UM GERENCIADOR PARA UM MICROCOMPUTADOR TOLERANTE A FALHAS PARA CONTROLE EM TEMPO REAL

(*) Helano de Sousa Castro

Este trabalho descreve a definição do software para um microcomputador tolerante a falhas para controle em tempo real. Este software, denominado Gerenciador, está dividido em várias tarefas que são executadas dentro de um ciclo de computação. O Gerenciador auxilia o hardware na implementação das estratégias de tolerância a falhas. Além disso, ele é responsável por tornar transparente a redundância do sistema para o usuário.

1. INTRODUÇÃO

O avanço da tecnologia dos semicondutores tem favorecido enormemente à utilização de sistemas de microcomputadores dedicados, sobretudo em aplicações de controle (p. ex.: controle de tráfego aéreo, sistemas de defesa militar

(*) Engenheiro eletricista (UFCe, 1982), mestre em Sistemas de Computação (PUC/RJ, 1985); Sistemas Distribuídos, Computação Tolerante a Falhas: Núcleo de Computação Eletrônica da ETFCE

etc.). No entanto, na medida em que seu emprego tem aumentado nestas áreas, surge também a preocupação com o problema da confiabilidade.

As conseqüências de uma eventual falha e seus reflexos em um sistema, variam de uma aplicação para outra. Desta forma, os requisitos de confiabilidade e disponibilidade variam também conforme à aplicação. Em algumas situações, a interrupção do serviço oferecido poderia representar apenas um inconveniente. Em outros casos, no entanto, isto poderia acarretar em grandes prejuízos materiais, chegando ao extremo, onde vidas humanas poderiam ser perdidas. Um requisito típico em aplicações espaciais é que com uma probabilidade de 95 a 99 por cento, a capacidade de computação ainda exista após 5 a 10 anos (2). Isto implica em um tempo médio entre falhas (MTBF) de 100 a 1000 anos.

Neste sentido, o objetivo deste trabalho é descrever o núcleo do software para um microcomputador tolerante a falhas para controle em tempo real, cujo hardware já se encontra implementado e testado como parte de um trabalho anterior (CAST 85).

2. REQUISITOS DO SISTEMA

O hardware implementado partiu, na verdade, de algumas especificações básicas, que assim influenciaram nas decisões de seu projeto. Estes requisitos exigiam que o sistema tivesse as seguintes características:

a) Composto por dois canais (duplex);
b) Seguro a falhas;
c) Inexistência de uma unidade central de gerenciamento;
d) Deve esforçar-se para sobreviver a falhas transientes e permanentes;
e) Redundância utilizada deve ser transparente ao usuário.

Os aspectos envolvidos na escolha por uma estrutura duplex deveu-se basicamente a dois fatores: confiabilidade e custo. Desta forma, o sistema é composto por dois subsistemas, denominados canais, baseados nos microprocessadores Z80A da Zilog e 8085 da Intel. A decisão para utilização de dois canais distintos reflete a opção por uma estrutura dissimilar, a fim de reduzir a probabilidade de erros de modo comum.

O sistema deve assegurar que sua saída seja correta, a menos que uma indicação de erro seja apontada (seguro a falhas). Desta forma, falhas simples, antecipadas ou não, devem ser detectadas.

A exigência, pela inexistência de uma unidade central de gerenciamento, esta intimamente relacionada com o problema da confiabilidade. Elementos centralizadores comprometem a confiabilidade de um sistema. Assim, todo o gerenciador deve estar distribuído nos dois canais, possibilitando a sobrevivência do sistema, mesmo quando degradado. No caso da existência de elementos críticos ("hardcore"), sua confiabilidade deve ser bem maior que a de um canal básico.

Outra característica importante deste sistema corresponde à sua capacidade de degradação. Ele deve ser capaz de detectar falhas permanentes e transientes, tentando sobreviver a seus efeitos.

Finalmente, a transparência da redundância utilizada, quando da execução dos programas do usuário, deverá ser levada a cabo pelo gerenciador.

A idéia básica do sistema consiste em ter-se dois canais distintos trabalhando paralela e independentemente em cima dos mesmos dados de entrada. Após processarem estes dados de entrada, os canais devem trocar, através de uma comunicação intercanal, informações relativas ao resultado da computação. Em caso de discordância, o canal com a melhor figura de mérito será o responsável pela saída do sistema (conforme será visto mais adiante).

As saídas dos dois canais estão conectadas ao seletor de saída (figura 1). Cada canal pode controlar o seletor, além de ter condições de saber seu "status" a qualquer instante. Desta forma, o canal que apresentará a saída do sistema, terá, naquele momento, o controle do seletor, cuja saída, consiste na própria saída do sistema. Convém observar que a saída do sistema é realimentada para cada canal. Além disso, a saída de cada canal é realimentada para ele.

