República Federativa do Brasil
Ministério do Desenvolvimento, Indústria e do Comércio Exterior
Instituto Nacional da Propriedade Industrial

(21) **PI 0901325-3 A2**

BRPI0901325A2

(22) Data de Depósito: 23/04/2009
(43) Data da Publicação: 28/02/2012
(RPI 2147)

(51) Int.Cl.:
G06F 11/07

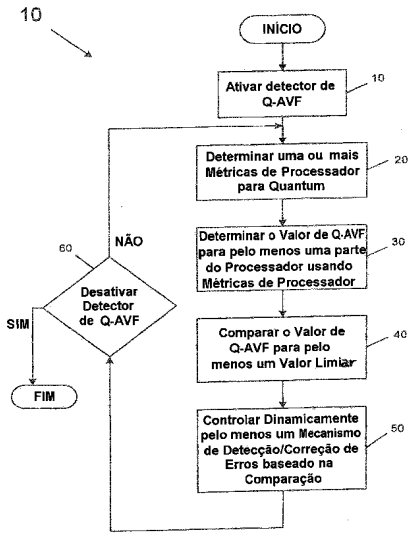
(54) Título: DETECÇÃO DE VULNERABILIDADE DE ARQUITETURA DE RECURSOS DE PROCESSADOR

(30) Prioridade Unionista: 23/04/2008 US 12/148,812

(73) Titular(es): INTEL CORPORATION

(72) Inventor(es): ARIJIT BISWAS, NIRANJAN SOUNDARARAJAN, SHUBHENDU MURKHERJEE

(57) Resumo: DETECÇÃO DE VULNERABILIDADE DE ARQUITETURA DE RECURSOS DE PROCESSADOR. A presente invenção provê, em uma modalidade, uni detector de quantum para detectar uma medida de vulnerabilidade para um processador baseado em uma métrica de processador, cada uma associada à operação de unia estrutura de processador durante um quantum, junto com um controlador para controlar uma unidade de mitigação de erro com base na medida de vulnerabilidade. Outras modalidades são descritas e reivindicadas.



"DETECÇÃO DE VULNERABILIDADE DE ARQUITETURA DE RECURSOS DE PROCESSADOR"

HISTÓRICO DA INVENÇÃO

Erros suaves decorrentes de radiação, causados principalmente por partículas de nêutron, tornaram-se o maior problema para projetistas de processadores. Devido ao fato de que este erro não reflete uma falha permanente do dispositivo, ele é denominado como erro suave ou transiente. Esses desarranjos de bits decorrentes de defeitos transientes acrescem àqueles causados por partículas alfa de material de empacotamento e impactos. Espera-se que o aumento exponencial no número de transistores em um único chip e a agressiva escalada na tensão tornará este problema significativamente pior nas futuras gerações de chips.

Para lidar com a incidência de raios cósmicos, algumas abordagens buscam proteger o maior percentual de latches em um processador ou outro dispositivo semicondutor com alguma forma de detecção de erros, tal como a paridade. De modo semelhante, a grande maioria dos arranjos tal como caches e arquivos de registro em processadores de elevado desempenho apresentam alguma forma de detecção e recuperação de erros. Conforme mais transistores são adicionados a um único chip, torna-se cada vez mais desafiador manter o mesmo nível de confiabilidade nas gerações sucessoras de processadores.

A confiabilidade é medida em falhas no tempo (FITs - Failures In Time), em que uma FIT representa uma falha em um bilhão de horas de operação. Existem três componentes principais da FIT: a taxa de erro intrínseca do circuito, a qual é uma função do processo de manufatura e esquemas de clock; o número de bits no microprocessador, o qual é um parâmetro de projeto; e o fator de vulnerabilidade arquitetural (AVF - Architectural Vulnerability Factor), que é a probabilidade de que uma troca de bit (bit flip) resulte em um erro visível ao usuário. Um erro visível ao usuário é definido como qualquer corrupção de bit que alcança os pinos do microprocessador e escapa para a memória principal ou um dispositivo de entrada/saída (I/O). Destes três componentes de FIT, o AVF é o único que pode variar

significativamente ao longo do tempo. De fato, estudos mostraram que o AVF pode variar muito (acima de 90% em alguns casos), de um programa a outro na média. O AVF também pode variar significativamente dentro de um programa, quando medido em tempo real ao longo de pequenos períodos de tempo conhecidos como quanta, ao invés da média de períodos longos.

A maioria dos mecanismos de detecção/recuperação de erros arquiteturais ou microarquiteturais tentam reduzir o AVF médio do microprocessador, melhorando assim a confiabilidade geral. No entanto, esta confiabilidade melhorada apresenta um custo na energia e no desempenho. Esquemas, tais como predição de paridade e resíduo que são usados principalmente para proteger unidades de execução podem ter um elevado custo de energia. Esquemas de redundância microarquiteturais tais como execução de redundância podem ter tanto um custo de energia como de desempenho uma vez que as unidades de execução que podem ser usadas para computar duas instruções diferentes em paralelo são usadas para computar uma única instrução de modo redundante. A maioria dos esquemas de mitigação de erros são sempre ativos, uma vez que não há atualmente forma confiável de medir o AVF em tempo real durante a execução de um programa. Como resultado, os custos de energia e desempenho para tais mecanismos são penalidades fixas.

BREVE DESCRIÇÃO DOS DESENHOS

A Figura 1 é um fluxograma de um método de acordo com uma modalidade da presente invenção.

A Figura 2 é um diagrama de blocos de um processador de acordo com uma modalidade da presente invenção.

A Figura 3 é um diagrama de blocos de um detector Q-AVF front end de acordo com uma modalidade da presente invenção.

A Figura 4 é um diagrama de blocos de um detector Q-AVF back end de acordo com uma modalidade da presente invenção.

A Figura 5 é um diagrama de blocos de um detector Q-AVF de sub-sistema de memória de acordo com uma modalidade da presente invenção.

A Figura 6 é um diagrama de blocos de um detector Q-AVF de núcleo de acordo com uma modalidade da presente invenção.

A Figura 7 é um diagrama de blocos de um sistema de acordo com uma modalidade da presente invenção.

DESCRIÇÃO DETALHADA

Nas várias modalidades, um AVF quantizado (Q-AVF) pode prover uma indicação em tempo real da vulnerabilidade arquitetural durante a execução do processador. Tal Q-AVF pode variar significativamente de um AVF médio. Devido ao fato de que o Q-AVF é uma indicação em tempo real da vulnerabilidade arquitetural, conseqüentemente as modalidades podem influenciar esta informação para controlar o hardware de mitigação de erros. Assim, as modalidades podem medir uma métrica AVF quantizada e com base em sua medida, determinar se o hardware de mitigação de erros deve ser usado e a extensão de tal uso. Por exemplo, um ou mais limiares podem ser estabelecidos para vulnerabilidade aceitável e o hardware de mitigação pode ser ativado quando o Q-AVF excede um determinado limiar. Desta forma, o hardware de erro pode ser controlado dinamicamente, poupando energia e melhorando o desempenho quando a vulnerabilidade estiver baixa e apenas incorrer nestas perdas em troca de uma confiabilidade aumentada, quando a vulnerabilidade de erro estiver realmente alta.

Alguns exemplos de tais economias podem ser vistas em esquemas de mitigação de erros tais como multitarefas redundantes (RMT - Redundant Multi-Threading), ou redundância de modo dual (DMR - Dual-Mode Redundancy). No RMT, duas cópias de uma única linha de execução (thread) são executadas redundantemente e seus resultados são verificados antes de remeter. A DMR envolve executar a mesma instrução redundantemente em duas peças separadas de hardware, tal como duas portas de execução simétricas e comparar os resultados. Em ambos os casos, a redundância custa tanto energia como desempenho, uma vez que os recursos de redundância podem, caso contrário, serem usados para executar diferentes instruções em paralelo, melhorando assim o desempenho e a energia geral. Ao usar modalidades da presente invenção, o Q-AVF pode ser computado para identificar períodos de tempo quando a vulnerabilidade de erro suave for baixa e usada para controlar dinamicamente o uso destas técnicas de mitigação de erros para, assim, poupar em desempenho, energia e mesmo área, uma vez que a

área que foi usada para execução redundante pode ser usada para melhorar o desempenho.

Outros esquemas de mitigação de erros, tal como predição de paridade ou resíduo, geram bits de código especial com cada operação, que são em seguida armazenados com o resultado. Quando o resultado é lido, os códigos são rearranjados e comparados aos códigos armazenados de modo a detectar erros. Com base em um Q-AVF determinado, as modalidades podem controlar a execução para executar apenas esta geração de código quando a vulnerabilidade de erro estiver alta, poupando assim custo de energia de executar tais operações durante fases de baixa vulnerabilidade do programa. A paridade e a codificação de correção de erros (ECC - Error Correction Coding) podem ser igualmente habilitadas ou desativadas dinamicamente de modo a poupar energia usando modalidades da presente invenção.

Em outro exemplo, múltiplos limiares de vulnerabilidade podem ser providos, ao invés de um único controle binário liga/desliga. Neste exemplo, mitigação de erro pode ser aumentada ou reduzida com base na vulnerabilidade. Presumindo dois limiares de vulnerabilidade. Qualquer coisa abaixo do primeiro limiar é considerada baixa vulnerabilidade e não requer detecção ou correção. Qualquer coisa entre os primeiro e segundo limiares é considerada média vulnerabilidade e pode causar detecção de erro a ser executada, mas não correção. Qualquer coisa acima do segundo limiar é considerada alta vulnerabilidade e pode causar a implementação tanto da detecção como correção de erros. Neste caso, uma métrica Q-AVF pode ser usada para determinar em qual nível de vulnerabilidade o programa se encontra. Se o Q-AVF for médio (isto é, entre os dois limiares), a proteção de paridade pode ser invocada nas estruturas vulneráveis. Se o Q-AVF atingir um alto nível (isto é, acima do segundo limiar), a seguir um mecanismo de correção de erros pode ser usado alternativamente. Neste caso, em que a paridade e/ou ECC não é alinhada (e assim apresenta uma penarda de desempenho), o desempenho também é otimizado.

Ainda outro exemplo de controle dinâmico de hardware de mitigação de erros usando informação de Q-AVF é aquele

com relação aos pontos de verificação. Os pontos de verificação são instantâneos periódicos de estado arquitetural que são usados para retomar a um ponto operacional "bom" conhecido quando um erro for detectado por algum mecanismo de detecção de erros. Os pontos de verificação tomam normalmente alguma quantidade de memória, que pode tornar-se significativa em sistemas de múltiplos núcleos ou múltiplos soquetes de larga escala, especialmente se múltiplos pontos de verificação forem requeridos. As modalidades podem ser usadas para determinar a freqüência na qual pontos de verificação são gerados e armazenados. Quando a vulnerabilidade for baixa, tanto memória como energia podem ser poupados ao reduzir a freqüência da geração de pontos de verificação. De modo recíproco, se a vulnerabilidade for alta, então a freqüência de geração de pontos de verificação pode ser aumentada. Por conseguinte, isto otimiza tanto a energia como o desempenho.

