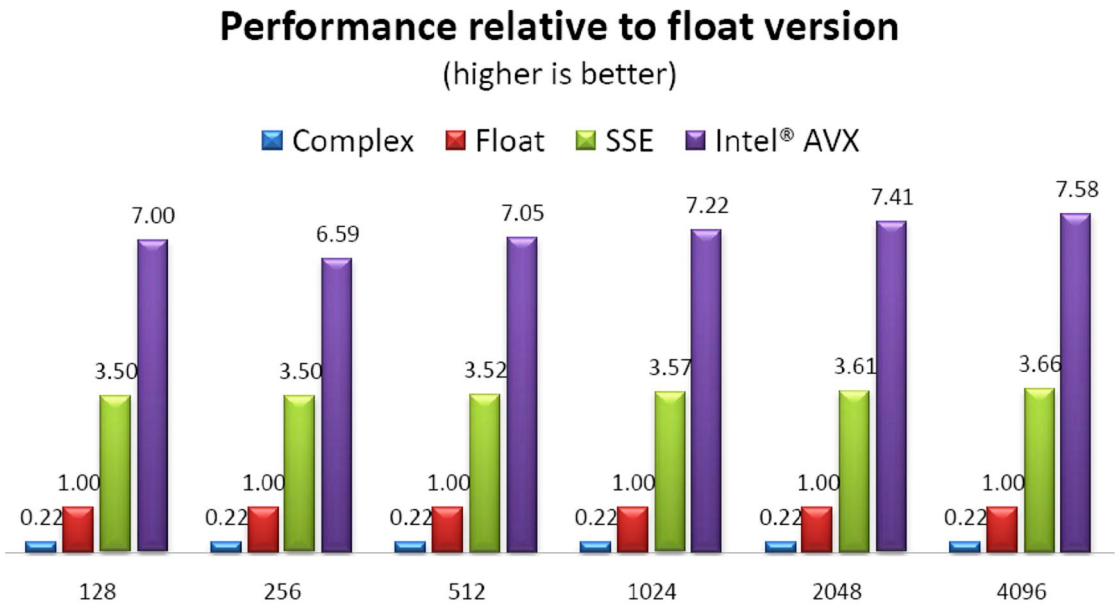


Figura 46: Ganacia en velocidade ao utilizar as diferentes extensións vectoriais para diferentes tamaños do problema.

7- Comparación de sistemas: conxuntos de programas de referencia (benchmarks)

Un aspecto importante para o deseño de procesadores, e tamén para os usuarios/compradores, é o de ter unha referencia para poder comparar diferentes procesadores. A opción máis aceptada na actualidade é a de utilizar un conxunto de programas de referencia (benchmark suites en inglés). Este conxunto de programas pretende ser representativo da carga de traballo real que levarán a cabo os procesadores. Dalgún xeito pretenden facer unha estimación da distribución de probabilidade de ter unha certa ganancia en velocidade con respecto a un sistema de referencia cando collemos un programa ao azar (dentro dos que se corresponden co tipo de carga de traballo considerada).

Para sistemas de sobremesa un dos conxunto de programas de referencia máis utilizados e SPEC CPU (actualmente coa versión do 2006). O conxunto consta de 12 programas que traballan con enteiros, e 17 que teñen carga elevada de computacións de punto flotante. Para analizar as prestacións dun sistema en termos de latencia, para cada programa facilítase o que se chama o SPECratio, que é a ganancia en velocidade cando se corre este programa no sistema en comparación cun sistema de referencia (tempo de execución no sistema de referencia partido polo tempo de execución no sistema avaliado). Para analizar as prestacións do sistema en termos de produtividade (throughput), o sistema corre múltiples copias de cada programa, para estimar cantas tarefas é capaz de realizar por segundo. Para cada programa facilítase este valor en relación ao sistema de referencia (normalizado). Para cada programa, distínguese entre a versión ñbaseò e ñpeakò. Na versión ñbaseò a compilación é estándar para todos os programas, e delimítanse os ñflagsò de compilación que se poden activar. A versión ñpeakò indica todo o potencial que se pode obter co compilador, xa que existe liberdade para aplicar as optimizacións de compilación que se estimen oportunas.

Un aspecto importante é o de como presentar os resultados cun número simple para todos os programas. A media aritmética da lugar a resultados enganosos, cando se comparan distintos sistemas. Isto é debido á distribución da probabilidade de obter unha certa ganancia en velocidade seleccionando un programa ao azar dentro dos típicos que se usan neste tipo de sistemas. A distribución non é Gaussiana (distribución normal), xa que o resultado do proceso non é por adición de certos efectos, senón por multiplicación dos mesmos (por exemplo o CPI a frecuencia, o número de instrucións, etc). Isto da lugar a un tipo de distribución log-normal, na que ao representar a variable estatística en escala logarítmica (eixo x), da lugar a unha distribución de probabilidade tipo Gaussiana. Neste tipo de distribucións, a media relevante é a xeométrica. Esta media obtense multiplicando as n mostras e logo calculando a raíz n. Existe tamén a correspondente definición de desviación estándar, pero formulada de xeito xeométrico. Esta desviación é un factor que indica a dispersión dos datos. Por exemplo, para unha media M, e unha dispersión S, nesta distribución, existe un 70% de probabilidade de atopar unha mostra ao facer unha medida no rango [M/S, M x S].

En resumo, os resultados de SPEC_CUP2006 que usualmente se dan son:

SPECint2006: a media xeométrica das ganancias en velocidade para o conxunto de programas con computación intensiva con enteiros, e na versión de compilación ñpeakò.
SPECint_base2006: o mesmo que a anterior pero coas opcións de compilación ñbaseò.
SPECint_rate2006: e media xeométrica das ganancias en produtividade con opcións de compilación ñpeakò.
SPECint_rate_base2006: o mesmo que a anterior pero con opcións de compilación ñbaseò.

As versións para os programas intensivos en computación de punto flotante son: **SPECfp2006**, **SPECfp_base2006**, **SPECfp_rate2006**, **SPECfp_rate_base2006**.

Normalmente só se utiliza o valor da media xeométrica, pero tamén é interesante e importante coñecer o valor da desviación S para ver o rango de dispersión.

SPEC conta con outros moitos conxuntos de programas para poder comparar diferentes sistemas noutros ámbitos (por exemplo **SPECweb2009** para sistemas web, **SPECvirt_sc2010** para avaliar as prestacións da virtualización de servidores en centros de datos, e **SPECpower_ssj2008** para avaliar a relación potencia-prestación en servidores, etc).

Outro conxunto de programas de referencia bastante popular son os TPC, que pretenden comparar sistemas para cargas de traballo baseadas en transaccións en grandes bases de datos (cargas OLTP). Existen diferentes versións: TPC-C, TPC-E, TPC-H e TPC-Energy. Todos miden as prestacións en número de transaccións completadas por segundo, e inclúen un límite de tempo de resposta (latencia) de cada transacción.

Dende logo existen outros moitos conxuntos de programas de referencia, que atopan certo impacto en eidos máis restrinxidos, como multimedia, gráficos ou supercomputación.