3. HARDWARE DO CANAL

Os canais são funcionalmente idênticos, embora sejam implementados utilizando-se componentes de fabricantes distintos. Na figura 2 está indicada a arquitetura de um canal, que, como pode-se observar, possui os seguintes recursos:

a) UCP de 8 bits;
b) Unidade de Detecção e Registro de Erro de Paridade;
c) 4k x 9 de RAM (8 bits de dados mais 1 bit de paridade);
d) 4k de EPROM;
e) Unidade de Detecção e Registro de Erro de Limite de Endereço;
f) Unidade de Comunicação Intercanal;
g) Unidade de Temporização;
h) Unidade de Comunicação com o Sistema de Desenvolvimento (MCPUC - 80);
i) Registradores de Comando e Status.

As Unidades de Detecção e Registro de Erro de Paridade e Limite de Endereço indicam, respectivamente, uma condição de erro de paridade na memória (RAM) e erro devido a

um acesso a um endereço de memória fora de limites pré-estabelecidos. A Unidade de Comunicação Intercanal é responsável pela comunicação entre os dois canais. Esta é uma unidade crítica na medida em que a comparação dos resultados computados por cada canal, supõe que estes resultados tenham sido trocados entre eles, o que se dá via esta unidade. Além disso, a sincronização dos canais é feita via troca de mensagens, que também é feita através desta unidade. Devido a isso, esta unidade foi duplicada a fim de melhorar sua confiabilidade.

A Unidade de Temporização possui temporizadores que executam o papel de "watch-dog timers" e geração de uma base de tempo. Um destes temporizadores é utilizado para detecção de erro por "time-out" na execução de um ciclo de computação. Um outro temporizador é utilizado na geração de uma base de tempo que corresponde ao tempo de execução de cada tarefa dentro do ciclo. Um terceiro temporizador é utilizado para detectar erros por "time-out" durante a comunicação intercanal.

Os registradores de status são empregados no registro de erros, enquanto que os registradores de comando são utilizados no controle das outras unidades do canal. Uma descrição mais detalhada do Hardware do canal pode ser encontrada em (CAST 85).

4. O GERENCIADOR

O hardware descrito, foi especificado para trabalhar sob um software que o complementa na função de implementação das estratégias de tolerância a falhas. Este software, que deverá também promover uma transparência da redundância do sistema ao usuário, é descrito a seguir, e é denominado de Gerenciador.

Todo o Gerenciador deve encontrar-se distribuído nos canais, compatível assim com a especificação de que não deve existir uma unidade central de gerenciamento. Assim, cada canal consiste de um sub-sistema independente capaz de exercer o controle sobre o sistema.

O Gerenciador está dividido em tarefas, cada uma das quais é responsável por uma determinada função. As tarefas são executadas ciclicamente dentro de um ciclo de computação, de acordo com um escalonador de tarefas. O próprio programa do usuário seria executado como sendo uma tarefa. Cada uma destas tarefas devem ser executadas dentro de um intervalo de tempo pré-determinado (regulado de acordo com a programação de um dos temporizadores), ao final do qual, caso a tarefa não tenha sido completada, uma condição de erro por "time-out" será indicada. Entre algumas tarefas que compõem o ciclo, pode-se citar:

a) Iniciação;
b) Leitura dos dados de entrada;
c) Comparação dos dados de entrada;
d) Comunicação Intercanal;
e) Rotinas de Teste;
f) Execução da saída do sistema, etc.

Quando a execução de uma tarefa for concluída, um "flag" que indica tarefa cumprida deve ser "setado", como parte da própria tarefa. O escalonador de tarefas, ao assumir o controle (depois de esgotado o tempo destinado àquela tarefa), deve verificar o "flag" a fim de saber se a tarefa foi realmente concluída. Caso o "flag" não esteja setado, uma condição de erro por "time-out" será indicada, e o controle

