MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO
AUTHORIZATION FOR PUBLICATION

PALAVRAS CHAVES/KEY WORDS

AUTORES / AUTHORS

TOLERÂNCIA A FALHAS - REDUNDÂNCIA - SISTEMAS DISTRIBUÍDOS - ARQUITETURA DE COMPUTADORES

AUTORIZADA POR/AUTHORIZED BY

Marco Antonio Raupp
Diretor Geral

AUTOR RESPONSAVEL
RESPONSIBLE AUTHOR

Alderico R.de Paula Jr.

DISTRIBUIÇÃO/DISTRIBUTION

- ☐ INTERNA / INTERNAL
- ☒ EXTERNA / EXTERNAL
- ☐ RESTRITA / RESTRICTED

REVISADA POR / REVISED BY

Klaus J. Johansen

CDU/UDC

681.3.02

DATA / DATE

Novembro 1987

PUBLICAÇÃO Nº
PUBLICATION NO

INPE-4433-RTR/110

ORIGEM
ORIGIN

DCA

PROJETO
PROJECT

SUBORD

TÍTULO/TITLE

ARQUITETURA DE SISTEMAS DE COMPUTAÇÃO TOLERANTES A FALHAS

Nº DE PAG. / NO OF PAGES

52

ULTIMA PAG. / LAST PAGE

46

VERSÃO / VERSION

01

Nº DE MAPAS / NO OF MAPS

AUTORES/AUTHORSHIP

Alderico Rodrigues de Paula Junior

RESUMO - NOTAS / ABSTRACT - NOTES

O objetivo deste trabalho é discutir aspectos gerais de arquiteturas de sistemas de computação tolerantes a falhas. Inicialmente são analisados sistemas de computação com uma, duas e três unidades de processamento, que operam sincronizadamente ou não e executam a mesma tarefa. A análise é, estendida para sistemas de computação distribuídos. Em seguida, a confiabilidade e a disponibilidade de diversas arquiteturas são avaliadas, e critérios para a seleção da arquitetura para uma dada aplicação são discutidos. Finalmente, exemplos de sistemas de computadores tolerantes a falhas são apresentados.

OBSERVAÇÕES / REMARKS

Este trabalho foi apresentado no II Simpósio em Sistemas de Computadores Tolerantes a Falhas, realizado em 24 a 28 de agosto de 1987, UNICAMP, Campinas, SP.

ABSTRACT

The goal of this work is to discuss general aspects of fault tolerant computer system architectures. Initially, computer systems having one, two and three processing units are analysed, which operate synchronously or not and execute the same task. Then, the analysis is extended to distributed computer systems. Next, the reliability and availability of different architectures are evaluated and criteria to select architecture for a specific application are discussed. Finally, examples of fault tolerant computer systems are presented.

SUMÁRIO

LISTA DE FIGURAS

Pág.

1 - INTRODUÇÃO

Sistemas de computadores vêm sendo utilizados de forma crescente no controle de sistemas críticos, tais como no controle de aeronaves, de sistemas ferroviários e de usinas nucleares. Para tais sistemas uma simples falha no sistema de computação poderá causar prejuízos econômicos e perda de vidas humanas.

A seleção da arquitetura do sistema de computação para uma aplicação específica dependerá da especificação dos requisitos de confiabilidade, segurança e disponibilidade. Na área de aeronáutica, por exemplo, a probabilidade de o sistema de computação de aeronaves falhar deverá ser menor que 10 elevado a - 9 em vôos de 10 horas de duração. Já em aplicações espaciais, um requisito geralmente utilizado na especificação de satélites é que o sistema de computação deverá suportar qualquer falha simples.

Ao projetar um sistema de computação, deve-se ter em mente que sempre existirá uma sequência de faltas que levará o sistema a um estado falho. Portanto, o objetivo do projetista deve ser minimizar a probabilidade de ocorrência de tais sequências de faltas.

As duas técnicas geralmente utilizadas para aumentar a confiabilidade de sistemas de computação são:

- utilização de componentes de alta confiabilidade (evitar faltas);

- utilização de redundância (tolerar faltas).

Embora a taxa de falhas dos componentes eletrônicos tenha diminuído com o desenvolvimento tecnológico, ainda não foi possível a fabricação de componentes livres de falhas. Devido ao alto preço dos componentes de alta confiabilidade, muitas vezes a utilização de redundância com componentes de menor qualidade propicia sistemas mais baratos com os

mesmos requisitos de qualidade. Para aplicações críticas, devem ser utilizados componentes de alta confiabilidade, bem como redundância.

A redundância é utilizada tanto para a deteção ou mascaramento de erros quanto para substituir módulos falhos de um sistema. É utilizada nas seguintes formas:

- redundância temporal;

- redundância ao nível de programa;

- redundância ao nível de circuito.

A redundância temporal consiste na repetição da execução de uma mesma tarefa, utilizando o mesmo programa ou programas equivalentes. Os resultados são comparados visando a deteção de erros durante a execução dos programas.

Ao nível de programa, a redundância geralmente consiste na utilização de instruções adicionais que visam verificar a consistência dos dados calculados.

Ao nível de circuito, a redundância é utilizada para a deteção de erros ou para substituir módulos defeituosos. Pode ser implementada tanto a nível de componentes, como ao nível de módulos ou ao nível de unidade de processamento.

A tolerância a falhas nos sistemas de computação é geralmente implementada em quatro fases. A primeira fase consiste na deteção do erro causado por um defeito interno de circuito ou de projeto. A fase seguinte consiste na avaliação do estrago causado e no confinamento da área danificada. A terceira fase visa recuperar o sistema, substituindo ou isolando o módulo danificado e restaurando o estado interno do sistema. Finalmente, a última fase tem como objetivo fazer o sistema retornar à operação normal.

Este trabalho será organizado da seguinte forma: na Seção 2 serão discutidas diversas arquiteturas para sistemas de computação tolerantes a falhas. A avaliação destas diferentes arquiteturas será efetuada na Seção 3 e, finalmente, na Seção 4 será apresentada a arquitetura de alguns computadores tolerantes a falhas.

2 - ARQUITETURA

Nesta seção serão discutidos aspectos gerais das principais arquiteturas de computadores utilizadas em sistemas que requerem alta confiabilidade e/ou segurança e/ou disponibilidade. A análise será iniciada para arquiteturas simples que evoluem para sistemas complexos.

2.1 - SISTEMAS COM UMA UNIDADE DE PROCESSAMENTO

O objetivo desta seção é apresentar técnicas ao nível de circuito e ao nível de programa a serem utilizadas para aumentar a segurança de uma unidade de processamento, visando diminuir a probabilidade de esta unidade gerar atuações ou fornecer resultados errôneos devido a faltas internas. Será considerado na análise que a unidade de processamento está ligada a um supervisor que poderá ser um outro computador ou um operador.

Uma das técnicas mais simples para deteção de erros consiste em executar periodicamente programas de testes ou diagnose. Estes programas geralmente cobrem uma grande classe de defeitos internos, principalmente no processador e na memória.

Caso estes programas detetem algum mau funcionamento, eles enviam uma mensagem ao supervisor, o qual suspende a operação da unidade. Em alguns sistemas as rotinas de testes enviam mensagens que indicam o resultado do teste. Caso ocorram faltas internas que bloqueiem a execução dos programas de testes, o supervisor detetará que a unidade falhou pela ausência de mensagens do programa de testes.

Esta técnica não evita que os programas aplicativos gerem comandos errôneos entre a execução dos programas de testes e, geralmente, não deteta erros transitórios.

Para reduzir a probabilidade de que uma falta interna, ocorrida após a execução do programa de testes, gere resultados ou comandos errôneos, uma possível técnica a ser utilizada consiste em verificar se os resultados calculados estão dentro de limites aceitáveis. Esta técnica, contudo, tem a limitação de não detetar se dados dentro dos limites aceitáveis estão corretos ou não.

O próximo passo, que visa reduzir a probabilidade de geração de comandos errôneos, consiste na utilização de circuitos de deteção que operam simultaneamente com a execução dos programas. As três técnicas mais utilizadas são:

a) codificação da informação e deteção de palavras não códigos;

b) duplicação e comparação;

c) atuação e verificação.

A codificação de dados é geralmente utilizada quando estes não recebem transformações, como, por exemplo, na transferência de dados entre dois módulos ou no armazenamento. Os códigos utilizados na transferência são geralmente códigos de paridade de um bit, "checksum" e códigos cíclicos (CRC). O código de paridade de um bit consiste em adicionar um bit de paridade à palavra transmitida e verificar se esta foi recebida corretamente. Com um bit de paridade é possível detetar todos os erros simples na palavra. O código "checksum" consiste em adicionar uma palavra ao bloco de dados a ser transmitido, o qual representa o complemento da soma das palavras deste bloco. Este código é facilmente implementado por programação.

O CRC consiste na adição de uma palavra de verificação ao bloco a ser transmitido e é utilizado geralmente quando um bloco de dados é transmitido serialmente. Consiste na adição de uma palavra de verificação ao bloco a ser transmitido. A palavra de verificação representa o complemento do resto da divisão, em módulo dois, do polimônio formado do bloco de dados pelo polimônio gerador do código. A geração e a verificação da palavra adicional são facilmente implementadas por meio de registradores de deslocamentos.

Os códigos geralmente utilizados no armazenamento de dados são o de paridade de um bit e o código de Hamming modificado; o último permite a deteção de erros duplos e a correção de erros simples nas palavras lidas do módulo de armazenamento.

Dado que no módulo de processamento os dados recebem transformações, geralmente não se utilizam códigos para a deteção de erros, apesar de alguns códigos conservarem suas propriedades nas operações aritméticas. Nos módulos de processamento microprogramados é possível a utilização de códigos de deteção de erros na memória de microprogramação. Atualmente, grande parte dos módulos de processamento são implementados com microprocessadores, os quais geralmente não possuem circuitos de deteção de erros internos. Para estes tipos de módulos de processamento, as técnicas empregadas são a duplicação do microprocessador e a utilização de comparadores em todas ou em algumas linhas de saída de dados do microprocessador.

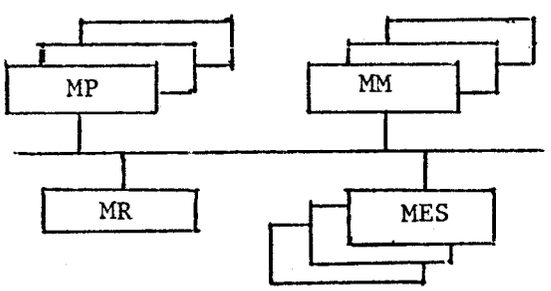
A deteção de erros nos módulos de entrada e saída consiste, em geral, na verificação dos dados transmitidos através da leitura dos registros de saída.