Em várias modalidades, diferentes formas de aproximar o Q-AVF podem ser concretizadas. Ademais, tais métricas Q-AVF podem ser determinadas para várias estruturas dentro de um processador ou outro dispositivo semicondutor. Em algumas implementações, as métricas Q-AVF podem ser analisadas em diferentes escalas. Por exemplo, em algumas implementações uma base de núcleo completo pode ser usada para medidas Q-AVF. Alternativamente ou adicionalmente, medidas refinadas de Q-AVF podem ser determinadas e usadas para controlar mecanismos de detecção/correção de erros. Por exemplo, em outras implementações diferentes partes de um processador, tal como um front end, um back end, um sub-sistema de memória, ou outra de tais partes de um processador pode apresentar análises Q-AVF independentes realizadas no mesmo. Além disso, métricas diferentes destes vários recursos de processador podem ser analisados para estimar o Q-AVF. Em uma implementação, funções lineares de determinadas métricas de processador podem ser usadas para aproximar o Q-AVF em diferentes pontos no tempo para várias estruturas.

Em uma modalidade, um mecanismo de hardware pode ser provido de modo que possa monitorar métricas específicas, obtidas a partir de uma análise de regressão linear que pode aproximar o AVF e o Q-AVF de uma parte significativa de um

processador. Equações lineares podem ser implementadas em hardware, embora o escopo da presente invenção não esteja limitado a este aspecto. O mecanismo de hardware pode incluir um conjunto de contadores de hardware que, quando o mecanismo estiver acionado, contam um determinado parâmetro para um período de tempo ajustado, designado aqui como um quantum. Ao final de cada quantum, os valores em cada um dos contadores podem ser processados através de uma função linear.

A função linear específica depende de detalhes microarquiteturais do processador e podem ser obtidos por análise de regressão linear ao correlacionar o AVF real a uma determinada lista de métricas. Em uma modalidade, as funções lineares usadas pelas várias lógicas aqui discutidas podem ser baseadas em uma ou mais análises de regressão linear executadas enquanto se projeta um processador. Tais análises são baseadas nos parâmetros de projeto do próprio processador e, se aqueles parâmetros de projeto mudam, a análise de regressão pode ser atualizada para obter uma nova função. Em algumas implementações, conforme nenhuma mudança fundamental ocorre no comportamento arquitetural dentro de uma família de processador (por exemplo, apenas diferentes tamanhos de parâmetros de projeto tal como tamanho do cache, tamanho do buffer, e assim por diante), ponderações (isto é, coeficientes) podem mudar, mas caso contrário a função pode permanecer a mesma ao longo de diversos projetos de processador. Em várias modalidades, os pesos (isto é, coeficientes) podem ser programáveis a partir de registradores de controle, de modo que valores padrão mutáveis dos registradores de controle podem atualizar as funções lineares.

Vários fatores podem ser considerados quando se constrói adequadamente um mecanismo de hardware. Por exemplo, contadores apresentam capacidade finita; portanto eles não podem contar por períodos extremamente longos de tempo sem transbordamento. Se eles transbordarem, eles perderão a informação. Assim, o quantum durante o qual o mecanismo opera pode ser ajustado para ser pequeno o suficiente tal que os contadores não transbordem, por exemplo, da ordem de aproximadamente uma centena a aproximadamente um milhar de ciclos de processador,

apesar do escopo da presente invenção não estar limitado a este aspecto. Alternativamente, um tamanho de quantum pode ser escolhido tal que o Q-AVF resultante proveja um nível específico de acurácia em tempo real. Neste caso, os contadores para as métricas de equação linear devem ser dimensionados tal que eles possam garantir que não transbordem durante o quantum.

A análise de regressão linear pode aproximar acuradamente valores Q-AVF para múltiplos programas diferentes para dezenas de milhões de ciclos. Com base em tais análises, uma equação linear de métricas de processador simples pode ser usada para estimar acuradamente a variação em tempo real na vulnerabilidade durante o tempo de execução de qualquer determinado programa. Esta variação na vulnerabilidade indica então que os mecanismos de recuperação de erros podem ser relaxados durante as fases menos vulneráveis, poupando assim custo de energia/desempenho de mitigação de erro durantes estas fases. Quando o programa entra em uma fase de alta vulnerabilidade, o mecanismo de mitigação de erros seria ativado, aumentado o consumo de energia e reduzindo o desempenho, mas reduzindo o Q-AVF a zero ou algum nível baixo aceitável quando ele for efetivamente necessário.

Um mecanismo de hardware gera então um valor Q-AVF para cada quantum no qual está ativo. Se este valor Q-AVF estiver abaixo de um valor aceitável especificado, o hardware de mitigação de erros pode ser ativado dinamicamente para reduzir o AVF. Se o valor Q-AVF cair abaixo de um limite específico, o hardware de mitigação de erros pode ser desativado dinamicamente. Desta forma, o hardware de mitigação de erros é usado apenas quando realmente for necessário e, caso contrário, não incorre em custo de operação na energia e desempenho.

Fazendo referência agora à Figura 1, é mostrado um fluxograma de um método de acordo com uma modalidade da presente invenção. Como mostrado na Figura 1, o método 10 pode ser usado para operar um detector de Q-AVF e controlar um ou mais mecanismos de correção/detecção de erros dinamicamente com base em informação proveniente do detector. Como mostrado na Figura 1, o método 10 pode começar ao ativar o detector de Q-AVF. Como

mencionado acima, o detector de Q-AVF pode ser um mecanismo que inclui um conjunto de contadores de hardware e lógica tal como lógica de controle, lógica de computação, lógica de comparação e assim por diante, como será discutido em implementações específicas abaixo. Em algumas implementações, pelo menos alguns dos contadores usados por um detector pode ativar contadores existentes, por exemplo, de uma unidade de monitoramente de desempenho de um processador.

Ainda com referência à Figura 1, depois da ativação, que pode ser controlada seletivamente (por exemplo, sob controle de programa, controle de usuário, controle de sistema operacional e assim por diante), uma ou mais métricas de processador podem ser determinadas para um determinado quantum (bloco 20). Por exemplo, a métrica de processador pode corresponder a contagens de diferentes recursos de processador que são usados durante um determinado quantum. Recursos específicos e sua informação de utilização serão discutidos adiante. Depois do término de um determinado quantum, a duração do qual pode variar em diferentes modalidades, um valor Q-AVF pode ser determinado para pelo menos uma parte do processador usando uma métrica de processador (bloco 30). Como exemplo, como será discutido em maiores detalhes abaixo, uma combinação de múltiplas métricas de processador de acordo com uma função determinada, tal como uma equação linear, pode ser usada para determinar o valor Q-AVF. Um ou mais valores Q-AVF podem ser determinados, tanto para recursos individuais de um processador, uma coleção de tais recursos correspondendo a uma determinada unidade de um processador, ou com base em um processador completo (por exemplo, núcleo).

A seguir, o controle passa ao bloco 40, em que o valor Q-AVF pode ser comparado a pelo menos um valor limiar. Como discutido acima, em algumas implementações um único limiar pode estar presente, enquanto em outras implementações múltiplos limiares podem estar presentes. Com base em tal comparação, pelo menos um mecanismo de detecção/correção de erros pode estar controlado dinamicamente (bloco 50). Por exemplo, se o valor Q-AVF estiver abaixo de um limiar, tais mecanismos podem ser desabilitados ou mecanismos de proteção adicionais podem ser

ativados. Em uma modalidade, um sinal de controle pode ser gerado por lógica de detector com base, pelo menos em parte, na comparação de um valor Q-AVF a um ou mais limiares. Em uma modalidade, um tal sinal de controle pode ser usado para controlar dinamicamente o hardware de mitigação de erros. Desta forma, um detector pode controlar dinamicamente uma penalidade para implementação de mitigação de erros. Em outras modalidades, uma lógica de detector pode prover um controle variável, tal como, ao invés de apenas ter dois ajustes (isto é, ligado ou desligado), múltiplos ajustes podem ser realizados. Por exemplo, redundância múltipla ou proteções múltiplas podem estar presentes em cada proteção cobrindo mais de um processador e também custando mais em termos de energia e desempenho. Em uma tal modalidade, múltiplos sinais de controle podem ser providos para escolher entre diferentes níveis de proteção. A seguir, o controle passa ao losango 60, onde pode ser determinado se o detector de Q-AVF deve ser ativado. Se for, o método 10 é finalizado, caso contrário o controle volta ao bloco 20, como discutido acima. Embora mostrado nesta implementação específica na modalidade da Figura 1, o escopo da presente invenção não está limitado a este aspecto. É possível observar que em algumas modalidades, um detector pode operar de modo on-line ou off-line. On-line significa que a detecção e correspondente controle de mitigação de erros pode ser realizado conforme o programa está sendo executado, ao passo que off-line significa que conforme o programa está sendo executado, podem ocorrer contagens de métricas de processador e,posteriormente, um Q-AVF pode ser determinado, que pode ser útil para determinar como um Q-AVF se parece para um determinado programa, de modo que ele possa ser modificado, compilado, ou controlado adequadamente.

Modalidades podem ser usadas em conexão com vários processadores ou outros dispositivos semicondutores. Porém, para a finalidade de discussão, a seguinte modalidade exemplificativa é realizada com referência a um processador, tal como um processador de múltiplos núcleos incluindo múltiplos núcleos homogêneos e heterogêneos. Fazendo referência agora à Figura 2, é mostrado um diagrama de blocos de um processador de acordo com uma modalidade da presente invenção. Como mostrado na

Figura 2, o processador 100 mostra um único núcleo, porém, deve ser entendido que este único núcleo pode ser replicado diversas vezes para formar um processador completo de múltiplos núcleos. Como mostrado na Figura 2, o processador 100 inclui normalmente três partes, a saber: um front end (FE) 110, um sub-sistema de memória (MS) 120, e um back end (BE) 130. O FE 110 é primeiramente uma rota (pipeline) ordenada e inclui filas de decodificação de instrução e unidades de fetch de instrução. O back end 130 inclui estações de reserva, buffers de retirada (retirement), unidades de execução, arquivos de registro e outras estruturas arquiteturais que regulam o fluxo de execução de dados. Em processadores fora de ordem, o BE 130 pode ser o local em que as instruções são programadas e executadas fora de ordem. Em determinadas arquiteturas, tais como: um processador Intel Architecture (IA32), o FE 110 e o BE 130 poder ser distinguidos adicionalmente pelo fato de que o FE 110 lida principalmente com macro-operações, que são instruções assembly, ao passo que o BE 130 lida com micro-operações (μops) decodificadas, que são instruções mais simples que apresentam mapeamento de muitos-para-um com macro-ops. O MS 120 inclui buffer de ordem de memória (MOB) e várias estruturas cache.

Assim, como mostrado na Figura 2, o front end 110 pode incluir um cache de instrução e lógica fetch 112 para obter informação de instrução proveniente de níveis cache inferiores (por exemplo, parte do sub-sistema de memória 120) e para armazenar temporariamente instruções que se esperam usar em breve ou múltiplas vezes. Acoplado ao cache de instrução 112 está um ou mais decodificadores 113, que podem decodificar instruções, uma fila de instrução 114 para armazenar instruções pendentes, em conexão a uma fila de decodificação de instrução 115. Além disso, uma unidade de predição de ramo (BPU - Branch Prediction Unit) 116 pode estar presente para predizer um ramo a ser tomado durante a execução de programa tal que as instruções associadas ao ramo predito possam ser obtidas antes do tempo. Como mostrado mais à frente na Figura 2, o front end 110 pode incluir um detector de Q-AVF de front end 118. Conforme será discutido mais à frente, o detector 118 pode realizar determinações de Q-AVF para vários

componentes front end mostrados na Figura 2 (e possivelmente outros tais componentes).