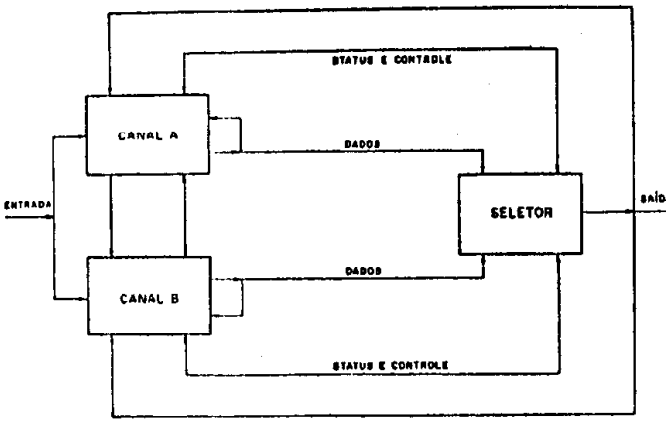

Fig 1. Arquitetura do Sistema

é passado para o gerenciador de erro, que deverá, após tentar uma recuperação do erro, sincronizar o canal com o outro canal. Caso o "flag" tenha sido "setado", o escalonador deve executar os seguintes procedimentos:

a) Resetar o "flag";
b) Incrementar um contador de tarefas (mantém a informação de qual será a próxima tarefa a ser executada. Esta informação é utilizada pelo escalonador de tarefas);
c) Verificar se todas as tarefas (do ciclo atual) já foram executadas; em caso negativo, ele deve passar para o procedimento (e), em caso positivo, ele deve zerar o contador de tarefas e passar para o próximo procedimento;
d) Efetuar a sincronização de ciclo;
e) Carregar o temporizador com o tempo para execução da próxima tarefa (isto pode ser feito pelo próprio hardware a fim de liberar mais o software) e transferir o controle para ela.

A necessidade de uma sincronização de ciclo decorre do fato de que, como cada canal possui sua própria base de tempo, existirá a propagação de uma pequena diferença entre estas bases que, caso não fosse corrigida periodicamente, poderia atingir valores tais que, uma sincronização não seria mais possível (LEIT 85).

Em caso de falha, o canal deve, após a detecção da mesma, sincronizar-se novamente com o outro canal, que poderá passar-lhe os parâmetros atuais, como parte da estratégia da recuperação. No Gerenciador estão previstas a implementação de rotinas de diagnóstico, que terão como função, efetuar testes periódicos nas várias unidades do hardware do sistema (Unidade de Detecção de Erro de Paridade etc.) Estes testes são importantes porquanto parte do esquema de detecção de falhas baseia-se nestas unidades.

Após cada erro detectado no canal, e execução dos procedimentos de detenção e recuperação, um contador de erros (mantido pelo gerenciador de erros) é incrementado de acordo com o tipo de erro. Este contador é utilizado na avaliação do desempenho do canal, que consiste da sua figura de mérito. É baseado nesta figura de mérito que é decidido qual canal apresentará a saída do sistema. Além disso, uma falha permanente poderá ser assumida caso o valor do contador ultrapasse um valor pré-determinado (após sucessivas tentativas de recuperação). As figuras de mérito de cada canal são trocadas entre eles, como parte de uma tarefa, a fim de que um canal conheça o desempenho do outro, e assim