Os circuitos adicionados para deteção de erros aumentam a taxa de falhas da unidade de processamento, mas permitem a deteção de um grande número de falhas internas, o que diminui a probabilidade de a unidade gerar comandos ou fornecer dados errôneos.

2.2 - REDUNDANCIA INTERNA

A redundância interna a uma unidade de processamento é utilizada para deteção ou mascaramento de erros e para aumentar o tempo de sobrevivência desta unidade. A redundância pode ser implementada tanto ao nível de componentes quanto ao nível de módulos. Devido à dificuldade de chaveamento, a redundância ao nível de circuito só é utilizada em componentes críticos que possuem taxa de falhas bem maior que a dos demais componentes e geralmente é implementada na forma de redundância estática.

Por outro lado, a grande maioria dos módulos de uma unidade de processamento é interconectada através de barramentos de dados padrões, o que facilita a adição de módulos redundantes e o chaveamento para módulos reservas através da desenergização do módulo ativo e da energização do módulo reserva. Em geral, o circuito que faz a interface com o barramento deverá estar sempre energizado. O diagrama de bloco de uma unidade de processamento com redundância interna é apresentado na Figura 1.



MP - Módulo de Processamento
MM - Módulo de Memória
MES - Módulo de Entrada e Saída
MR - Módulo de Reconfiguração

Fig. 1 - Unidade de processamento com redundância interna.

A energização e a desenergização dos módulos podem ser controladas externamente por um supervisor, ou automaticamente através de um módulo interno de reconfiguração.

Um dos pontos críticos da redundância interna ao nível de módulos é que os módulos são geralmente conectados a um grande número de linhas de dados como, por exemplo, barramento (dutos) de dados, endereço e controle. No caso de falha do circuito que interliga o módulo com o barramento, este pode afetar uma das linhas do barramento. Este fato poderá levar a unidade a um estado falho. Um outro ponto crítico aparece quando um módulo de reconfiguração é utilizado para chaveamento automático. Este módulo também está sujeito a falhas, o que inviabiliza a reconfiguração do sistema. Portanto, o módulo de reconfiguração deverá ser também redundante e ter a taxa de falhas bem menor que a dos demais módulos.

2.3 - SISTEMAS COM DUAS UNIDADES DE PROCESSAMENTO

Os sistemas com duas unidades de processamento geralmente operam de duas formas distintas, dependendo da aplicação:

- uma unidade ativa e uma unidade desligada;

- duas unidades ativas.

2.3.1 - ARQUITETURA COM UMA UNIDADE ATIVA E UMA UNIDADE DESLIGADA

Nesta arquitetura geralmente é utilizado um circuito supervisor quando o chaveamento é automático. O diagrama de blocos desta configuração é apresentado na Figura 2.

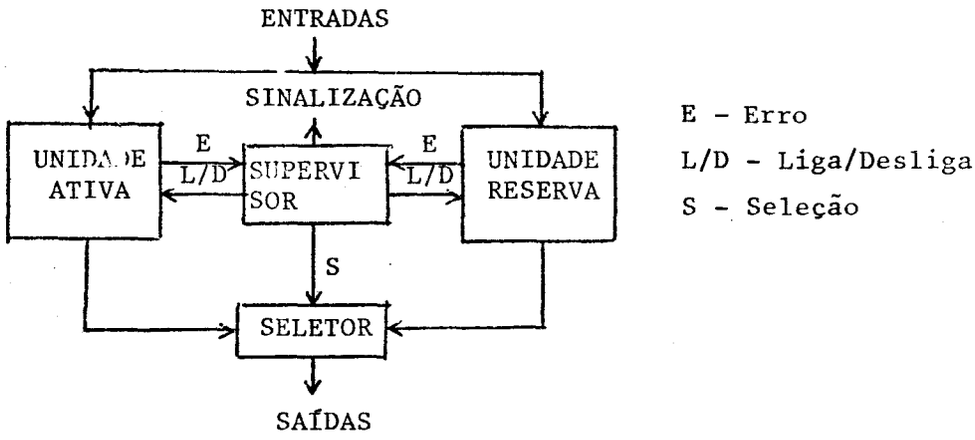


Fig. 2 - Sistemas com duas unidades de processamento e com chaveamento automático.

O circuito supervisor é responsável pelo chaveamento das unidades de processamento quando a unidade ativa falha. As duas técnicas geralmente utilizadas para sinalizar ao supervisor que a unidade ativa falhou são:

- relógio cão de guarda;

- indicação de erro.

O relógio cão de guarda é implementado no circuito supervisor através de um contador que é zerado periodicamente pela unidade de processamento ativa, quando a unidade está operando corretamente. Antes de enviar um comando para zerar o contador, a unidade ativa verifica o seu estado interno através da execução de um

programa de testes ou diagnose. Caso a unidade detete um defeito irrecuperável ou não seja capaz de executar o programa de testes, o comando para zerar o contador não é enviado. Portanto, o contador atingirá o limite de tempo programado, o que levará o supervisor a desenergizar a unidade ativa e a energizar a unidade reserva.

Na segunda técnica, as unidades de processamento possuem um conjunto de mecanismos implementados tanto por programação quanto por circuito, que sinalizam ao supervisor quando um erro irrecuperável é detetado. Ao receber a sinalização de falhas da unidade ativa, o supervisor comuta o sistema para a unidade reserva.

Quando o sistema não possui um supervisor, a unidade ativa aciona um alarme para indicar que um mau funcionamento foi detetado.

Os principais pontos críticos desta arquitetura são os seguintes:

a) O supervisor e os mecanismos para deteção de erros podem falhar, o que permite que a unidade de processamento gere comandos ou forneça resultados errôneos. Este problema pode ser minimizado através de circuitos supervisores redundantes e de circuitos de deteção de erros autotestáveis.

b) O sistema terá de ser reinicializado quando a comutação para unidade reserva for realizada. Caso haja necessidade de continuar a operação do ponto onde foi detetada a falha, faz-se necessário o uso de um sistema de armazenamento compartilhado pelas duas unidades, onde é armazenado periodicamente o estado atual do sistema.

2.3.2 - ARQUITETURA COM DUAS UNIDADES ATIVAS

Esta arquitetura se divide em duas classes:

- fortemente conectada;

- fracamente conectada.

Na arquitetura fortemente conectada, as duas unidades de processamento operam sincronizadamente, executando simultaneamente o mesmo programa. Geralmente as duas unidades possuem o mesmo relógio; os principais sinais, tais como de endereço, de dados, de controle e de entrada e saída, são comparados periodicamente. A Figura 3 apresenta o digrama de blocos desta arquitetura.

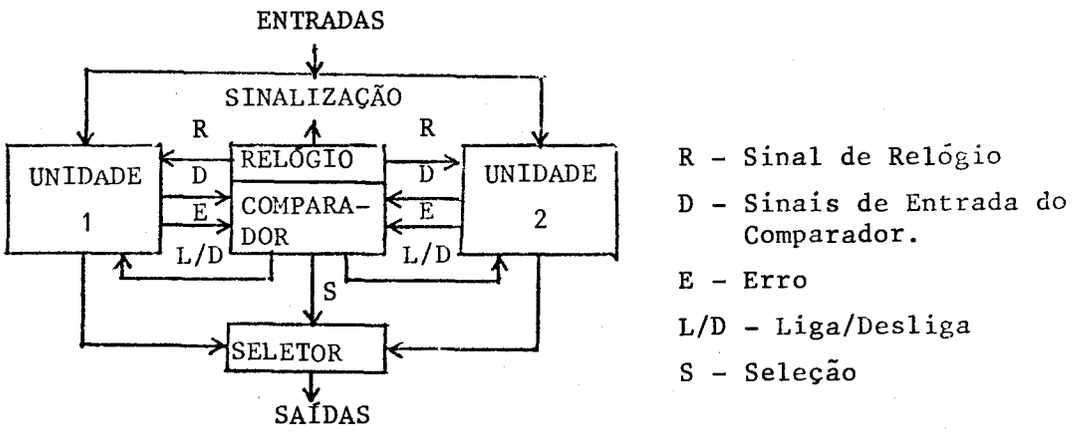


Fig. 3 - Arquitetura com duas unidades de processamento fortemente conectadas.

Ao detetar uma discrepância neste tipo de arquitetura, o circuito comparador inibe as saídas das unidades de processamento, as quais entram em fase de recuperação.

Geralmente a primeira fase da recuperação consiste em executar novamente a última tarefa. Caso haja sucesso, o erro é considerado como transitório e o sistema retorna à operação normal. Caso o erro se repita novamente, o problema consiste em detetar qual executados nas duas unidades para definir qual das unidades está falha. Uma vez, detetada a unidade falha, dependendo da aplicação, poder-se-à proceder:

a) ao desligamento seguro do sistema;

b) à continuação da operação com apenas uma unidade.

Neste último caso, o sistema opera de forma não segura e, portanto, erros internos poderão gerar comandos externos errôneos.

Os pontos críticos desta arquitetura são:

a) O comparador e o relógio são geralmente circuitos de falhas simples.

b) O comparador, ao detetar um erro, não identifica qual das unidades falhou. É necessário, portanto, executar programas de testes nas unidades e, consequentemente, a operação do sistema terá de ser suspensa temporariamente. Após a identificação da unidade falha, a unidade não- falha poderá continuar a operação de modo não-seguro ou, então, levar o sistema para um estado seguro e suspender a operação.

Na arquitetura fracamente conectada, as duas unidades de processamento possuem seu próprio relógio e não operam sincronizadamente. O diagrama desta arquitetura é apresentado na Figura 4.

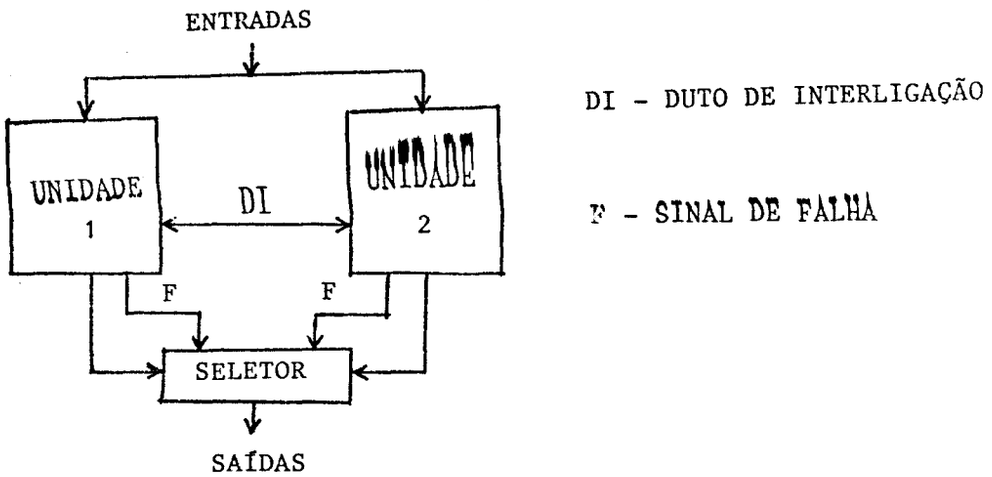


Fig. 4 - Arquitetura com duas unidades fracamente conectadas.