Como mostrado, o front end 110 comunica-se com o sub-sistema de memória 120. O sub-sistema 120 inclui caches 122, que podem ser caches de nível 1 (L1) e nível 2 (L2), em que o cache L1 pode ser um cache de dados e o cache L2 um cache unificado incluindo dados e intruções. Para ajudar com as traduções de endereço, o sub-sistema 120 inclui ainda um buffer lookaside de tradução de dados (DTLB) 124 e um gerenciador de perda de página (PMH - Page Miss Handler) 126 para ajudar no acesso às informações requisitadas provenientes de outros níveis de uma hierarquia de memória quando uma perda ocorre ao DTLB 124. Além disso, o sub-sistema de memória pode incluir um buffer de ordem de memória (MOB) 128, que pode incluir buffers de armazenagem e carga para armazenar entradas associadas às instruções de transferência de dados pendentes, por exemplo, instruções de armazenagem e carga. Além disso, o sub-sistema 120 inclui um detector de Q-AVF de sub-sistema de memória 129. Como será discutido mais à frente, o detector 129 pode realizar determinações Q-AVF para vários componentes de sub-sistema de memória mostrados na Figura 2 (e possivelmente outros tais componentes).

Acoplado posteriormente, tanto ao front end 110 como ao sub-sistema de memória 120, está o back end 130, que pode ser usado para processar instruções fora de ordem e reordenar tais instruções para retirada. Assim, como mostrado na Figura 2, o back end 130 inclui um buffer de ordem de retirada e arquivos de registro 132, uma ou mais unidades de execução 134, que podem incluir unidades inteiras, unidades de ponto flutuante, unidades de vetor e assim por diante. Além disso, o back end 130 pode incluir estações de reserva 136, que podem ser usadas para prover instruções e operandos para as unidades de execução 134. Por outro lado, entradas são providas para as estações de reserva 136 por tabelas de alocação e renomeação 138, as quais podem receber instruções de entrada provenientes do front end 110 e alocá-las nos vários recursos, incluindo unidades de execução 134 e arquivos de registro 132, junto com a execução de quaisquer sobra do número

limitado de registradores lógicos ao número elevado de registradores físicos presentes nos arquivos de registro 132. O back end 130 inclui ainda um Q-AVF de back end 139. Como será discutido mais a frente, o detector 139 pode realizar determinações Q-AVF para vários componentes back end mostrados na Figura 2 (e possivelmente outros tais componentes). Como mostrado adicionalmente na Figura 2, um detector Q-AVF de nível de núcleo 140 pode estar presente ainda para gerenciar determinações Q-AVF baseadas em núcleo.

Enmbora mostrada com esta implementação específica na modalidade da Figura 2, o escopo da presente invenção não está limitado a este aspecto. Por exemplo, em várias implementações o Q-AVF pode ser computado em diferentes granularidades. Em um extremo é uma base por estrutura (que poderia requerer hardware de controle de Q-AVF customizado para cada estrutura). No outro extremo, o Q-AVF pode ser calculado em uma base por núcleo, com outras implementações intermediárias entre as duas. Por exemplo, agregar estruturas que se associam naturalmente em blocos grandes tais como FE, BE e MS podem prover um método eficiente para avaliar o Q-AVF em regiões distintas de um processador, enquanto mantém o custo de implementação do hardware em um mínimo.

Embora a Figura 2 mostre que cada parte do processador 100 pode incluir seu próprio detector e que aquele processador 100 pode ainda incluir um detector de nível de núcleo, deve ser entendido que nem todos estes detectores podem estar presentes em uma implementação específica. Por exemplo, em várias implementações um número pequeno ou apenas um único de tais detectores pode estar presente em um determinado processador. Para facilidade de discussão, a funcionalidade de cada detector mostrado na Figura 2 será discutida. Porém, observa-se que, em uma determinada implementação de processador, apenas um, um subconjunto ou todos os detectores descritos podem estar presentes.

Como discutido acima, o FE 110 inclui a fila de instruções que mantém macro-ops decodificadas e a fila de decodificação de instruções que mantém micro-ops decodificados. Em

uma modalidade, um conjunto mínimo de métricas de processador básicas podem ser usadas para computar um Q-AVF de front end para entre 95% de acurácia na média, com um pior caso de acurácia melhor que 80%. Nesta modalidade, três parâmetros para determinações de Q-AVF de FE são: (1) utilização de fila de decodificação de instruções (IDQ) (que pode ser determinado com base no número de entradas válidas em cada ciclo IDQ e no tamanho de quantum); utilização de buffer de ordem de retirada (ROB) (que pode ser determinado com base no número de entradas válidas em cada ciclo ROB e no tamanho de quantum); e o número de não-predições de ramos que ocorrem durante um quantum.

A utilização pode ser definida como o número médio de entradas válidas em uma determinada estrutura durante um quantum. É essencialmente uma medida da taxa de ocupação da estrutura. As métricas de utilização são normalmente determinadas como alguma contagem dividida por alguma quantidade de tempo. Neste caso, a contagem é o número total de entradas válidas somadas a cada ciclo de energia e a quantidade de tempo é fixada como o tamanho de quantum. No interesse de simplificar o hardware, os valores de tamanho de quantum podem ser escolhidos para serem uma potência de dois. Desta forma, uma operação de divisão é simplificada a um simples deslocamento de bit para o valor de contagem por log2 do tamanho do quantum, resultando em um valor inteiro como resultado da operação de divisão.

Em uma modalidade, esta operação log2 pode ser realizada com uma máquina de estados que conta o número de zeros do valor de quantum, a partir do bit menos significativo ao mais significativo, até o primeiro lógico um. O resultado é log2 do valor do quantum e esta operação apenas pode ser retomada quando o tamanho de quantum muda. A seguir, este valor pode ser armazenado em um registrador que é usado para todas as computações de utilização. Uma simples verificação para assegurar que apenas o bit mais significativo do valor de tamanho de quantum especificado é gravado no registrador de configuração de tamanho de quantum poderia forçar todos os valores de tamanho de quantum a serem arredondados para baixo para a menor potência de 2.

Fazendo referência agora à Figura 3, é mostrado um diagrama de blocos de um detector de Q-AVF de front end de acordo com uma modalidade da presente invenção. Como mostrado na Figura 3, o detector 200 pode incluir uma pluralidade de contadores, a saber um contador de instrução IDQ 210, um contador de instrução ROB 220, e um contador não-preditivo de ramo 230. Cada um destes contadores pode ser configurado para contar a utilização do determinado recurso (ou não-predições de ramo) que ocorrem durante um determinado quantum. O comprimento do quantum pode ser armazenado em um registrador de configuração de tamanho de quantum 235, que pode ser ajustado por um registrador de controle de processador, por exemplo, sob controle do sistema operacional (OS). Além disso, um registrador de configuração log2 de quantum 245 pode estar presente para prover um valor log2 de quantum, como discutido acima. Este valor log2 de quantum pode ser gerado usando uma máquina de estados 240, com base no valor no registrador de configuração de tamanho de quantum 235.

Como mostrado mais à frente na Figura 3, uma lógica dependente de arquitetura 260, que pode ser adaptada para realizar uma equação linear dependente de arquitetura com base em informações de métrica de processador, é acoplada para receber várias entradas. Especificamente, a lógica 260 recebe a saída do contador de não-predição de ramo 230, junto com a saída de um par de deslocadores de bit 250 e 255, cada um dos quais é adaptado para prover uma medida da utilização, a saber uma utilização ROB e uma utilização IDQ. Com base nas informações providas a eles, a lógica 260 pode gerar um valor Q-AVF que pode ser armazenado em um registrador 265. Este valor Q-AVF pode ser determinado para cada quantum.

Deve-se observar que a lógica 260 pode implementar uma equação linear que leva em conta os vários dados de entrada providos ao mesmo. Em uma modalidade, a equação linear pode ser como a seguir:

Q-AVF de FE = -6,94888841 + 2,13964287(X) + 0,00000984(Y) - 0,00025137(Z) [EQ. 1]

Em que

X = Utilização de IDQ,

Y = Utilização de ROB,

Z = Número de não-predições de ramo.

Deve-se observar que os coeficientes podem ser números de ponto flutuante muito pequenos. Porém, isto pode ser remediado ao preencher todos os valores com zeros o suficiente (ou multiplicar por uma grande potência de 10) tal que os números tornem-se inteiros. Uma operação semelhante também pode ser realizada nos valores limiares antes que a operação de comparação seja realizada em uma lógica de comparação 270. Deve-se observar que esta operação pode ser realizada off-line ou apenas uma vez a cada momento em que os coeficientes ou valores limiares forem alterados. Desta forma, a operação precisa apenas ser realizada uma vez quando novos valores são alimentados, uma ocorrência normalmente rara.

Em várias implementações, os coeficientes mudarão dependendo da arquitetura e parâmetros de projeto do processador, e uma análise de regressão pode ser usada para determinar coeficientes para um determinado projeto. Em algumas implementações, o termo constante também pode ser ignorado para simplificar o hardware. Se a constante for removida, ela pode ser considerada no valor limiar.

Como mostrado na Figura 3, o Q-AVF determinado pode ser provido para a lógica de comparação 270, onde pode ser comparado com um ou mais limiares de Q-AVF que podem ser armazenados em um registrador limiar 275. Dependendo de o valor Q-AVF determinado exceder um ou mais limites, um sinal de controle 280 pode ser gerado por lógica de comparação 270 e usado para controlar um ou mais mecanismos de detecção/correção de erros como discutido acima. Embora mostrado nesta implementação específica na modalidade da Figura 3, o escopo da presente invenção não está limitado a este aspecto e outras implementações são possíveis. Além disso, deve-se observar que o detector 200 pode ser implementado em hardware, software, firmware ou combinações de tais para permitir a determinação de um valor Q-AVF e seu uso para controlar um ou mais mecanismos de erro.

Adicionalmente ao detector de front end 200, algumas modalidades podem incluir ainda um detector de back end,

ou uma implementação pode incluir apenas um detector de back end. Conforme descrito acima, o back end 130 inclui as estações de reserva (RS), que mantém instruções de micro-op e dados e as programa para as unidades de execução e o ROB, que mantém micro-operações que estão em andamento e as reordena antes de remetê-las ao estado arquitetural. Em uma modalidade, um conjunto mínimo de métricas básicas de processador pode ser usado para computar o Q-AVF para dentro de 95% de acurácia na média, com um pior caso de acurácia melhor que 75%. Para o Q-AVF de back end, tais parâmetros podem incluir, em uma modalidade: utilização de estação de reserva (RS) (que pode ser determinado com base no número de entradas válidas no RS a cada ciclo e no tamanho de quantum); utilização de buffer de armazenamento (STB) (que pode ser determinado com base no número de entradas válidas no ROB a cada ciclo e no tamanho de quantum); o número de não-predições de ramo; e utilização de fila de decodificação de instruções (IDQ) (que pode ser determinada com base no número de entradas válidas no IDQ a cada ciclo e no tamanho de quantum).

Fazendo referência agora à Figura 4, é mostrado um diagrama de blocos de um detector Q-AVF de back end de acordo com uma modalidade da presente invenção. Como mostrado na Figura 4, o detector 300 pode incluir uma pluralidade de contadores, a saber um contador de instrução RS 310, um contador de instrução STB 320, um contador de instrução IDQ 330 e um contador de não-predição de ramo 333. Cada um destes contadores pode ser configurado para utilização de contagem do determinado recurso (ou não-predições de ramo) que ocorrem durante um determinado quantum. O comprimento do quantum pode ser armazenado em um registrador de configuração de tamanho de quantum 335. Ainda, um registrador de configuração log2 de quantum 345 pode estar presente para prover um valor log2 de quantum, como discutido acima. Este valor log2 de quantum pode ser gerado usando uma máquina de estados 340, com base no valor no registrador de configuração de tamanho de quantum 335.