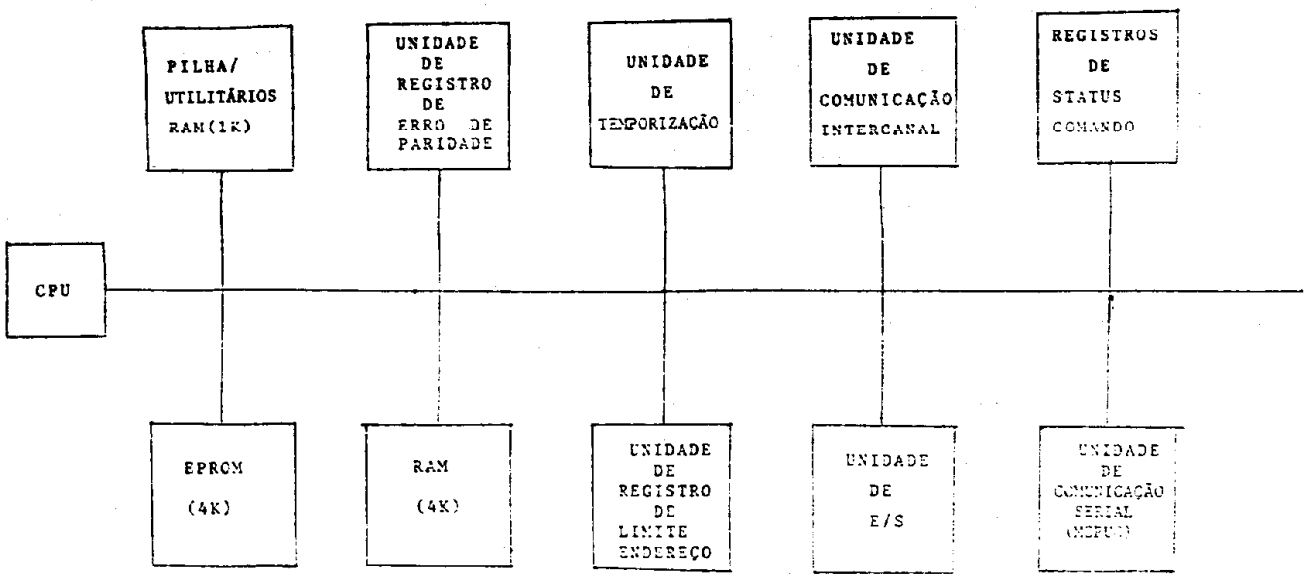

Fig 3 - Estrutura do Canal

possa ser estabelecido qual deles apresentará a saída do sistema.

A tarefa de iniciação consiste em transmitir o Gerenciador que se encontra na EPROM para a RAM, onde testes de paridade podem ser efetuados; nesta transferência é empregada uma palavra de teste (CRC, p. ex.) que deve ser incluída ao final do Gerenciador, e que é verificada ao final da transferência. A seguir é feito um teste das condições do canal, a fim de determinar se ele encontra-se pronto para entrar em operação (testes estes realizados nas diversas unidades do canal). Após estes procedimentos, uma outra tarefa será encarregada de ler os dados de entrada, os quais serão comparados com os lidos pelo outro canal. Os resultados na saída também são comparados por software. Caso os resultados computados por cada canal não coincidam, a figura de mérito será o fator decisivo na determinação de qual canal será responsável pela saída do sistema. Caso haja concordância entre os canais, um canal previamente escolhido, apresentará a saída. Neste caso, se ao comparar as figuras de mérito descubra-se que elas são diferentes, pode-se invocar uma tarefa que efetua uma re-execução do programa, como medida de segurança.

Caso um dos canais esteja fora de operação, devido a uma falha permanente, o sistema irá degradar-se e ficará com apenas um canal. Mesmo assim, algum tipo de comparação pode ser feita a fim de detectar-se alguma falha, transiente, por exemplo. Para isto, basta executar uma tarefa na qual uma nova execução do programa do usuário (possivelmente utilizando um algoritmo diferente) é feita, e comparar este último resultado com o obtido na primeira computação. Este procedimento pode ser repetido mais vezes, a fim de obter-se uma maior segurança. Aqui alguns compromissos devem ser assumidos, como por exemplo, o tempo que os resultados devem estar presentes na saída contra o número de re-execuções. O tempo de execução das tarefas, e conseqüentemente do ciclo, pode ser ajustado de acordo com a aplicação, através da programação dos temporizadores. Além disso, estão previstas a inclusão de pontos de recuperação, como parte da estratégia de recuperação de erros.

5. CONCLUSÃO

O Gerenciador corresponde à segunda parte de um trabalho iniciado no grupo de sistemas de computação, na área de tolerância a falhas da PUC-RJ, que consistiu na definição e implementação de um microcomputador tolerante a falhas para controle em tempo real. O software descrito, no entanto, não se encontra totalmente concluído, tendo alguns pontos a serem definidos; a parte de tratamento de eventos assíncronos, por exemplo, cogita-se em colocá-la como parte de uma tarefa. O autor gostaria de agradecer ao professor Julius C. B. Leite, do Departamento de Engenharia Elétrica da PUC-RJ, pelo incentivo no prosseguimento deste trabalho.

6. BIBLIOGRAFIA

(CAST 85) HELANO DE SOUSA CASTRO, *"Um Microcomputador Tolerante a Falhas para Controle em Tempo Real"*, dissertação de mestrado, Depto. de Engenharia Elétrica, PUC, RJ/1985.

(LEIT 85) J. C. B. LEITE, HELANO S. CASTRO, *"Um Microcomputador de Alta Disponibilidade"*, I Simpósio de Sistemas de Computadores Tolerantes a Falhas, Instituto Nacional de Pesquisas Espaciais (INPE), S. J. dos Campos 1985.

(RIAL 80) A. RIAL F., *Dissimilar Redundancy in Fault - Tolerant Microcomputer Systems"*, PhD. Thesis, Dept. of Electrical Eng. and Electronics University of Manchester, Institute of Science and Technology, outubro de 1980.

(DEPL 81) P. G. DEPLENDGE, *"Fault Tolerant Computer Systems"*, IEE Proc., Vol 128, part - A, No. 4, maio de 1981, p. 257 - 272.

(MESQ 79) A. C. MESQUITA JR., *"Tolerância a Falhas por Diversificação"* Dissertação de Mestrado, Depto. Eng. Elétrica, PUC/RJ, Agosto de 1974.