As duas unidades de processamento executam os mesmos programas ou programas equivalentes e, em pontos específicos do programa, trocam mensagens para comparar os dados calculados internamente com os dados calculados pela outra unidade. Os pontos selecionados para troca de mensagem são geralmente escolhidos após a aquisição de dados externos ou antes de uma atuação externa.

A vantagem da arquitetura fracamente conectada em relação à fortemente conectada consiste no fato de que na primeira não existem pontos de falhas simples, tais como o relógio e o circuito de comparação. A desvantagem é que na fracamente conectada a comparação é realizada por programa, e o sistema torna-se mais lento. Um outro problema que aparece nos sistemas fracamente concectados é que as aquisições de dados externos não são síncronas e, portanto, os dados externos podem ser diferentes, o que requer uma comparação por janela.

2.4 - SISTEMAS COM TRES UNIDADES DE PROCESSAMENTO

Como discutido na seção anterior, os dois principais problemas nos sistemas que utilizam duas unidades ativas são:

a) a determinação da unidade falha após a deteção de uma discordância nos dados comparados;

b) a operação de forma não-segura, caso o sistema degrade para uma unidade após a falha de uma das unidades ativas.

Para suplantar estes dois problemas é necessária a utilização de três unidades ativas. Nesta arquitetura, os dados gerados pelas três unidades são submetidos a um votador majoritário que mascara os possíveis erros gerados por uma dada unidade de processamento. Tal como na arquitetura com duas unidades, as unidades do sistema com três unidades ativas de processamento podem ser forte ou fracamente conectadas.

Nos sistemas fortemente conectados, denominados TMR, as três unidades trabalham sincronizadamente, recebendo sinais de um único relógio e executando os mesmos programas. Os sinais de saída destas unidades são submetidos a circuitos votadores como os apresentados na Figura 5.

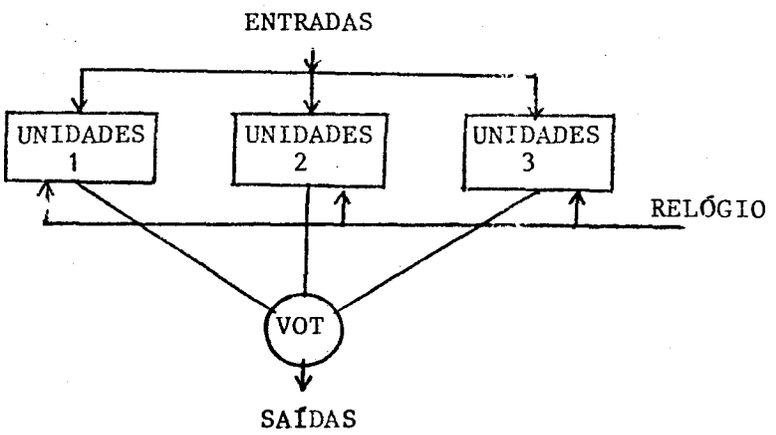
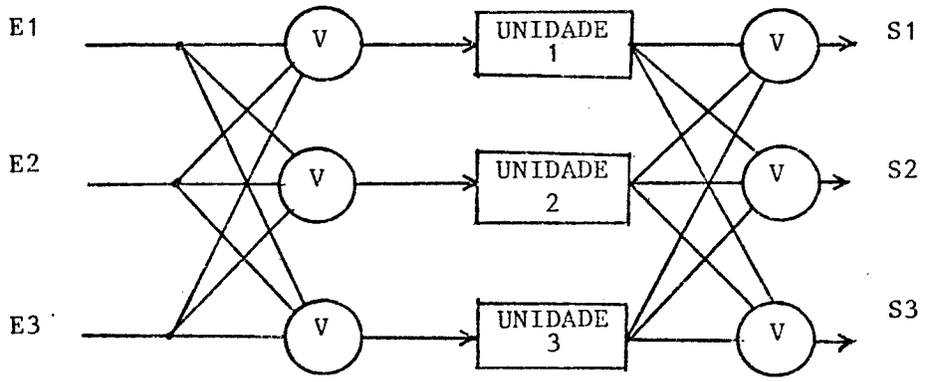


Fig. 5 - Arquitetura com três unidades fortemente conectadas.

Os pontos críticos desta arquitetura são o relógio e os votatores, os quais são pontos de falha simples. Esquemas com relógios múltiplos tolerantes a uma falha simples já foram desenvolvidos.

Para evitar que falhas nos votadores levem o sistema a um estado falho, votadores redundantes devem ser utilizados. A Figura 6 apresenta um esquema quando dados de entrada e saída são triplicados.



E1, E2, E3 - ENTRADAS

S1, S2, S3 - SAÍDAS

Fig. 6 - Arquitetura com dados de entrada e saída triplicados.

Para deteção de uma unidade falha, é necessária a utilização de circuitos comparadores que comparam os sinais de saída dos votadores com os seus sinais de entrada, como apresentado na Figura 7.

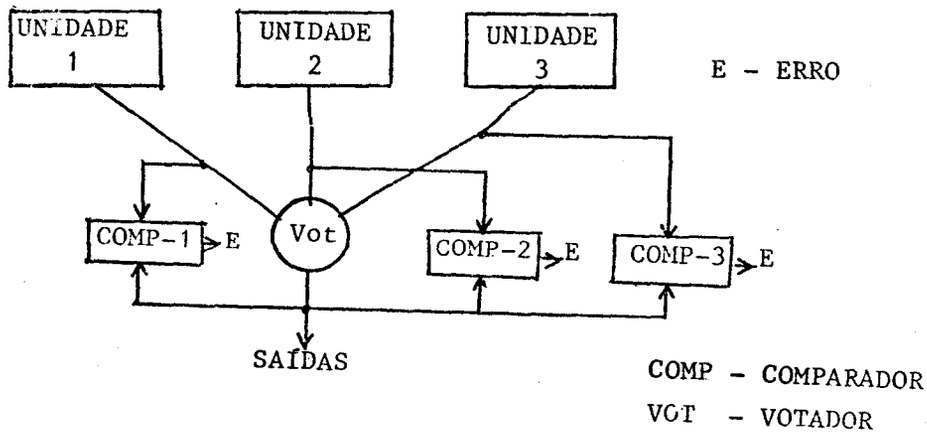


Fig. 7 - Circuitos de deteção de erros nos sistemas TMR.

Nos sistemas com três unidades fracamente conectadas, cada unidade de processamento tem seu próprio relógio não-sincronizado com os demais. As três unidades executam o mesmo programa ou programas equivalentes. Em pontos predeterminados do programa, as unidades trocam mensagens e realizam a votação, eliminando desta forma erros de uma unidade. A Figura 8 apresenta o esquema de interligação de sistemas com três unidades fracamente conectadas.

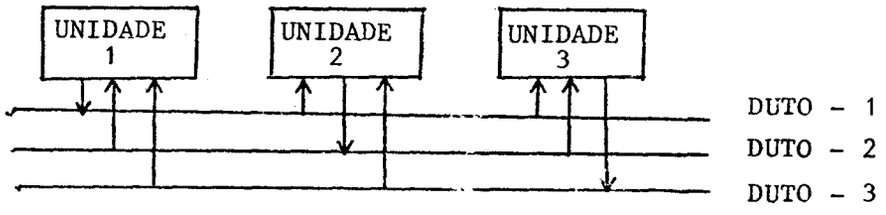


Fig. 8 - Arquitetura com três unidades fracamente conectadas.

A vantagem dos sistemas fracamente conectados sobre os fortemente conectados é a eliminação dos pontos de falhas simples representados pelo relógio e pelo votador. A desvantagem é a necessidade da implementação de programas para troca de mensagens entre as unidades e para votação, o que ocasiona maior tempo de execução.

Tanto no esquema fracamente conectado quanto no fortemente conectado, o sistema pode degradar para duas unidades quando uma das unidades falha. Entretanto, o sistema não suportará uma segunda falta sem que haja necessidade de suspender a operação e determinar a unidade falha. O sistema pode ainda degradar para uma unidade e operar de forma não-segura.

2.5 - SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos são constituídos por um conjunto de unidades de processamento interligadas entre si. Funcionalmente as unidades de processamento podem ser organizadas em níveis hierárquicos. Neste caso, uma ou mais unidades de processamento funcionam como unidades de supervisão, enquanto as demais executam funções específicas.

Os sistemas distribuídos apresentam algumas características interessantes para aplicações em sistemas tolerantes a falhas:

a) Tarefas críticas podem ser executadas simultaneamente em duas ou mais unidades de processamento, tendo os resultados comparados ou votados com vistas na deteção ou no mascaramento de erros. Por outro lado, as tarefas não críticas podem ser executadas em apenas uma unidade de processamento.

b) As unidades não-falhas podem cooperar no teste e na recuperação de uma unidade falha. Em caso de falha permanente, uma ou mais unidades podem assumir as tarefas que estavam sendo executadas pela unidade falha.

c) Os sistemas distribuídos são facilmente adaptados para aplicações que requerem maior processamento apenas pela adição de unidades extras de processamento, sem necessidade de reprojetar o sistema.

Os sistemas distribuídos se diferenciam principalmente na forma como as unidades de processamento são interligadas. Em geral, nos sistemas distribuídos tolerantes a falhas os sistemas de interligação são redundantes para evitar que uma falta no sistema de interligação interrompa a comunicação entre as unidades de processamento. Os tipos de interligação mais utilizados em sistemas distribuídos são:

- ponto-a-ponto;

- estrela;

- barramento (duto) de dados;

- anel.

2.5.1 - ESQUEMA DE INTERLIGAÇÃO PONTO-A-PONTO

Neste esquema de interligação, cada unidade de processamento é interligada com as demais por dois ou mais dutos de dados dedicados, como apresentado na Figura 9.

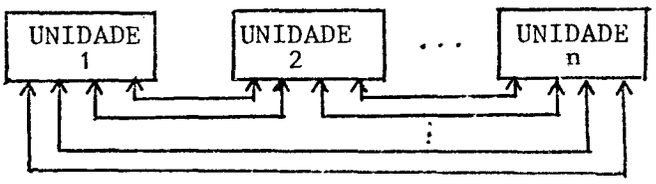


Fig. 9 - Esquema de interligação ponto-a-ponto.