Como mostrado mais à frente na Figura 4, uma lógica dependente de arquitetura 360, que pode ser adaptada para realizar uma equação linear dependente de arquitetura baseada em

informação de métrica de processador, é acoplada para receber várias entradas. Especificamente, a lógica 360 recebe a saída do contador de não-predição de ramo 333, junto com a saída dos deslocadores de bit 350, 355 e 358, cada um dos quais é adaptado para prover uma medida de utilização, a saber uma utilização de RS, uma utilização de STB e uma utilização de IDQ. Com base nas informações providas ao mesmo, a lógica 360 pode gerar um valor Q-AVF que pode ser armazenado em um registrador 365. Este valor Q-AVF pode ser determinado para cada quantum.

Como mostrado na Figura 4, o Q-AVF determinado pode ser provido a uma lógica de comparação, em que pode ser comparado com um ou mais limites de Q-AVF que podem ser armazenados em um registrador de limiar 375. Dependendo em se o valor Q-AVF determinado excede um ou mais limiares, um sinal de controle 380 pode ser gerado pela lógica de comparação 370 e usado para controlar um ou mais mecanismos de detecção/correção de erros como discutido acima. Embora mostrado com esta implementação específica na modalidade da Figura 4 o escopo da presente invenção não está limitado a este aspecto, e outras implementações são possíveis. Além disso, deve-se observar que o detector 300 pode ser implementado em hardware, software, firmware ou em uma combinação de tais para permitir a determinação do valor Q-AVF e seu uso para controlar um ou mais mecanismos de erro.

Em uma modalidade, a lógica 360 pode executar uma equação linear dependente de arquitetura de acordo com a Equação 2:

Q-AVF de BE = 5,31777325 + 0,17012719(W) - 0,10485529(X) - 0,00007263(Y) + 0,25594201(Z) [EQ. 2]

Em que

W = Utilização de RS

X = Utilização de STB

Y = Número de não-predições de ramo

Z = Utilização de IDQ

Deve-se observar que, como discutido acima com relação à lógica 260 e à Equação 1, os coeficientes e/ou termos constantes podem ser manipulados anteriormente para facilitar a complexidade de computação. Embora mostrado para esta equação

específica com relação à modalidade da Figura 4, deve ser entendido que o escopo da presente invenção não está limitado a este aspecto.

Em ainda outra implementação, um detector de AVF pode ser associado ao sub-sistema de memória 120. Um tal detector de AVF pode ser o único detector presente em um processador, ou pode ser complementado com outros detectores tais como: os detectores front end e back end descritos acima. Como supracitado, o MOB 128 inclui um buffer de armazenamento que mantém endereços virtuais e físicos e dados para operações de armazenamento e um buffer de carga que mantém endereços para operações de carga. Em uma modalidade um conjunto mínimo de métricas básicas de processador pode ser usado para computar o Q-AVF para o sub-sistema de memória para dentro de 95% de acurácia na média, com um pior caso de acurácia melhor que 80%. Em uma modalidade, três parâmetros podem ser usados para determinar um Q-AVF para o MOB 128: utilização de buffer de armazenamento (STB) (que pode ser determinado com base no número de entradas válidas no STB a cada ciclo e no tamanho de quantum); utilização de estação de reserva (RS) (que pode ser determinado com base no número de entradas válidas no RS a cada ciclo e no tamanho de quantum; e um número de instruções descarregadas do buffer de armazenamento antes de uma resposta DTLB (isto é, contagem dos armazenamentos que foram descarregados depois do início de acesso DTLB, mas antes do retorno da resposta).

Fazendo referência agora à Figura 5, é mostrado um diagrama de blocos de um detector de Q-AVF de sistema de memória de acordo com uma modalidade da presente invenção. Como mostrado na Figura 5, o detector 400 pode incluir uma pluralidade de contadores, a saber um contador de instrução STB 410, um contador de instrução RS 420 e um contador de armazenamento descarregado (flushed) 430. Cada um destes contadores pode ser configurado para contar a utilização de um determinado recurso (ou armazenamentos descarregados) que ocorre durante um determinado quantum. O comprimento do quantum pode ser armazenado em um registrador de configuração de tamanho de quantum 435. Um registrador de configuração log2 de quantum 445 pode prover um

valor log2 de quantum, que pode ser gerado usando uma máquina de estados 440, com base no valor no registro de configuração de tamanho de quantum 435.

Como mostrado mais à frente na Figura 5, uma lógica dependente de arquitetura 460, que pode ser adaptada para executar uma equação linear dependente de arquitetura com base em informação de métrica de processador, é acoplada para receber várias entradas. Especificamente, a lógica 460 recebe a saída do contador de armazenagem descarregado 430, junto com a saída de um par de deslocadores de bit 450 e 455, cada um dos quais é adaptado para prover uma medida de utilização, a saber uma utilização STB e uma utilização RS. Com base na informação provida ao mesmo, a lógica 460 pode gerar um valor Q-AVF que pode ser armazenado em um registrador 465. Este valor Q-AVF pode ser determinado para cada quantum.

Como mostrado na Figura 5, o Q-AVF determinado pode ser provido para uma lógica de comparação 470, em que ele pode ser comparado com um ou mais limites Q-AVF que podem ser armazenados em um registrados limite 475. Dependendo de o valor Q-AVF determinado exceder um ou mais limites, um sinal de controle 480 pode ser gerado pela lógica de comparação 470 e usado para controlar um ou mais mecanismos de detecção/correção de erros como discutido acima. Embora mostrado nesta implementação específica na modalidade da Figura 5, o escopo da presente invenção não está limitado a este aspecto e outras implementações são possíveis. Além disso, deve-se observar que o detector 400 pode ser implementado em hardware, firmware ou em uma combinação de tais para permitir a determinação de um valor Q-AVF e seu uso para controlar um ou mais mecanismos de erro.

Em uma modalidade, a lógica 460 pode determinar um Q-AVF para o MOB de acordo com a Equação 3:

Q-AVF de MOB = 1,61820419 - 0,09218011(X) + 1,88263525(Y) - 0,0000307(Z) [EQ. 3]

Em que

X = Utilização de RS

Y = Utilização de STB

Z = Número de descargas STB antes de resposta DTLB.

Em outras modalidades, ao invés de detectores individuais para diferentes rotas de processador (por exemplo, os detectores de front end e back end e o detector de sub-sistema de memória discutido acima), em outras modalidades, um unido núcleo ou detector de chip completo, tal como o detector 140 da Figura 2 pode estar presente. Em algumas implementações, este pode ser o único detector, como as rotas FE e BE normalmente rastreiam as tendências de Q-AVF entre si. Além disso, como observado acima, existe uma sobreposição significativa nas métricas que determinam o Q-AVF de cada bloco do processador. Uma vez que determinados esquemas de mitigação de erro comprimem o AVF de todo o chip, um detector de Q-AVF pode ser provido para toda a rota de microprocessador como um todo. Um conjunto mínimo de métricas básicas de processador pode ser usado para computar o Q-AVF de um processador dentro de 95% de acurácia na média, com um pior caso de acurácia melhor que 80%. Em uma modalidade, sete parâmetros para a determinação de Q-AVF de chip completo são: utilização de fila de decodificação de instrução (IDQ); utilização de buffer de ordem de retirada (ROB); utilização de estação de reserva (RS); utilização de buffer de armazenagem (STB); número de não-predições de ramo; número de ciclos em que ROB está vazio; e número de instruções descarregadas do buffer de armazenamento antes da resposta DTLB.

Fazendo referência agora à Figura 6, é mostrado um diagrama de blocos de um detector de Q-AVF de back end de acordo com uma modalidade da presente invenção. Como mostrado na Figura 6, o detector 500 pode incluir uma pluralidade de contadores, a saber um contador de instrução IDQ 510, um contador de instrução ROB 515, um contador de instrução RS 520, um contador de instrução STB 525, um contador vazio ROB 530, um contador não-preditivo de ramo 533 e um contador de armazenagem descarregado 534. Cada um destes contadores pode ser configurado para contar a utilização do determinado recurso (ou não-predições de ramo ou armazenamentos descarregados) ocorrendo durante um determinado quantum. O comprimento do quantum pode ser armazenado em um

registrador de configuração de tamanho de quantum 535. Ainda adicionalmente, um registrador de configuração log2 de quantum 545 pode estar presente para prover um valor log2 de quantum, como discutido acima. Este valor log2 de quantum pode ser gerado usando uma máquina de estados 540, com base no valor no registro de configuração de tamanho de quantum 535.

Como mostrado mais à frente adicionalmente na Figura 6, uma lógica dependente de arquitetura 560, que pode ser adaptada para executar uma equação linear dependente de arquitetura baseada em informação de métrica de processador, é acoplada para receber várias entradas. Especificamente, a lógica 560 recebe a saída de contadores, a saber os contadores 510, 515, 520, 525, contador vazio ROB 530, contador de não-predição de ramo 533 e contador de armazenagem descarregada 534, junto com as saídas dos deslocadores de bit 550, 552, 554 e 556. Com base nas informações providas ao mesmo, a lógica 560 pode gerar um valor Q-AVF que pode ser armazenado em um registrador 565. Este valor Q-AVF pode ser determinado para cada quantum.

Como mostrado na Figura 6, o Q-AVF determinado pode ser provido a uma lógica de comparação 570, em que pode ser comparado com um ou mais limites Q-AVF que podem ser armazenados em um registrador de limite 575. Dependendo de o valor Q-AVF determinado exceder um ou mais limites, um sinal de controle 580 pode ser gerado pela lógica de comparação 570 e usado para controlar um ou mais mecanismos de detecção/correção de erros como discutido acima. Embora tenha sido mostrado com esta implementação específica na modalidade da Figura 6, o escopo da presente invenção não está limitado a este aspecto, e outras implementações são possíveis. Além disso, deve-se observar que o detector 500 pode ser implementado em hardware, software, firmware ou em uma combinação de tais para permitir a determinação de um valor Q-AVF e seu uso para controlar um ou mais mecanismos de erro.

Usando um detector de chip completo, os valores Q-AVF individuais para um front end, back end e MOB podem ser determinados de acordo com as Equações 4 a 6, respectivamente.

Q-AVF de FE = -10,5340689 + 1,91609831 (A) + 0,00279797(B) + 0,29720333(C) + 0,11824924(D) - 0,00026378(E) + 0,00001177(F) + 0,00000496(G) [EQ. 4]

Q-AVF de BE = 3,11760199 + 0,15379405(A) + 0,06943891(B) + 0,50535518(C) - 0,15898313(D) - 0,00007247(E) + 0,00000233(F) - 0,00001199(G) [EQ. 5]

Q-AVF de MOB = 3,5974914 + 0,11588197(A) - 0,02028408(B) - 0,22281406(C) + 1,89341998(D) + 0,00003006(E) - 0,00000246(F) - 0,00006469(G) [EQ. 6]

Em que

A = Utilização de IDQ

B = Utilização de ROB

C = Utilização de RS

D = Utilização de STB

E = Número de não-predições de ramo

F = Número de ciclos vazios ROB

G = Número de descargas STB antes de resposta DTLB.

Além disso, usando estas mesmas métricas de processador, um valor Q-AVF de chip completo pode ser determinado como descrito na Equação 7.

Q-AVF de Chip Completo = -1,2535192 + 0,728591443(A) + 0,0173176(B) + 0,19324815 (C) + 0,61756203(D) - 0,00010206(E) + 0,00000388(F) - 0,00002391(G) [EQ. 7]

Deve-se observar que, embora descrita por estas equações específicas, a lógica 560 pode determinar um Q-AVF de chip completo (junto com vários Q-AVFs das unidades) usando outras equações lineares.

Assim, em várias modalidades, o controle dinâmico do hardware de mitigação de erros com base em estimação Q-AVF pode otimizar grandemente a confiabilidade, energia e desempenho, reduzindo em mais de 50% os custos de energia e desempenho de implementação de hardware de mitigação de erros, ao passo que não compromete a confiabilidade, de modo significativo. As modalidades, assim, levam em conta o fato de que o AVF durante qualquer quantum específico pode variar significativamente do quanta prévio ou futuro. Como resultado, os programas apresentam

fases de alta vulnerabilidade e fases de baixa vulnerabilidade. Várias métricas de processador podem ser usadas para estimar o Q-AVF com acurácia significativa, permitindo controlar o que pode ser usado para modificar direta ou indiretamente o comportamento de mecanismos de mitigação de erros de modo a otimizar a energia, o desempenho e a confiabilidade.

As modalidades podem ser implementadas em diversos tipos de sistemas diferentes. Fazendo referência agora à Figura 7, é mostrado um diagrama de blocos de um sistema de acordo com uma modalidade da presente invenção. Como mostrado na Figura 7, o sistema multiprocessador 600 é um sistema interconectado ponto-a-ponto e inclui um primeiro processador 670 e um segundo processador 680 acoplado por meio de uma interconexão ponto-a-ponto 650. Como mostrado na Figura 7, cada um dos processadores 670 e 680 podem ser processadores de múltiplos núcleos, incluindo primeiro e segundo núcleos de processador (isto é, núcleos de processador 674a e 674b e núcleos de processador 684a e 684b), embora núcleos adicionais possam estar presentes nos processadores. Cada núcleo de processador pode incluir um ou mais detectores de Q-AVF, de acordo com uma modalidade da presente invenção. Desta forma, vários hardwares de mitigação de erros do núcleo podem ser controlados dinamicamente para reduzir quaisquer perdas de desempenho de energia quando não necessária.

Ainda fazendo referência à Figura 7, o primeiro processador 670 inclui ainda um hub de controlador de memória (MCH - Memory Controller Hub) e interfaces ponto-a-ponto (P-P) 676 e 678. De modo similar, o segundo processador 680 inclui um MCH 682 e interfaces P-P 686 e 688. Como mostrado na Figura 2, os MCHs 672 e 682 acoplam os processadores às memórias respectivas, a saber uma memória 632 e uma memória 634, que podem ser partes da memória principal (por exemplo, uma memória de acesso aleatório dinâmica (DRAM)) presa localmente aos respectivos processadores. O primeiro processador 670 e o segundo processador 680 podem ser acoplados a um chipset 690 por meio de interconexões P-P 652 e 654, respectivamente. Como mostrado na Figura 7, o chipset 690 inclui interfaces P-P 694 e 698.

Ademais, o chipset 690 inclui uma interface 692 para acoplar o chipset 690 a um mecanismo de gráficos de alto desempenho 638, por uma interconexão P-P 639. Por outro lado, o chipset 690 pode ser acoplado a um primeiro barramento 616 por meio de uma interface 696. Como mostrado na Figura 7, vários dispositivos de I/O 614 podem ser acoplados ao primeiro barramento 616, junto com uma ponte de barramento 618 que acopla o primeiro barramento 616 a um segundo barramento 620. Vários dispositivos podem ser acoplados ao segundo barramento 620 incluindo, por exemplo, um teclado/mouse 622, dispositivos de comunicação 626 e uma unidade de armazenamento de dados 628 tal como um disco rígido ou outro dispositivo de armazenamento de massa que pode incluir o código 630, em uma modalidade. Adicionalmente, um I/O de áudio pode ser acoplado ao segundo barramento 620.

O controle de uma unidade de mitigação de erros inclui, mas não está limitada a usar informação de controle ou valores Q-AVF diretamente ou armazenar temporariamente (buffering) a informação, de modo a identificar tendências ao longo dos múltiplos quanta, e apenas agir sob a informação se indicar que o Q-AVF estiver além de algum limiar para um único quantum ou algum número de quanta. Tal controle também pode ser usado para não apenas ativar/desativar o hardware de mitigação de erros, mas também para fazer a transição do hardware de mitigação de erros entre diferentes camadas de atividade tal como um modo de alta proteção ou um modo de baixa proteção ou modos intermediários, de modo a escolher entre diferentes esquemas de proteção com diversos custos de energia/desempenho. Um exemplo disto inclui esquemas de mitigação de erros que usam geração de pontos de verificação. A freqüência com as quais tais pontos de verificação são gerados determina a quantidade de progresso que será perdido quando da recuperação do erro. Uma modalidade pode ser usada para aumentar ou reduzir a freqüência com a qual os pontos de verificação são tomados dependendo da probabilidade real de erro sério como determinado pelo valor Q-AVF. Em tais casos, o valor Q-AVF pode ser comparado ao longo de múltiplos limiares de modo a determinar qual camada de proteção ativar.

Em uma modalidade, controle baseado em OS pode ser realizado por meio de fornecimento de informação do detector ao OS. Em tal modalidade, o OS pode determinar que um programa está entrando em uma região vulnerável e requisita uma mudança aos limiares. Por exemplo, se o limiar do detector (por exemplo, um primeiro limiar) funcionasse a 10% do limiar, o OS poderia forçar o limiar a um nível diferente ou fazer com que o detector mude seu tamanho de quantum.

As modalidades podem ser implementadas em código e podem ser armazenadas em um meio de armazenamento em que instruções que possam ser usadas para programar um sistema a realizar as instruções. O meio de armazenamento pode incluir, mas não está limitado a qualquer tipo de disco, incluindo disquetes, discos ópticos, memórias apenas para leitura de disco compacto (CD-ROMs), discos compactos regraváveis (CD-RWs) e discos magneto-ópticos, dispositivos semicondutores tais como: memórias apenas para leitura (ROMs), memórias de acesso aleatório (RAMs) tais como memórias de acesso aleatório dinâmicas (DRAMs), memórias de acesso aleatório estáticas (SRAMs), memórias apenas para leitura programáveis apagáveis (EPROMs), memórias flash, memórias apenas para leitura apagáveis eletricamente (EEPROMs), cartões magnéticos ou ópticos, ou qualquer outro tipo de mídia adequado para armazenar instruções eletrônicas.

Embora a presente invenção tenha sido descrita com relação a um número limitado de modalidades, os técnicos na área apreciarão diversas modificações e variações do mesmo. Pretende-se que as reivindicações anexas cubram todas as tais modificações e variações se estiverem dentro do espírito e escopo da presente invenção.

REIVINDICAÇÕES

1. Equipamento, caracterizado pelo fato de que compreende:

uma pluralidade de contadores, cada um para manter um valor de utilização para uma unidade de armazenamento para um processador durante um quantum;

um segundo contador para contar as diversas não-predições realizadas por uma unidade de predição do processador durante o quantum; e

uma primeira lógica para determinar uma medida de vulnerabilidade para pelo menos uma primeira parte do processador para o quantum com base, pelo menos em parte, nos valores de utilização e no número de não-predições.

2. Equipamento, de acordo com a reivindicação 1, caracterizado pelo fato de que compreende ainda uma segunda lógica para determinar uma medida de vulnerabilidade para uma segunda parte do processador para o quantum baseado, pelo menos em parte, em um segundo conjunto de valores de utilização e no número de não-predições.

3. Equipamento, de acordo com a reivindicação 2, caracterizado pelo fato de que pelo menos a primeira parte do processador corresponde a uma unidade de front end do processador, e a segunda parte do processador corresponde a uma unidade de back end do processador.

4. Equipamento, de acordo com a reivindicação 1, caracterizado pelo fato de que a primeira lógica está destinada a receber os valores de utilização para determinar a medida de vulnerabilidade de acordo com uma função linear possuindo uma pluralidade de termos, cada um incluindo um coeficiente e um dos valores de utilização.

5. Equipamento, de acordo com a reivindicação 4, caracterizado pelo fato de que compreende uma segunda lógica para receber a medida de vulnerabilidade e para gerar um sinal de controle para controlar pelo menos uma unidade de mitigação de erro do processador com base na medida de vulnerabilidade.

6. Equipamento, de acordo com a reivindicação 5, caracterizado pelo fato de que a segunda lógica é destinada a

habilitar um mecanismo de detecção de erros da pelo menos uma unidade de mitigação de erros se a medida de vulnerabilidade for maior que um primeiro limite e é destinada a habilitar um mecanismo de correção de erros da pelo menos uma unidade de mitigação de erros se a medida de vulnerabilidade for maior que um segundo limite.

7. Equipamento, de acordo com a reivindicação 5, caracterizado pelo fato de que a segunda lógica é destinada a receber uma pluralidade de medidas de vulnerabilidade de cada um dentre uma pluralidade de quanta e é destinada a gerar o sinal de controle com base na pluralidade de medidas de vulnerabilidade.

8. Equipamento, de acordo com a reivindicação 5, caracterizado pelo fato de que a segunda lógica é destinada a desativar a pelo menos uma unidade de mitigação de erros se a medida de vulnerabilidade for menor que o primeiro limiar, e em que pelo menos uma unidade de mitigação de erros deve ser configurada para executar tarefas de mitigação de não-erros quando desativados pela segunda lógica.

9. Equipamento, de acordo com a reivindicação 5, caracterizado pelo fato de que a segunda lógica é destinada a controlar dinamicamente pelo menos uma unidade de mitigação de erros a ser habilitada durante uma primeira parte de um programa com base em medidas de vulnerabilidade de uma primeira pluralidade de quanta e a ser desativada com base em medidas de vulnerabilidade de uma segunda pluralidade de quanta.

10. Equipamento, de acordo com a reivindicação 1, caracterizado pelo fato de que o equipamento compreende um processador e compreende adicionalmente um detector incluindo a pluralidade de contadores, o segundo contador e a primeira lógica, em que o detector está associado a uma primeira parte do processador.

11. Equipamento, de acordo com a reivindicação 10, caracterizado pelo fato de que pelo menos um dentre a pluralidade de contadores do detector seja associado a uma estrutura em uma segunda parte do processador.

12. Método, caracterizado pelo fato de que compreende:

determinar um valor de fator de vulnerabilidade arquitetural quantizado (Q-AVF) para um quantum para pelo menos uma parte de um processador usando métricas de processador;

comparar o valor Q-AVF a pelo menos um valor limite; e

controlar dinamicamente pelo menos uma unidade de mitigação de erros do processador com base na comparação.

13. Método, de acordo com a reivindicação 12, caracterizado pelo fato de que compreende adicionalmente determinar o valor Q-AVF de acordo com uma equação linear possuindo uma pluralidade de termos, cada um correspondendo a um valor de métrica de processador e um valor ponderado.

14. Método, de acordo com a reivindicação 13, caracterizado pelo fato de que cada um dos valores de métricas de processador corresponde a um valor de utilização para uma dentre uma pluralidade de estruturas de front end do processador.

15. Método, de acordo com a reivindicação 14, caracterizado pelo fato de que um primeiro valor de utilização corresponde a diversas entradas válidas em uma estrutura para cada ciclo do quantum.

16. Método, de acordo com a reivindicação 15, caracterizado pelo fato de que compreende adicionalmente determinar o valor de primeira utilização ao deslocar o número de entradas válidas por um valor logarítmico que corresponde ao quantum.

17. Método, de acordo com a reivindicação 12, caracterizado pelo fato de que compreende adicionalmente determinar o valor Q-AVF para uma unidade de front end do processador com base nos valores de métrica de processador para uma fila de decodificador de instruções da unidade de front end, um preditor de ramo da unidade de front end, e um buffer de ordem de retirada de uma unidade de back end do processador.