Este esquema é utilizado quando o número de unidades de processamento é pequeno, pois para sistemas com diversas unidades o número de interligações torna-se extremamente grande. Adicionalmente, o circuito de interface com o sistema de interligação torna-se complexo. A vantagem deste esquema de interconexão é a não necessidade de compartilhar o mesmo duto de dados com as demais unidades. Outra vantagem é que este esquema permite o isolamento de uma unidade falha.

2.5.2 - ESQUEMA DE INTERLIGAÇÃO EM ESTRELA

Neste esquema de interligação, as unidades de processamento são interligadas a um conjunto de módulos centrais de comutação, tal como apresentado na Figura 10.

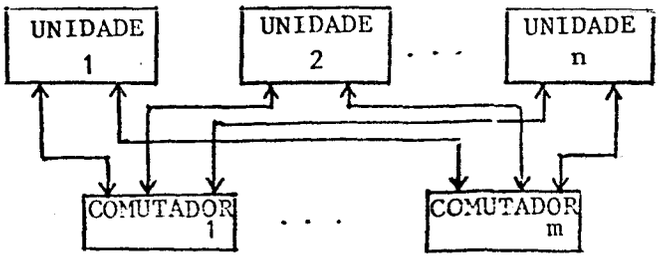


Fig. 10 - Esquema de interligação em estrelas múltiplas.

Geralmente a unidade de comutação realiza apenas uma interligação por vez, que pode ser do tipo ponto-a-ponto ou do tipo "broadcast". Dado que cada módulo de comutação é compartilhado por todas as unidades de processamento, faz-se necessário implementar neste módulo um algoritmo que selecione apenas uma unidade por vez.

Do ponto de vista da tolerância a falhas, este esquema apresenta a característica de que uma unidade de processamento ou um módulo de comutação, falhando, não interfere nos demais módulos. A vantagem deste esquema sobre o esquema ponto-a-ponto é que o número de interligações é bem menor. A desvantagem é a necessidade de utilização de um circuito adicional para realizar a comutação entre as linhas.

2.5.3 - ESQUEMAS DE INTERLIGAÇÃO ATRAVÉS DE BARRAMENTO DE DADOS

Neste esquema de interligação todas as unidades de processamento são ligadas a um conjunto de barramentos (dutos) de dados, de forma que os dados transmitidos por uma unidade são recebidos simultaneamente pelas demais unidades. Este esquema é apresentado na Figura 11.

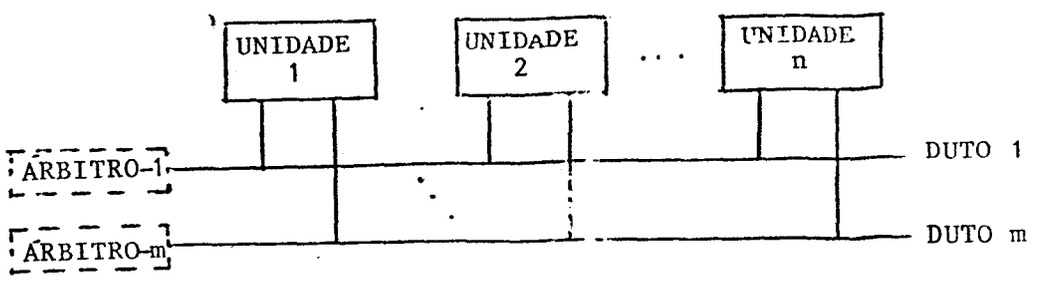


Fig. 11 - Esquema de interligação através de barramento de dados múltiplos.

Dado que cada barramento de dados é compartilhado por todas as unidades de processamento, é necessária a implementação de um mecanismo que garanta que apenas um processador transmita dados para o barramento por vez. Os dois mecanismos geralmente utilizados são o árbitro e o sensor de portadora. O árbitro é um esquema centralizado e consiste na implementação de um circuito que seleciona apenas uma unidade quando mais de uma unidade solicita o barramento. No esquema sensor de portadora, antes de transmitir um dado, a unidade escuta o barramento. Se este está desocupado, a unidade inicia a tranmissão. Se mais de uma unidade iniciar a transmissão ao mesmo tempo, haverá uma colisão e as unidades tentarão a retransmissão após um tempo aleatório.

O ponto crítico no esquema de interligação através de barramento de dados é que se uma unidade falhar, poderá afetar um ou mais barramentos, o que impedirá que as demais unidades os utilizem. Falha no árbitro afeta apenas o barramento por ele controlado.

2.5.4 - ESQUEMAS DE INTERLIGAÇÃO EM ANEL

Neste esquema de interligação, as unidades de processamento são interligadas através de anéis múltiplos, como apresentado na Figura 12.

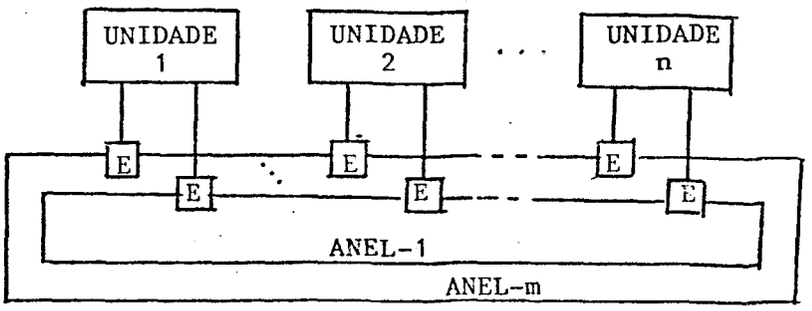


Fig. 12 - Esquema de interligação através de anéis múltiplos.

No esquema em anel, as mensagens enviadas por uma unidade circulam de estação em estação até retornar à estação original de onde são retiradas. Os três mecanismos geralmente utilizados para controlar o anel são: passagem de permissão ("token"), escaninhos ("slots") e a inserção de registradores.

No método de passagem de permissão, um padrão especial de bits (ficha) circula no anel quando este está livre. Quando uma unidade deseja transmitir uma mensagem, espera a chegada da ficha, retém a ficha, transmite a mensagem e, em seguida, libera a ficha. No método de escaninho, o anel é dividido em escaninhos que por ele circulam. Cada escaninho possui um bit que indica se este está ocupado ou não. Quando uma unidade deseja transmitir uma mensagem, espera um escaninho livre, coloca a mensagem no escaninho e comuta o bit para ocupado. Quando o escaninho retorna à origem, a mensagem é retirada e o bit é comutado para livre.

No método de inserção de registradores, a unidade inicia a transmissão logo que o anel está livre. Se durante a transmissão a estação recebe uma mensagem de uma unidade anterior a ela, esta mensagem é armazenada em um registrador. A mensagem armazenada é retransmitida logo após o término da transmissão da mensagem gerada na própria unidade.

Falha em uma estação do anel bloqueia a transmissão de dados no anel; entretanto, não afeta os outros anéis nem as unidades de processamento ligadas a este anel. As estações podem ser projetadas de forma a isolar uma unidade de processamento falho.

2.5.5 - SELEÇÃO DE UM ESQUEMA DE INTERLIGAÇÃO

Todos os esquemas de interligação para sistemas distribuídos, discutidos nas seções anteriores, apresentam características de tolerância a falhas e são plausíveis de ser utilizados em sistemas distribuídos tolerantes a falhas. A escolha vai depender da aplicação, da banda de passagem necessária e da distância entre as unidades de processamento.

3 - AVALIAÇÃO E SELEÇÃO DE SISTEMAS DE COMPUTAÇÃO TOLERANTES A FALHAS

Os três parâmetros geralmente utilizados para avaliar arquiteturas tolerantes a falhas são:

- R(t) - confiabilidade;
- MTBF - tempo médio entre falhas;
- A - disponibilidade.

A confiabilidade de um sistema no tempo t, R(t). é definida como a probabilidade de o sistema operar sem falhar até o tempo t dado que ele estava operando corretamente no início da operação (t=0). Esta definição para sistemas tolerantes a falhas não leva em conta a concorrência de faltas internas, desde que estas não influenciem o comportamento externo do sistema.

O MTBF, como o próprio nome indica é o tempo médio entre duas falhas do sistema. Esta definição é utilizada somente para sistemas reparáveis e inclui o tempo para deteção de erro e correção do defeito. Nos sistemas não reparáveis, utiliza-se o parâmetro MTBF, que indica o tempo médio até a ocorrência da primeira falha.

A disponibilidade de um sistema indica a percentagem de tempo que o sistema opera corretamente. Este parâmentro é utilizado

somente para sistemas reparáveis e depende do tempo médio de reparo e do MTBF.

3.1 - AVALIAÇÃO DE SISTEMAS TOLERANTES A FALHAS

Diversas configurações de sistemas tolerantes a falhas, reparáveis ou não, serão avaliadas nesta seção. A análise será realizada considerando que a taxa de falhas das unidades de processamento é constante e apresenta os seguintes valores:

λ -taxa de falha da unidade de processamento energizada e sem mecanismos de deteção de erros ao nível de circuito;

μ -taxa de falhas da unidade de processamento desenergizada que inclui os mecanismos de deteção de erro;

$\emptyset$ -taxa de reparo de uma unidade de processamento.

A confiabilidade de uma unidade de processamento com taxa de falha constante e sem reparo é dada pela Equação 1 e o MTTF, pela Equação 2.

$$R(t) = e^{-\lambda t} \qquad \text{Eq. 1}$$

$$MTTF = \frac{1}{\lambda} \qquad \text{Eq. 2}$$

Se a unidade de processamento é reparável, a disponibilidade é dada pela Equação 3.

$$A = \frac{\emptyset}{\emptyset + \lambda} \qquad \text{Eq. 3}$$

A Figura 13 apresenta o gráfico da confiabilidade em função do tempo normalizado ($T=\lambda t$) para uma unidade de processamento sem circuitos para deteção de erros e com circuito de deteção de erro para 20% e 50% de circuito adicional.