18. Sistema, caracterizado pelo fato de que compreende:

um processador de múltiplos núcleos possuindo uma pluralidade de núcleos, em que um primeiro núcleo inclui um detector para detectar uma medida de vulnerabilidade para o

primeiro núcleo com base em uma pluralidade de valores de métrica de processador, cada um associado a uma operação de uma estrutura de núcleo durante um primeiro período de tempo, e um controlador para controlar uma unidade de mitigação de erros do primeiro núcleo com base na medida de vulnerabilidade; e

uma memória de acesso aleatório dinâmica (DRAM) acoplada ao processador de múltiplos núcleos.

19. Sistema, de acordo com a reivindicação 18, caracterizado pelo fato de que o detector inclui uma pluralidade de contadores, cada um para manter os valores de métrica de processador durante um primeiro período de tempo, um segundo contador para contar diversas não-predições realizadas pela unidade de predição do primeiro núcleo durante um primeiro período de tempo, e uma primeira lógica para determinar a medida de vulnerabilidade para, pelo menos, uma primeira parte do primeiro núcleo do primeiro período de tempo com base, pelo menos em parte, nos valores de métrica de processador e no número de não-predições.

20. Sistema, de acordo com a reivindicação 19, caracterizado pelo fato de que o detector compreende adicionalmente uma segunda lógica para determinar uma segunda medida de vulnerabilidade para uma segunda parte do primeiro núcleo para o primeiro período de tempo com base, pelo menos em parte, em um segundo conjunto de valores de métrica de processador e no número de não-predições.

21. Sistema, de acordo com a reivindicação 19, caracterizado pelo fato de que a primeiro lógica é para receber os valores de métrica de processador e para determinar a medida de vulnerabilidade de acordo com uma função linear que possui uma pluralidade de termos, cada um incluindo um coeficiente e um dos valores de métrica de processador.

22. Sistema, de acordo com a reivindicação 18, caracterizado pelo fato de que o controlador é destinado a habilitar um mecanismo de detecção de erros da unidade de mitigação de erros se a medida de vulnerabilidade for maior que um primeiro limiar e é destinado a habilitar um mecanismo de correção

de erros da unidade de mitigação de erros se a medida de vulnerabilidade for maior que um segundo limiar.

23. Sistema, de acordo com a reivindicação 22, caracterizado pelo fato de que o controlador é destinado a receber uma pluralidade de medidas de vulnerabilidade, cada uma para um dentre uma pluralidade de períodos de tempo e é destinado a gerar um sinal de controle com base na pluralidade de medidas de vulnerabilidade.

24. Sistema, de acordo com a reivindicação 22, caracterizado pelo fato de que a unidade de mitigação de erros deve ser configurada para executar tarefas de mitigação de não-erros quando desabilitados pelo controlador.

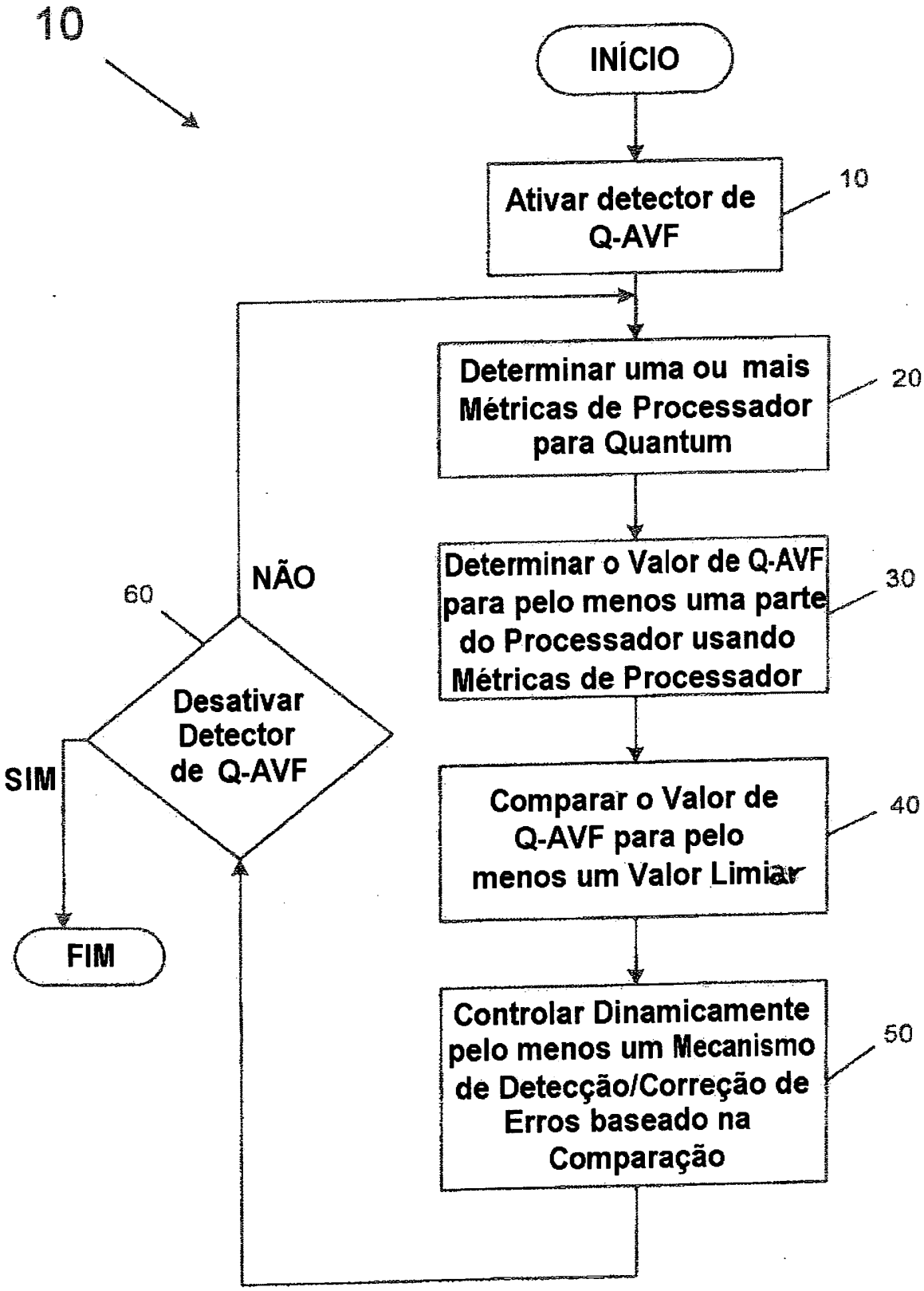

10
INÍCIO
Ativar detector de Q-AVF
10
Determinar uma ou mais Métricas de Processador para Quantum
20
Determinar o Valor de Q-AVF para pelo menos uma parte do Processador usando Métricas de Processador
30
Comparar o Valor de Q-AVF para pelo menos um Valor Limiar
40
Controlar Dinamicamente pelo menos um Mecanismo de Detecção/Correção de Erros baseado na Comparação
50
Desativar Detector de Q-AVF
60
NÃO
SIM
FIM

FIG. 1

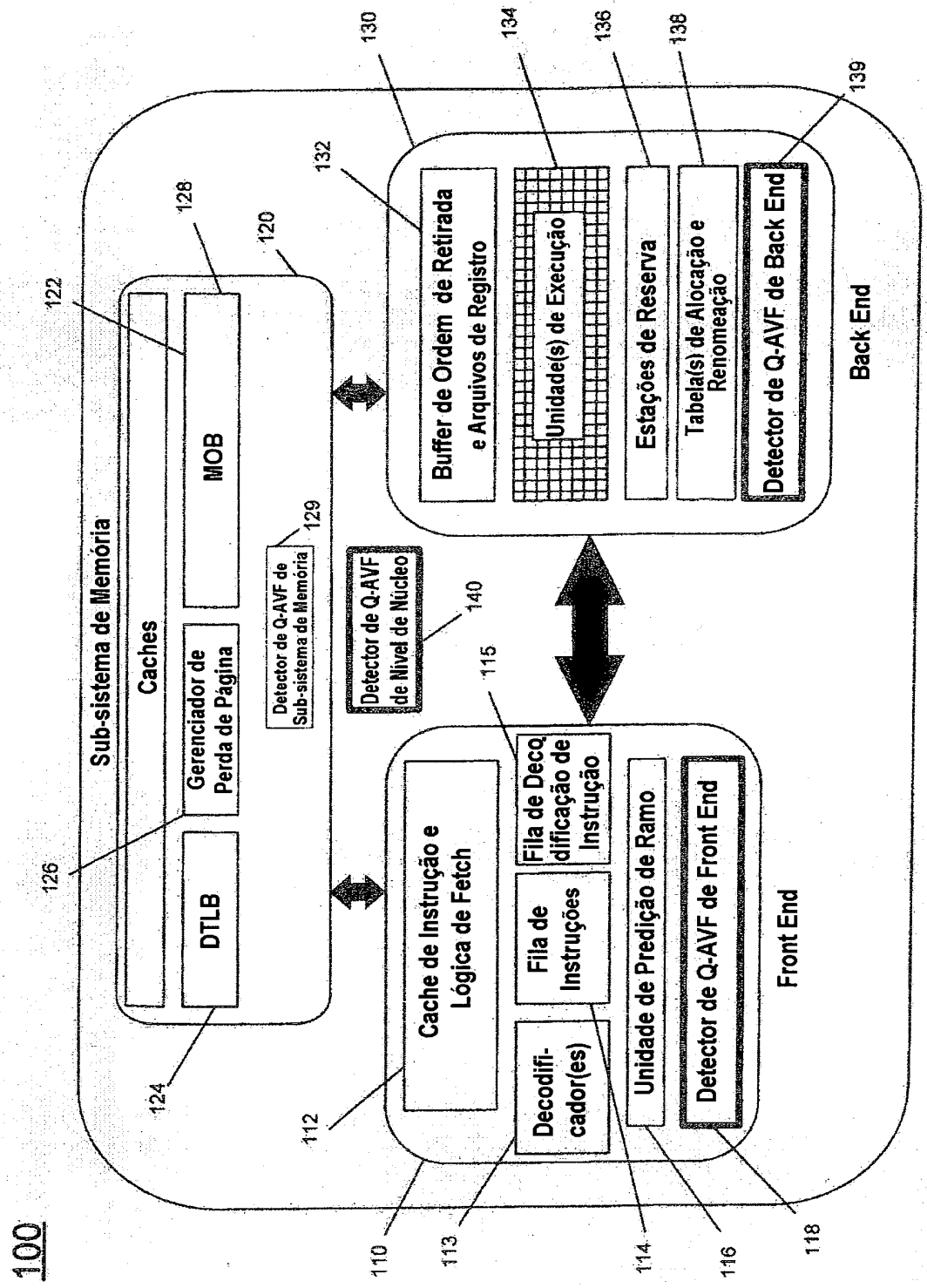

100
Sub-sistema de Memória
Caches
DTLB
Gerenciador de Perda de Página
MOB
Detector de Q-AVF de Sub-sistema de Memória
Detector de Q-AVF de Nível de Núcleo
Front End
Cache de Instrução e Lógica de Fetch
Decodifi-cador(es)
Fila de Instruções
Fila de Decodificação de Instrução
Unidade de Predição de Ramo
Detector de Q-AVF de Front End
Back End
Buffer de Ordem de Retirada e Arquivos de Registro
Unidade(s) de Execução
Estações de Reserva
Tabela(s) de Alocação e Renomeação
Detector de Q-AVF de Back End
110
112
113
114
115
116
118
120
122
124
126
128
129
130
132
134
136
138
139
140