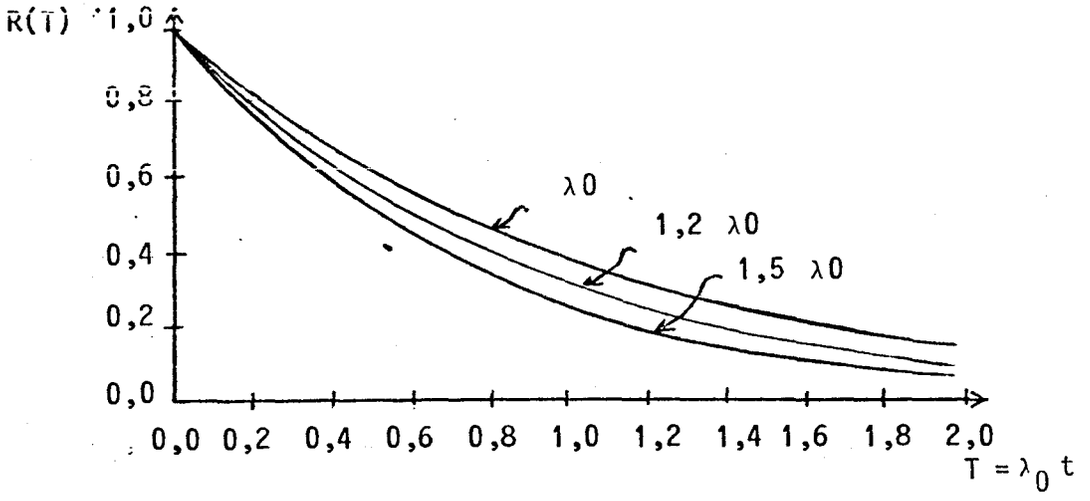


Fig. 13-Confiabilidade de uma unidade de processamento em função do circuito para deteção de erros.

A seguir será analisada a confiabilidade dos seguintes sistemas, considerando que eles não são reparáveis e que a taxa de falha de cada unidade de processamento é λ:

- sistemas com uma unidade de processamento - R1(T);

- sistemas com duas unidades de processamento sem degradação - R2 (T);

- sistemas com duas unidades de processamento com degradação para uma unidade - R2d(T);

- sistemas com três unidades de processamento com degradação para duas unidades (TMR) - R3(T).

As equações que representam a confiabilidade destes sistemas em função do tempo são apresentadas a seguir:

$$R1(t) = e^{-\lambda t} \qquad \text{Eq. 4}$$

$$R2(t) = [R1(t)]^2 = e^{-2\lambda t} \qquad \text{Eq. 5}$$

$$R2d(t) = [R1(t)]^2 + 2\,R1(t)\,[1 - R1(t)]$$

$$= 2\,e^{-\lambda t} - e^{-2\lambda t} \qquad \text{Eq. 6}$$

$$R3(t) = [R1(t)]^3 + 3\,[R1(t)]^2\,[1 - R1(t)]$$

$$= 3\,e^{-2\lambda t} - 2\,e^{-3\lambda t} \qquad \text{Eq. 7}$$

A Figura 14 apresenta o gráfico da confiabilidade destes sistemas em função do tempo normalizado ($T=\lambda t$).

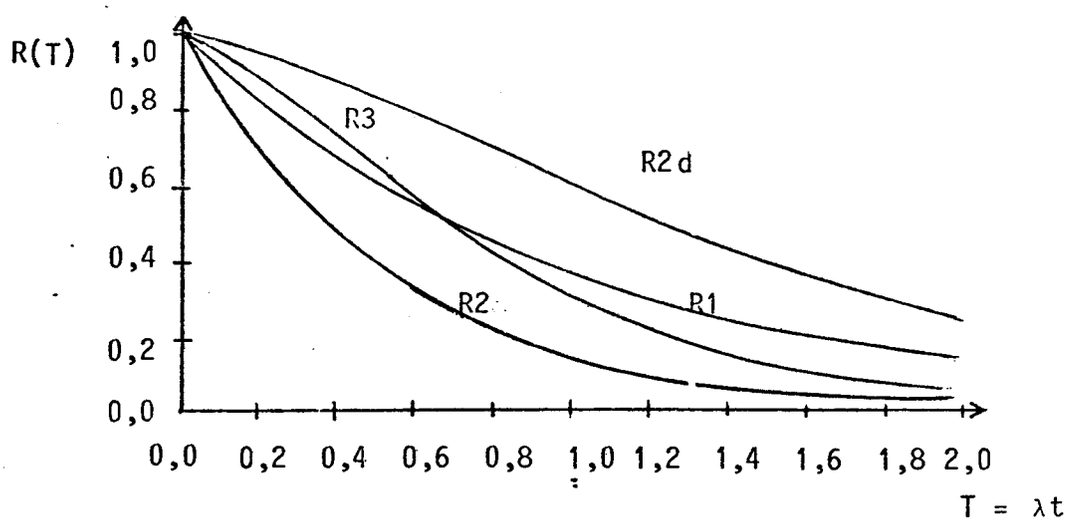


Fig.14-Confiabilidade em função do tempo em sistemas com uma, duas e três unidades de processamento.

Com base na Figura 14, podem-se tirar as seguintes conclusões:

- A confiabilidade de sistemas com duas unidades sem degradação é sempre menor que a de um sistema com apenas uma unidade; entretanto, a confiabilidade do sistema com duas unidades com degradação para uma unidade é sempre maior que a confiabilidade do sistema com apenas uma unidade.

- A confiabilidade de um sistema com três unidades com degradação para duas unidades é maior que a confiabilidade de um sistema com uma unidade para $t>0,7/\lambda$ e menor para $t>0,7/\lambda$.

Uma das formas utilizadas para aumentar a confiabilidade de sistemas sem reparo é a adição de unidades reservas de processamento que são

utilizadas para repor as unidades ativas quando estas falham. Para analisar estes tipos de sistemas considerar-se-á que:

- o sistema necessita de q unidades ativas e possui s unidades reservas que ficam desligadas;

- a taxa de falhas de uma unidade é constante e igual a λ quando a unidade está energizada e igual a μ quando está desenergizada;

- a probabilidade de o sistema ter sucesso no chaveamento para uma unidade reserva é igual a C.

A fórmula utilizada para calcular a confiabilidade destes sistemas é a seguinte:

$$R(t) = e^{-q\lambda t} \sum_{k=0}^{s} \binom{k-1 + q\frac{\lambda}{\mu}}{k} C^k \, (1 - e^{-\mu t})^k \qquad \text{Eq. 8}$$

onde $\binom{k-1 + q\frac{\lambda}{\mu}}{k}$ é a função binomial

Quando λ/μ é um número inteiro, a função binomial reduz-se a:

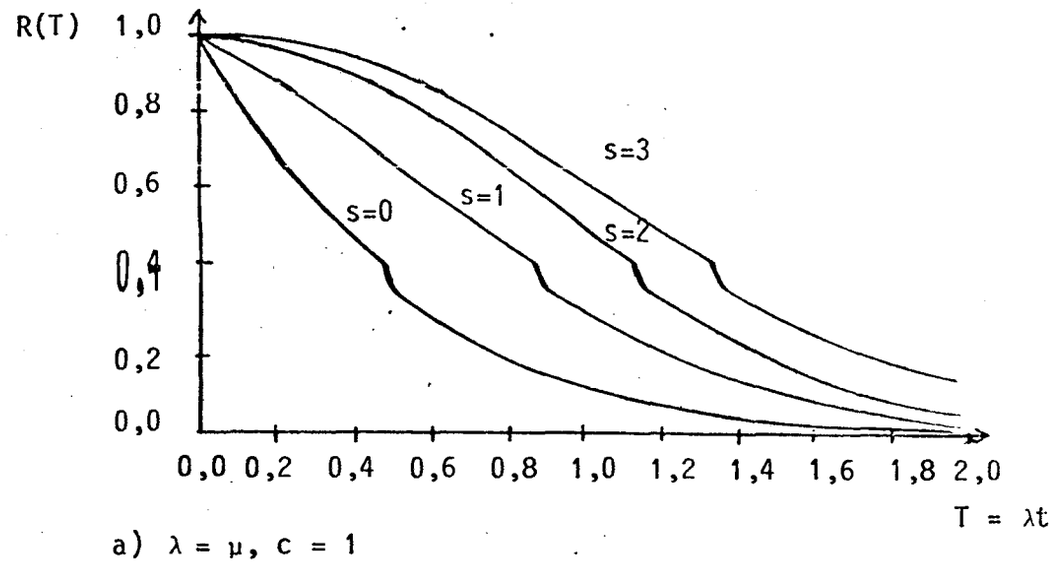
$$\frac{(k + n)!}{k!n!} \quad \text{onde } n = \frac{q\lambda}{\mu} - 1$$

A Figura 15 apresenta a confiabilidade de um sistema em função do tempo normalizado ($T=\lambda t$) com duas unidades ativas e até três unidades reserva para os seguintes casos:

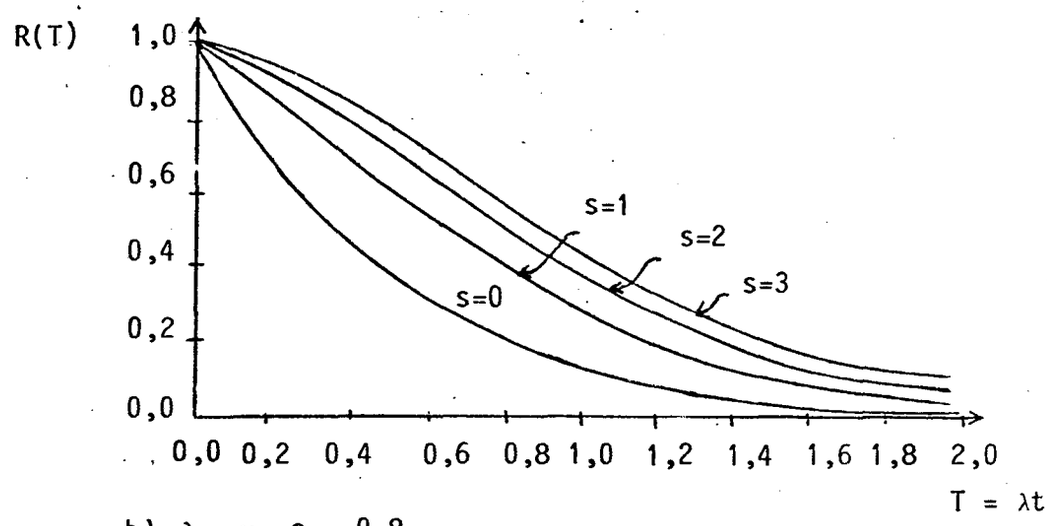
a) $\lambda = \mu$, $C = 1$;

b) $\lambda = \mu$, $C = 0{,}8$;

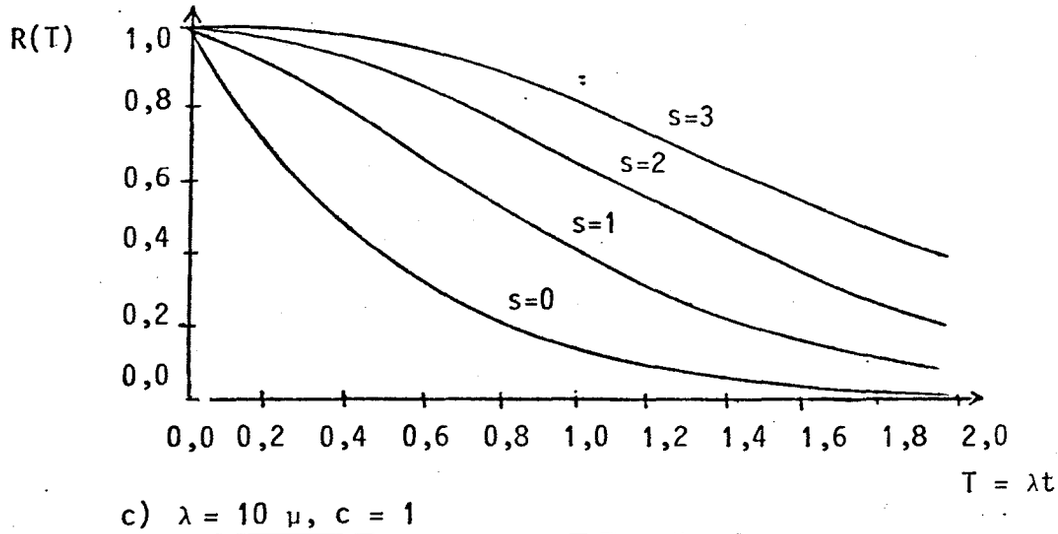
c) $\lambda = 10\,\mu$, $C = 1$;



a) $\lambda = \mu$, $c = 1$



b) $\lambda = \mu$, $c = 0,8$



c) $\lambda = 10\ \mu$, $c = 1$

Fig. 15 - Confiabilidade de um sistema com duas unidades ativas e três reservas.