FIG. 2

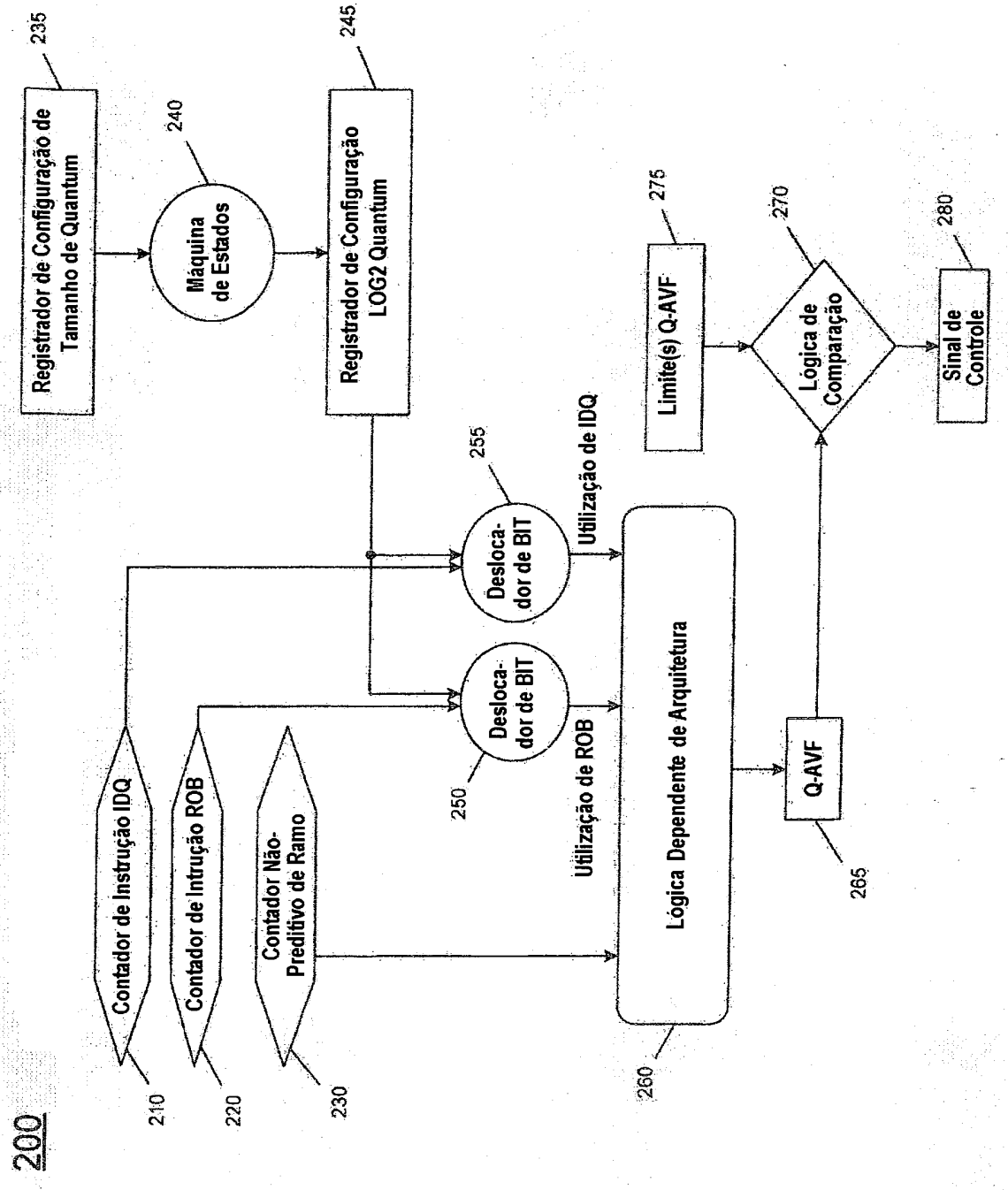

200
Registrador de Configuração de Tamanho de Quantum
235
Máquina de Estados
240
Registrador de Configuração LOG2 Quantum
245
Contador de Instrução IDQ
210
Contador de Intrução ROB
220
Contador Não-Preditivo de Ramo
230
Desloca-dor de BIT
250
Desloca-dor de BIT
255
Utilização de IDQ
Utilização de ROB
Lógica Dependente de Arquitetura
260
Q-AVF
265
Limite(s) Q-AVF
275
Lógica de Comparação
270
Sinal de Controle
280

FIG. 3

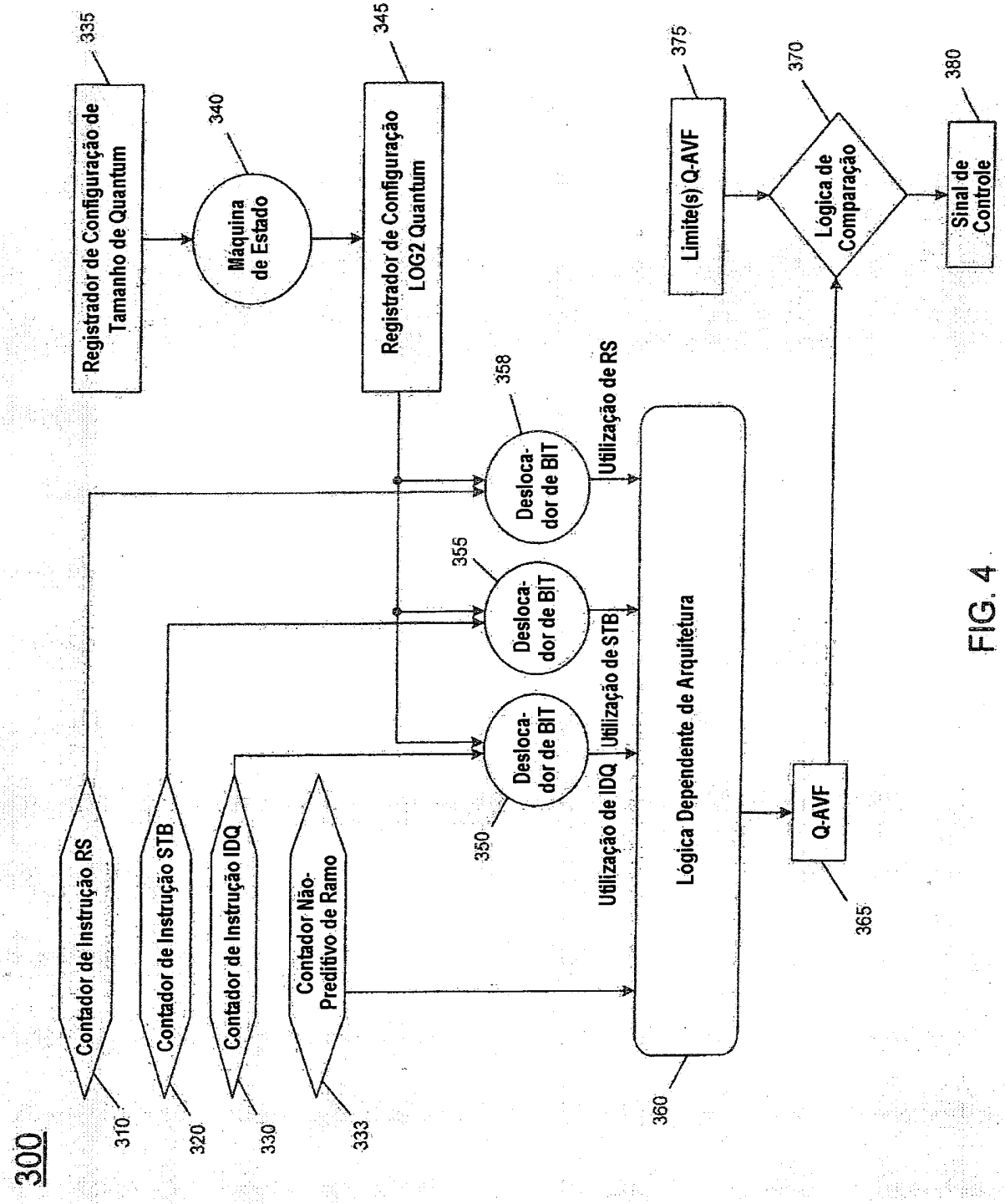

300
Registrador de Configuração de Tamanho de Quantum
335
Máquina de Estado
340
Registrador de Configuração LOG2 Quantum
345
Limite(s) Q-AVF
375
Lógica de Comparação
370
Sinal de Controle
380
Contador de Instrução RS
310
Contador de Instrução STB
320
Contador de Instrução IDQ
330
Contador Não-Preditivo de Ramo
333
Desloca-dor de BIT
358
Desloca-dor de BIT
355
Desloca-dor de BIT
350
Utilização de RS
Utilização de STB
Utilização de IDQ
Lógica Dependente de Arquitetura
360
Q-AVF
365