Das curvas apresentadas na Figura 15, pode-se concluir que a utilização de unidades reservas aumenta a confiabilidade do sistema. O ganho é maior quando a taxa de falhas da unidade reserva é menor do que a da unidade ativa. Entretanto, o ganho diminui se a probabilidade de chaveamento com sucesso para uma unidade reserva é menor que um.

Para sistemas com reparo, as fórmulas analíticas que representam a sua confiabilidade tornam-se mais complexas. Os sistemas com taxa de falha e reparo constante podem ser modelados através de sistemas markovianos.

Como exemplo de sistema com reparo, é analisado a seguir um sistema com duas unidades de processamento, com possibilidade de degradar para uma unidade e com uma estação de reparo.

O modelo de Markov para o cálculo da confiabilidade é apresentado na Figura 16a e para o cálculo da disponibilidade, na Figura 16b.

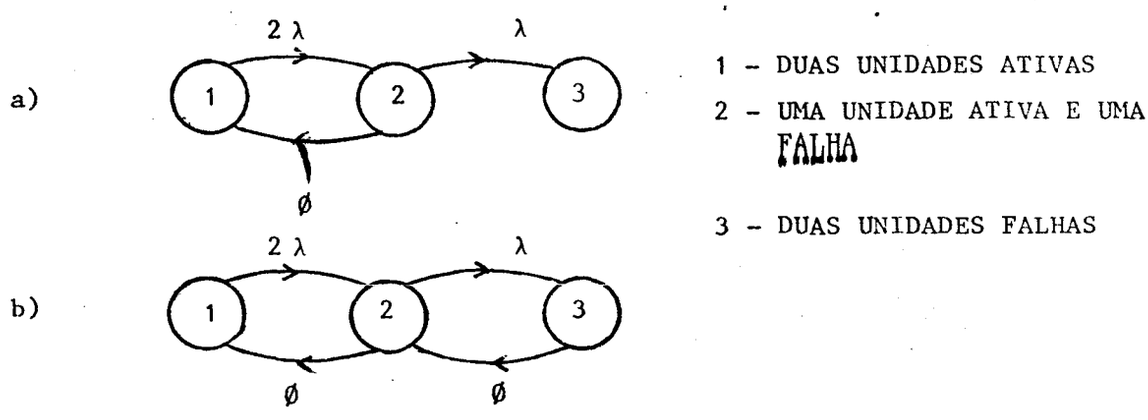


Fig. 16- Modelo de Markov: a) cálculo da confiabilidade;
b) cálculo da disponibilidade.

A confiabilidade do sistema é dada pela probabilidade de o sistema permanecer nos estados 1 e 2 do diagrama de Markov apresentado na Figura 16a e é dada pela seguinte equação:

$$R(t) = \frac{4\lambda^2 \, e^{-\frac{1}{2}(3\lambda + \phi - \delta)t}}{(3\lambda + \phi)\delta - \delta^2} = \frac{4\lambda^2 \, e^{-\frac{1}{2}(3\lambda + \phi + \delta)}}{(3\lambda + \phi)\delta + \delta^2} \qquad \text{Eq. 9}$$

onde $\delta = \sqrt{\lambda^2 + 6\lambda + \phi^2}$

A Figura 17 apresenta a confiabilidade do sistema em função do tempo normalizado (T=λt) com duas unidades de processamento sem reparo e com reparo, para a taxa de reparo de 10, 100 e 1000 vezes maior que a taxa de falhas de uma unidade de processamento.

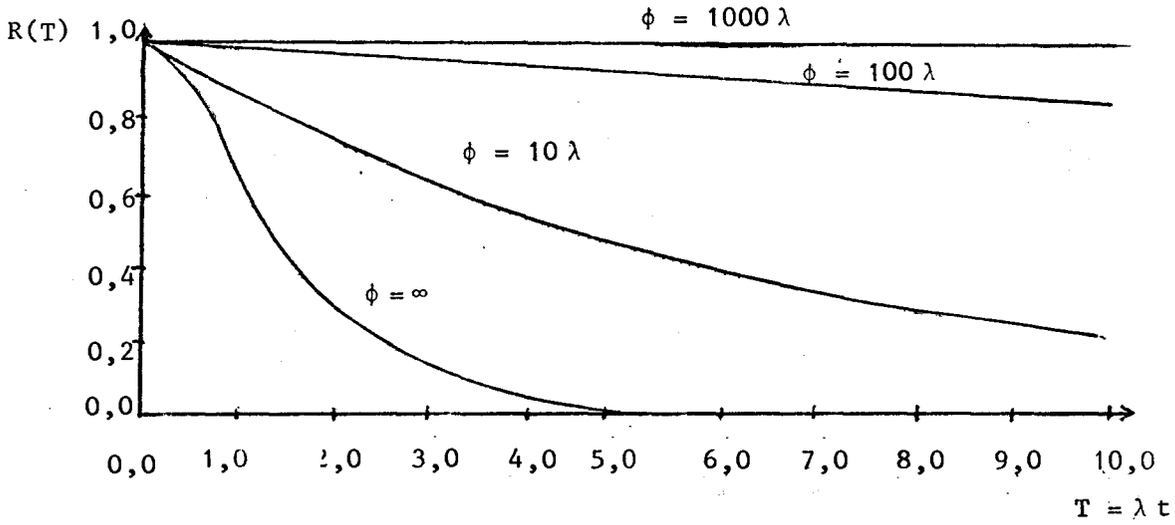


Fig. 17 - Confiabilidade do sistema com duas unidades em função da taxa de reparo.

A disponibilidade do sistema indica a percentagem de tempo em que o sistema permanece nos estados 1 e 2 do diagrama de Markov, apresentado na Figura 16b, e é expressa pela seguinte fórmula:

$$A = \frac{k^2 + 2}{k^2 + 2k + 1} \qquad \text{Eq. 10}$$

onde $k = \frac{\phi}{\lambda}$

Por exemplo, para R igual a 100, A vale 0,980; para R iqual 1000, A vale 0,998.

Pode-se concluir através da Figura 17 e da Equação 10 que quanto menor for o tempo médio de reparo, maior serão a confiabilidade e a disponibilidade do sistema.

3.2 - SELEÇÃO DA ARQUITETURA

A seleção da arquitetura de um sistema de computação para uma dada aplicação dependerá grandemente da criticalidade das tarefas que o sistema deverá executar. Para facilitar a seleção, as tarefas a serem executadas pelo sistema de computação serão classificadas em:

- não- críticas: quando eventuais falhas e atrasos na execução da tarefa são toleráveis;

- semicríticas: quando atrasos na execução da tarefa são toleráveis mas falhas não;

- críticas: quando nem falhas nem atrasos na execução da tarefa são toleráveis.

A análise a seguir será restrita a arquiteturas que tolerem faltas em apenas uma unidade de processamento. Esta análise poderá ser estendida a sistemas que tolerem faltas em mais de uma unidade de processamento. Para estes sistemas, faz-se necessária a utilização de unidades reservas que são utilizadas para substituir uma unidade ativa quando esta falha.

Como na execução de tarefas não-críticas eventuais falhas e atrasos são toleráveis, estas tarefas podem ser executadas em sistemas com apenas uma unidade de processamento, desde que programas de testes ou diagnose sejam executados periodicamente. Mecanismos de deteção de erros devem ser implementados ao nível de programas e de circuitos para minimizar a probabilidade de ocorrência de falhas no intervalo de tempo entre a execução dos programas de testes. O supervisor deve ser acionado quando um erro for detetado, e a unidade de falha deve, então, ser reparada ou substituída por uma unidade reserva.

Como na execução de tarefas semicríticas a ocorrência de falhas não é tolerável, estas tarefas devem ser executadas por uma unidade de processamento desde que esta seja totalmente autotestável, ou por sistemas com duas unidades de processamento desde que estas executem a mesma tarefa e tenham os resultados comparados antes de uma operação externa. Para esta classe de tarefas, atraso na execução é permitido. Portanto o sistema, ao detetar um erro, pode executar programas de diagnose para identificar a unidade falha, a qual poderá ser reparada ou substituída por uma unidade reserva.

Nas tarefas críticas, atrasos e falhas não são toleráveis. Portanto, estas tarefas devem ser executadas em duas unidades totalmente autotestáveis, ou em sistemas com três unidades de processamento. No caso de duas unidades totalmente autotestáveis, a execução da operação poderá ser prosseguida sem atraso em apenas uma das unidades, enquanto a outra é reparada ou substituída. No caso de três unidades, elas devem executar o mesmo programa, ou programas equivalentes, e ter seus resultados votados e comparados. Caso se detete que uma unidade está falha, a execução da tarefa continuará sem interrupção por duas unidades, que terão seus resultados comparados entre si, enquanto a unidade falha é reparada ou substituída. Neste caso, o reparo ou a substituição da unidade falha deverá ser executado no menor tempo possível para diminuir a probabilidade de uma segunda unidade falhar enquanto a unidade falha estiver sendo reparada ou substituída.

Em sistemas onde o reparo não é possível, como por exemplo em satélites, o chaveamento para unidades reservas deverá ser implementado de forma automática. O número de unidades reservas dependerá da duração da missão e do nível de confiabilidade especificado para o sistema de computação.

4 - EXEMPLOS DE SISTEMAS DE COMUTAÇÃO TOLERANTES A FALHAS

Diferentes sistemas de computação tolerantes a falhas já foram implementados até a presente data. Alguns foram desenvolvidos em laboratório para estudo e validação das técnicas de tolerância a falhas e não chegaram a ter aplicações práticas. Diversos outros foram ou estão sendo empregados com sucesso em diferentes aplicações, tais como satélites, foguetes, aeronaves, centrais telefônicas e controle industrial.

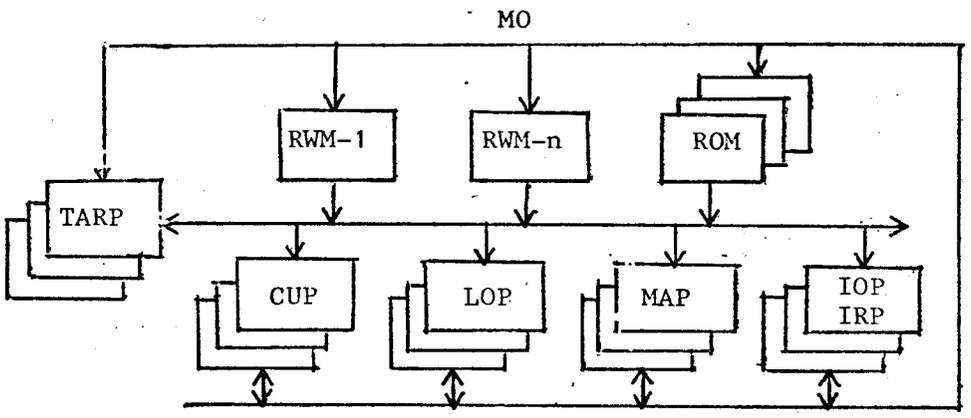
Os seguintes sistemas de computação tolerantes a falhas serão analisados:

- STAR;
- FTMP;
- UDS;
- ESS;
- TANDEM - 16.

4.1 - STAR

O Self-Testing and Repairing Computer (STAR) foi desenvolvido pelo Jet Propulsion Laboratory (JPL) de 1961 a 1972 para missões espaciais de até 10 anos. As missões espaciais que utilizariam a STAR foram canceladas: entretanto, seu projeto influenciou bastante os sistemas de computação tolerantes a falhas que foram desenvolvidos posteriormente.

A arquitetura do STAR é baseada na utilização de módulos internos redundantes, como apresentado na Figura 18.



COP - PROCESSADOR DE INSTRUÇÃO
LOP - PROCESSADOR LÓGICO
MAP - PROCESSADOR ARITMÉTICO
IOP - PROCESSADOR DE ENTRADA E SAÍDA
IRP - PROCESSADOR DE INTERRUPÇÃO
RWM - MEMÓRIA DE ESCRITA E LEITURA
RUM - MEMÓRIA DE LEITURA
TARP - PROCESSADOR DE TESTE E REPARO
MI, MO - DUTOS DE DADO

Fig. 18 - Arquitetura do STAR.

Os mecanismos de tolerância a falhas são implementados no STAR da seguinte forma:

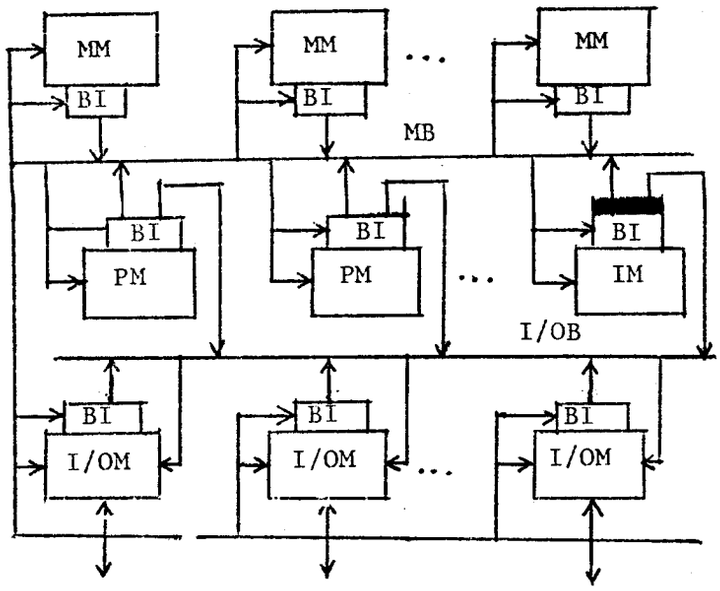
- Todas as palavras de instrução ou dado são codificadas em um código de deteção e erro, e o TARP verifica a validade das palavras enviadas aos dutos MI, MO. Após a execução de uma função, o processador que a executou envia uma palavra de status para o TARP indicando se a função foi executada com sucesso.

- O TARP, ao detetar um defeito em um módulo, desliga este módulo e ativa um módulo reserva.

- O TARP é triplicado (TMR) para mascarar os erros internos. Módulos reservas são utilizados para substituir um módulo defeituoso do TARP.

4.2 - FTMP

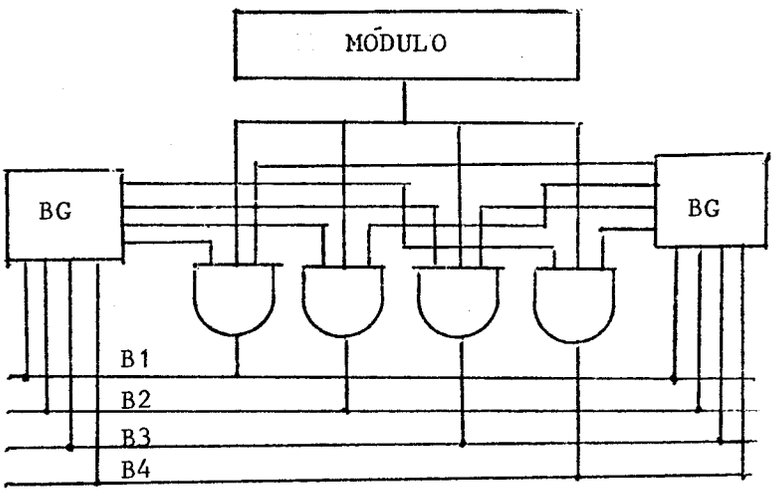
O FAULT-Tolerant Multiprocessor (FTMP) foi desenvolvido de 1970 a 1979 para aplicações em aeronaves. O FTMP é constituído de um conjunto de módulos de processamento, módulos de memória e módulos de entrada e saída. O diagrama de bloco deste sistema é apresentado na Figura 19.



MM - MÓDULO DE MEMÓRIA
PM - MÓDULO DE PROCESSAMENTO
I/OM - MÓDULO DE ENTRADA E SAÍDA
MB - DUTO DE MEMÓRIA
I/OB - DUTO DO I/OM

Fig. 19- Arquitetura do FTMP.

Os diversos módulos operam sincronizadamente e são agrupados de três em três. Um conjunto de três módulos de memória, de três módulos de processamento e de três módulos de entrada e saída formam uma unidade de processamento. Cada módulo de processamento da unidade executa a mesma tarefa e realiza, através de circuito, a votação majoritária de seus resultados. A Figura 20 descreve o circuito de interface com os dutos de dado (BI).



BG - "BUS GUARDIAN"

B1, B2, B3, B4 - DUTOS DE DADOS

Fig. 20 - Interface com os dutos de dados.

A interface do módulo com os dutos de dados é controlada por dois Circuitos de Reconfiguração (BG) que recebem comandos de reconfiguração através de três dutos de dados. Os dados recebidos são votados, e os BGs geram os comandos para ligar ou desligar o módulo e para selecionar o duto que será conectado à saída do módulo. Esta conexão só é realizada quando os dois BGs decodificam o mesmo comando. A reconfiguração é controlada por um conjunto de três módulos de processamento.

4.3 - UDS

O Unified Data System UDS foi desenvolvido pelo JPL para aplicações em satélites. Ele é constituído por um conjunto de unidades de processamento interconectadas por dutos de dados redundantes. As unidades de processamento são organizadas em dois níveis hierárquicos. As unidades de alto nível são utilizadas em computação em geral e na supervisão das demais unidades. As unidades de baixo nível são acopladas aos subsistemas do satélite para aquisição de dados e controle destes subsistemas.

O diagrama de blocos do UDS é apresentado na Figura 21.

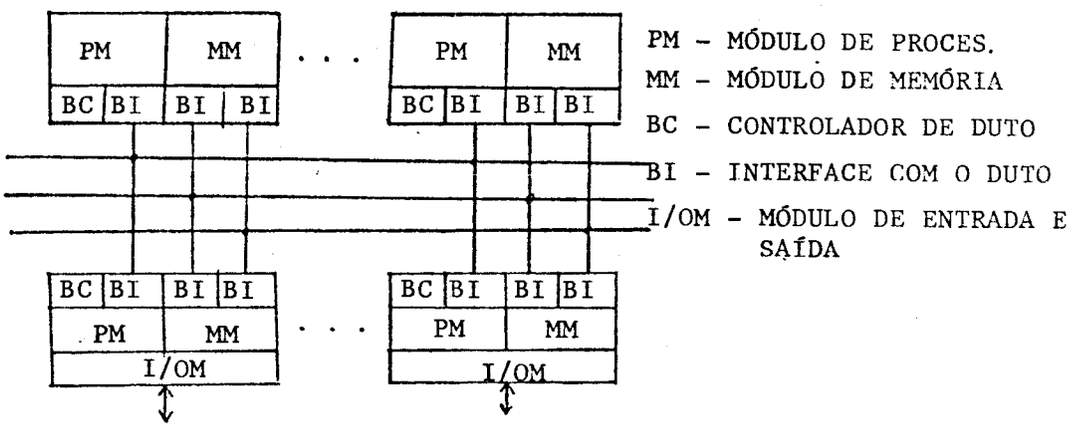


Fig. 21 - Arquitetura do UDS.

As unidades de processamento são totalmente autotestáveis e implementadas em VLSI. O código de Hamming é utilizado na memória para deteção de erros duplos e correção de erros simples. As tarefas são geralmente executadas em duas unidades de processamento que comparam seus resultados. Tarefas críticas podem ser programadas para ser executadas simultaneamente em três unidades de processamento.

4.4 - NO 3A ESS

O No 3A Electronic Switching System (ESS) foi desenvolvido pela Bell System para controlar até 5.000 linhas telefônicas. Este sistema tem como requisito não ficar fora de serviço, a não ser no máximo durante alguns minutos por ano, e processar erroneamente menos de 0,01% das ligações. O No 3A ESS é formado por dois módulos de processamento, dois módulos de memória e duas unidades de fita do tipo cartucho. O diagrama de bloco do No 3A ESS é apresentado na Figura 22.