FIG. 4

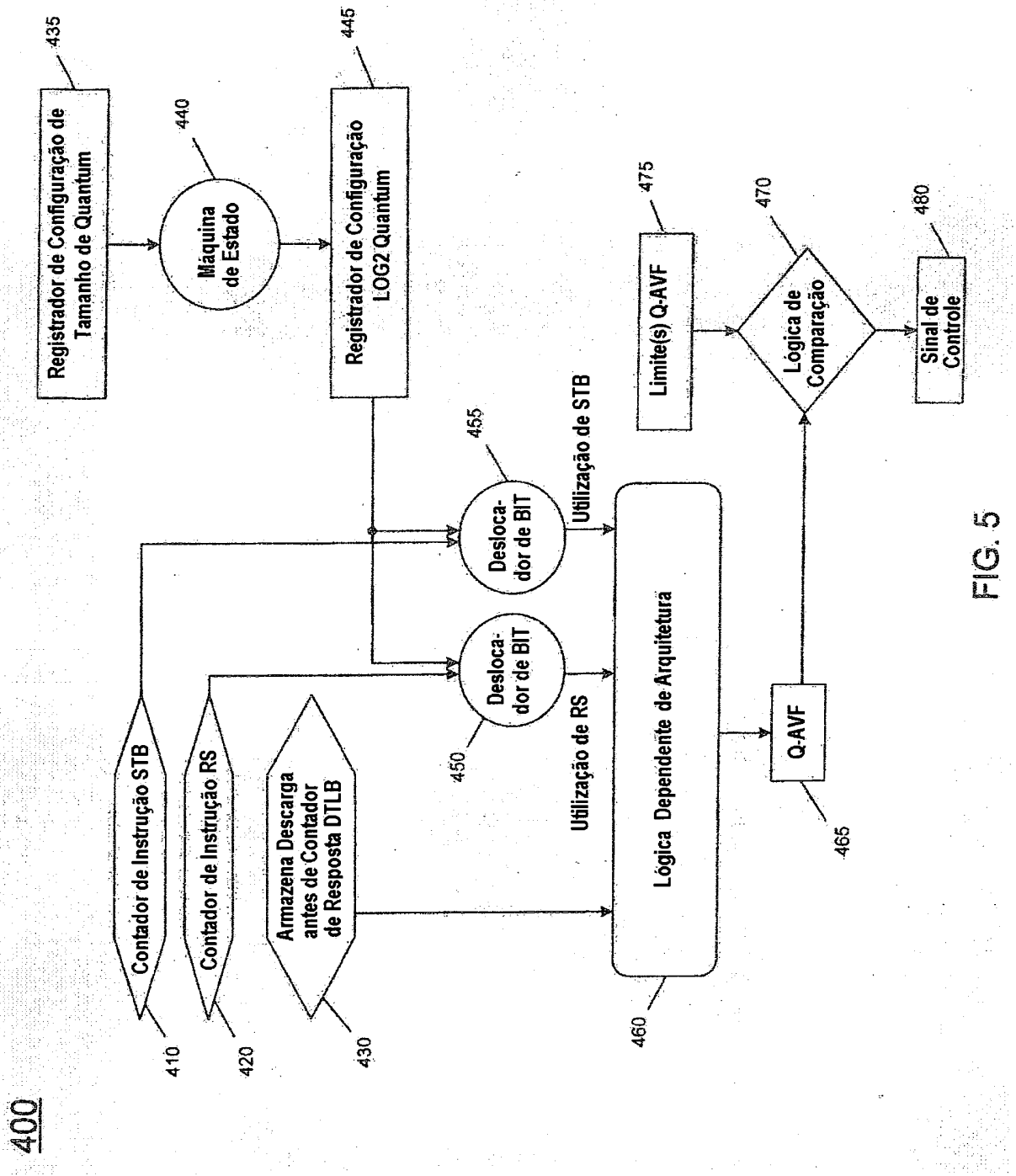

400
Registrador de Configuração de Tamanho de Quantum
435
Máquina de Estado
440
Registrador de Configuração LOG2 Quantum
445
Contador de Instrução STB
410
Contador de Instrução RS
420
Armazena Descarga antes de Contador de Resposta DTLB
430
Desloca-dor de BIT
450
Desloca-dor de BIT
455
Utilização de STB
Utilização de RS
Lógica Dependente de Arquitetura
460
Q-AVF
465
Limite(s) Q-AVF
475
Lógica de Comparação
470
Sinal de Controle
480

FIG. 5

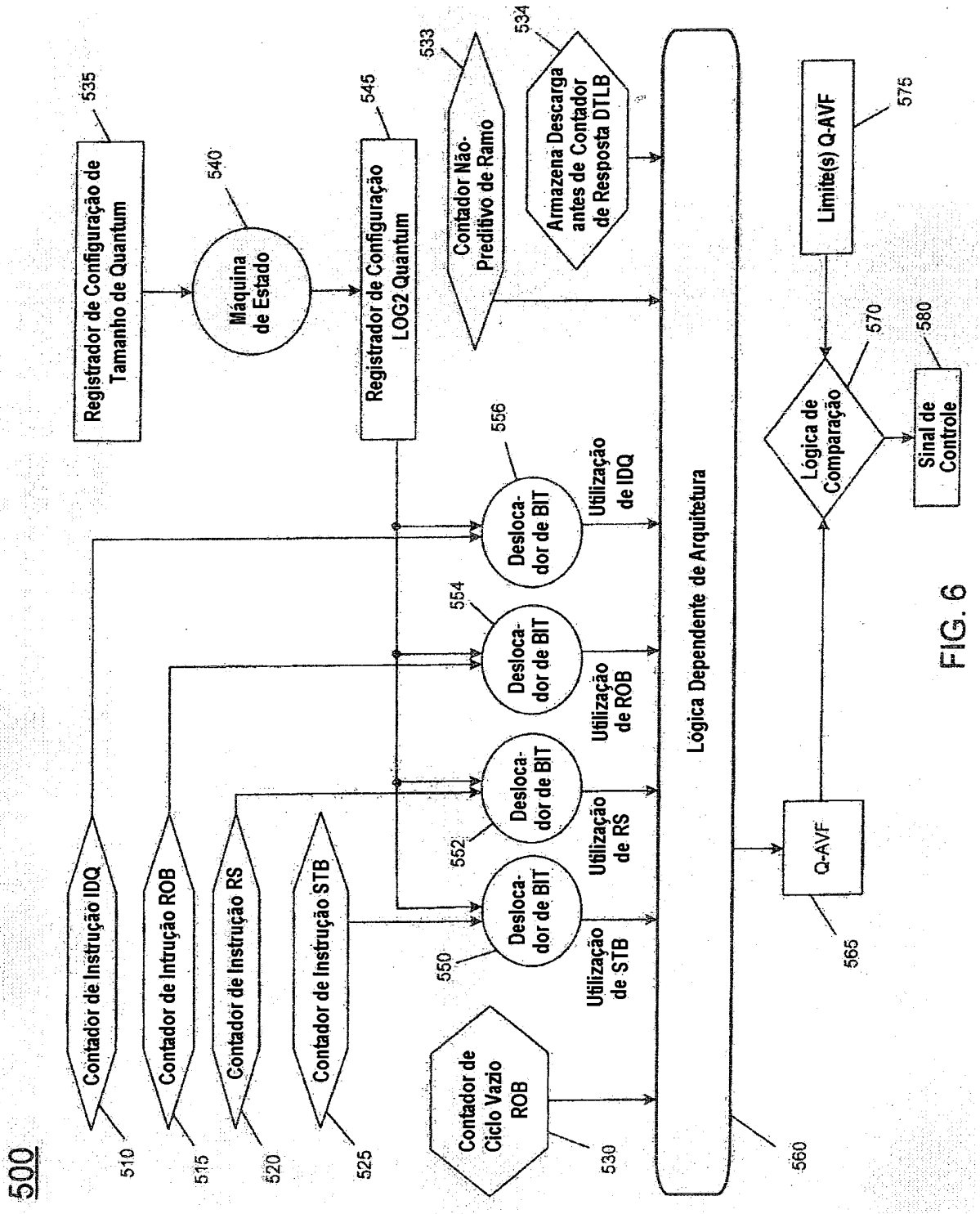

500
Registrador de Configuração de Tamanho de Quantum
535
Máquina de Estado
540
Registrador de Configuração LOG2 Quantum
545
Contador Não-Preditivo de Ramo
533
Armazena Descarga antes de Contador de Resposta DTLB
534
Limite(s) Q-AVF
575
Contador de Instrução IDQ
510
Contador de Intrução ROB
515
Contador de Instrução RS
520
Contador de Instrução STB
525
Contador de Ciclo Vazio ROB
530
Desloca-dor de BIT
550
Desloca-dor de BIT
552
Desloca-dor de BIT
554
Desloca-dor de BIT
556
Utilização de STB
Utilização de RS
Utilização de ROB
Utilização de IDQ
Lógica Dependente de Arquitetura
560
Q-AVF
565
Lógica de Comparação
570
Sinal de Controle
580

FIG. 6

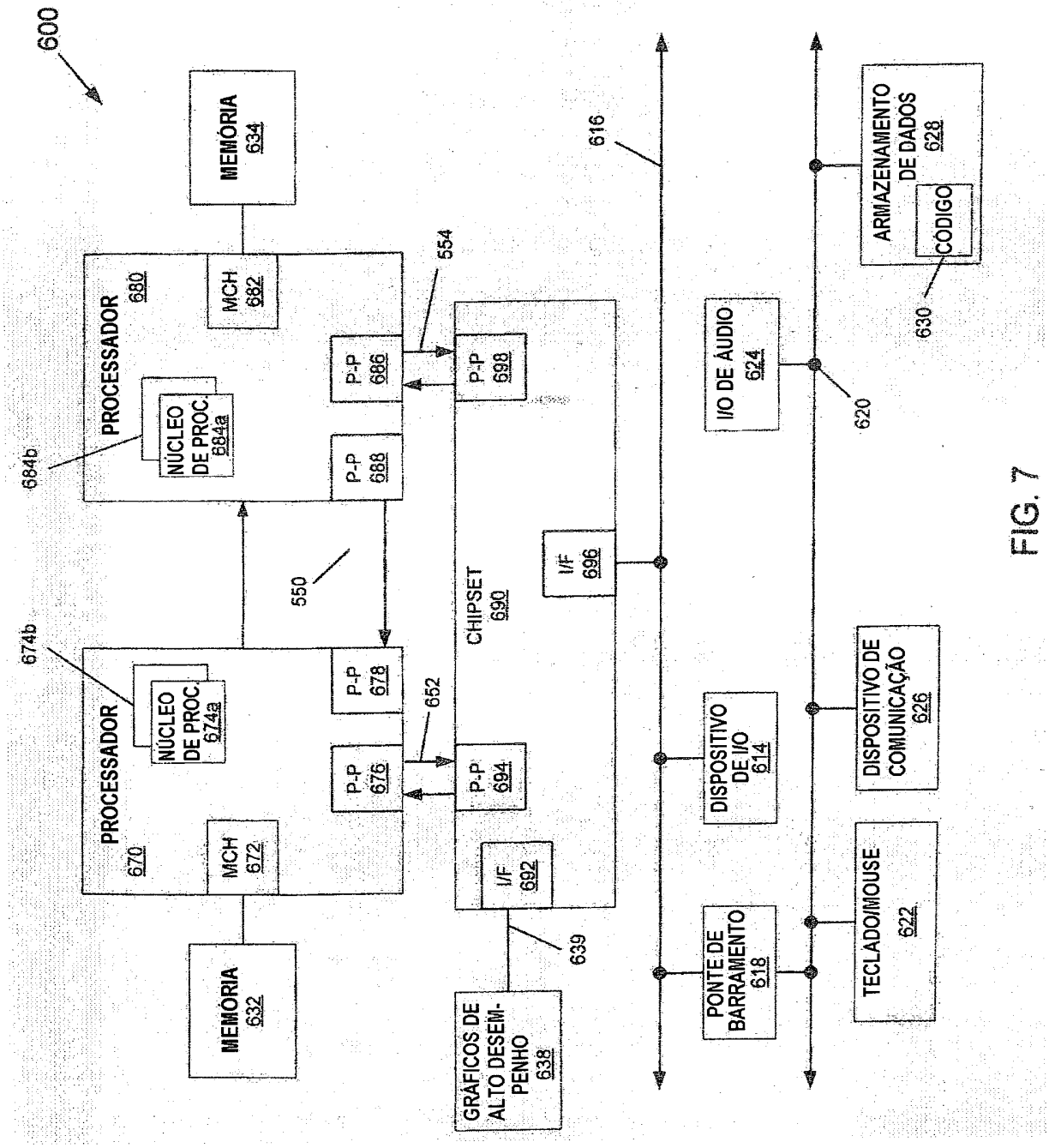

600
MEMÓRIA 634
PROCESSADOR 680
MCH 682
NÚCLEO DE PROC. 684a
684b
P-P 686
P-P 688
554
P-P 698
550
CHIPSET 690
I/F 696
616
I/O DE ÁUDIO 624
620
ARMAZENAMENTO DE DADOS 628
CÓDIGO 630
674b
PROCESSADOR 670
NÚCLEO DE PROC. 674a
P-P 678
652
P-P 676
P-P 694
MCH 672
I/F 692
639
MEMÓRIA 632
GRÁFICOS DE ALTO DESEMPENHO 638
DISPOSITIVO DE I/O 614
DISPOSITIVO DE COMUNICAÇÃO 626
PONTE DE BARRAMENTO 618
TECLADO/MOUSE 622

FIG. 7

RESUMO

"DETECÇÃO DE VULNERABILIDADE DE ARQUITETURA DE RECURSOS DE PROCESSADOR"

A presente invenção provê, em uma modalidade, um detector de quantum para detectar uma medida de vulnerabilidade para um processador baseado em uma métrica de processador, cada uma associada à operação de uma estrutura de processador durante um quantum, junto com um controlador para controlar uma unidade de mitigação de erro com base na medida de vulnerabilidade. Outras modalidades são descritas e reivindicadas.