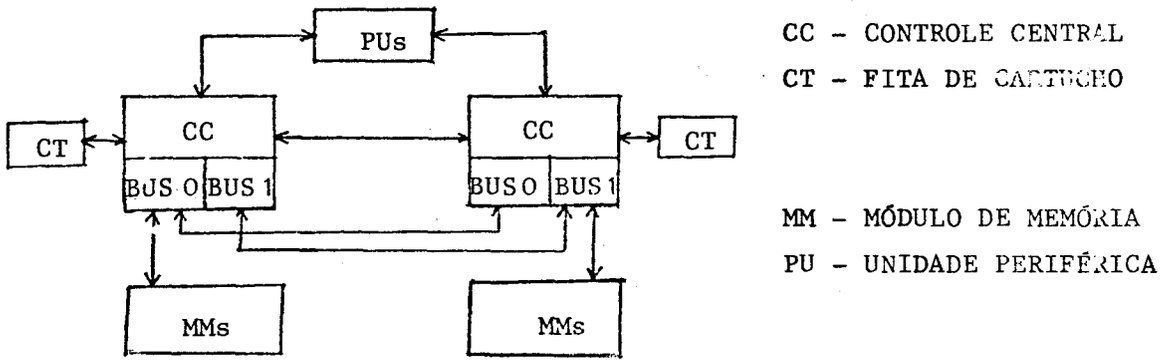


Fig. 22 - Diagrama de blocos do No 3A ESS.

Durante a operação normal, apenas um dos módulos de processamento opera enquanto o outro fica desativado. O módulo ativo escreve nos dois módulos de memória de forma a mantê-los sempre atualizados.

O módulo de processamento é microprogramável e quase totalmente autoverificável. As palavras da memória de microprogramação são codificadas em um código de deteção de erro. As palavras de dados são codificadas com um código de paridade de dois bits. O circuito que realiza as operações lógicas e aritméticas é duplicado e compara os resultados. Quando um erro é detetado,a unidade de reserva é ativada, os programas são carregados da unidade de fita de cartucho, e a operação é reestabelecida do ponto onde foi interrompida.

4.5 - TANDEM 16

O Tandem 16 é um computador de uso geral para aplicações que requerem alta disponibilidade. O Tandem é constituído por um conjunto de unidades de processamento interligadas por dois dutos de dados denominados Dynabus. A arquitetura do Tandem configurado com quatro unidades de processamento é apresentada na Figura 23.

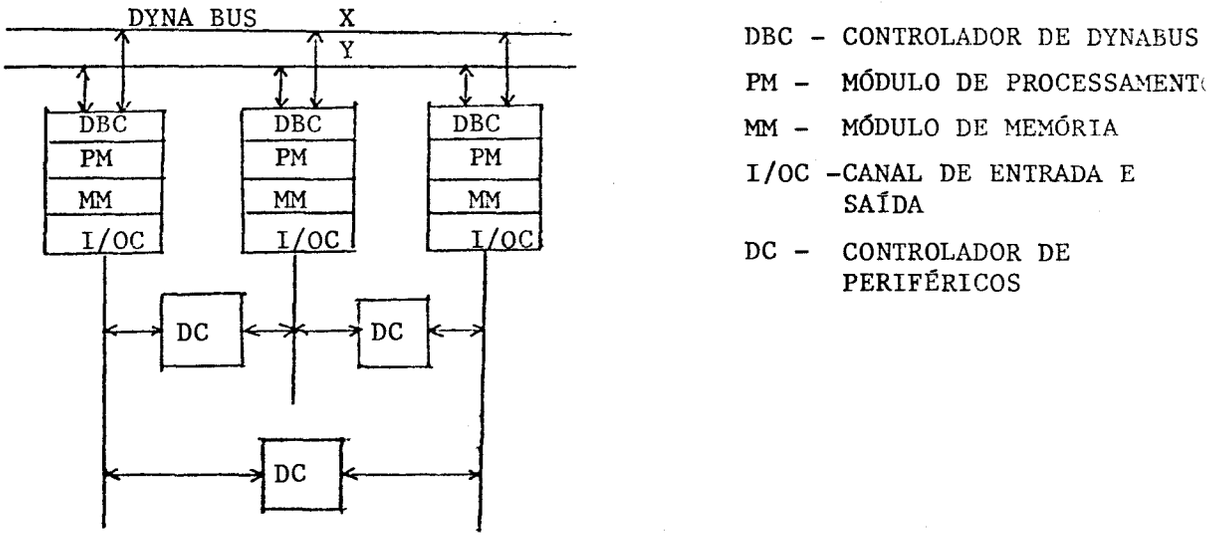


Fig.23 - Diagrama de blocos do Tandem 16.

Cada unidade de processamento é constituída de um módulo de processamento, um módulo de memória, um controlador do Dynabus e um controlador de entrada e saída interligado aos controladores de periféricos. Cada controlador de periférico é ligado a duas unidades de processamento.

A falha de uma unidade de processamento não afeta as demais unidades. Da mesma forma, a falha de um controlador de entrada e saída não bloqueia os controladores do periférico que está ligado a ele.

As unidades de processamento comunicam-se através de trocas de mensagens. Cada unidade de processamento envia para as demais unidades, a cada segundo, uma mensagem especial que indica seu status interno. A cada dois segundos as unidades de processamento verificam se receberam mensagens das outras unidades. Se a mensagem de uma dada unidade de processamento não é recebida neste intervalo de tempo, a unidade é considerada como falha, e o programa operacional reconfigura o sistema.

5.0 - CONCLUSÕES

A seleção da arquitetura de um sistema de computação tolerante a falhas para uma dada aplicação depende dos requisitos de confiabilidade, disponibilidade e segurança especificados, bem como do esquema de manutenção definido. Em alguns sistemas, como por exemplo satélites, reparo geralmente não é possível, ou é extremamente caro. Para que estes sistemas tolerem uma ou mais faltas, faz-se necessária a utilização de módulos ou unidades de processamento reserva. Para estes casos, a reconfiguração deve ser automática ou telecomandada. Por outro lado, para sistemas reparáveis não há necessidade da utilização de módulos ou unidades reservas quando o sistema pode ser interrompido. Para estes casos é necessária apenas a implementação de mecanismos de deteção de erros, os quais podem ser implementados por programação, por circuito, ou por ambos.

Nas aplicações que requerem alta disponibilidade é necessária a utilização de unidades redundantes de forma a substituir a unidade ativa quando esta falha. Para minimizar a probabilidade de a unidade ativa falhar quando a outra unidade estiver em reparo, é necessário reduzir ao mínimo possível o tempo de reparo.

O requisito principal para um sistema que requer alta segurança é que ele não gere comandos externos ou forneça dados errôneos. Para sistemas nos quais é permitida a interrupção da operação, é aconselhável a utilização de uma unidade totalmente autotestável ou de duas unidades que executem o mesmo programa e tenham seus resultados comparados. Estas duas unidades podem operar sincronizadamente ou não. No primeiro caso a comparação é realizada por circuito enquanto no segundo, por programação. Nos sistemas em que não é possível interromper a operação, faz-se necessária a utilização de duas unidades totalmente autotestáveis, ou de três unidades que tenham os mesmos programas ou programas equivalentes e seus resultados votados. Para este caso, as três unidades podem operar sincronizadamente ou não e os resultados podem ser votados por circuito ou por programação.

Com exceção de sistemas que utilizam o mascaramento dos erros através de redundância estática, a forma usual de redundância ao nível de circuito é a redundância dinâmica, a qual é geralmente implementada em quatro fases. A primeira fase consiste na deteção do erro. Esta fase pode ser implementada por programação e/ou por circuito. Quanto mais rápida for a deteção do erro gerado por um defeitom, mais fácil será a próxima fase.

A segunda fase consiste no isolamento do módulo falho e na avaliação do dano provocado na estrutura de dados. Portanto, ao projetar um sistema tolerante a falhas é necessário definir as áreas de isolamento para determinadas classes de erro. As áreas de isolamento podem ser definidas tanto ao nível de módulo quanto ao nível de unidade.

A terceira fase consiste na substituição do módulo ou unidade falha e na restauração da estrutura de dados. Em alguns sistemas, ao se esgotarem as unidades reservas, poderá haver degradação do sistema para um estado de menor poder de computação, mas ainda com a capacidade de executar as tarefas críticas. Em outros sistemas, faz-se necessário levar o sistema para um estado seguro e interromper a operação. A quarta fase consiste em retornar à operação normal.

Atualmente, as técnicas de tolerância a falhas já estão bastante consolidadas. Anualmente é realizado um simpósio internacional na área de tolerância a falhas, onde são apresentados os seus principais desenvolvimentos . A bibliografia na área de tolerância a falhas é bastante ampla. Diversos livros sobre este assunto já foram escritos. Podem-se ressaltar os livros escritos por Siewiorek e Swarz (1982) e por Anderson e Lee (1981). O primeiro apresenta as diversas técnicas de tolerância a falhas com bastante detalhes. O segundo fornece uma visão mais geral e descritiva do assunto. Ambos apresentam uma lista ampla de relatórios técnicos na área. O Proceeding do IEEE (1978) é uma ediçao especial sobre sistemas tolerantes a falhas e descreve em detalhes a arquitetura de diversos sistemas.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

- Anderson, T. e Lee, P.A, "Fault Tolerante Principles and Practice", Prentice Hall International, 1981

- Siewiorek, D.P e Swarz, R.S, "The Theory and Practice of Rialiable System Design" Digital Press, 1982.

- IEEE, "Proceeding of the IEEE. Special Issue on Fault-Tolerant Digital System", Outubro de 1978.

CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO
INSTITUTO DE PESQUISAS ESPACIAIS

PROPOSTA PARA PUBLICAÇÃO

DATA: 08.09.87

IDENTIFICAÇÃO

TÍTULO: ARQUITETURA DE SISTEMAS DE COMPUTAÇÃO TOLERANTES A FALHAS

AUTORIA: Alderico Rodrigues de Paula Junior

PROJETO/PROGRAMA: RECOM

DIVISÃO:

DEPARTAMENTO: DCA

DIVULGAÇÃO ☐ EXTERNA ☒ INTERNA MEIO: RTR

REVISÃO TÉCNICA

REVISOR TÉCNICO: KLAUS JUERGEN JOHANSEN

RECEBI EM: _____ REVISADO EM: _____

OBSERVAÇÕES: ☐ NÃO HÁ ☐ VER VERSO

DEVOLVI EM: _____ Klaus J. Johansen
ASSINATURA

APROVAÇÕES

APROVADO: ☐ SIM ☐ NÃO ☐ VER VERSO

_____ DATA _____ CHEFE DIVISÃO

APROVADO: ☑ SIM ☐ NÃO ☐ VER VERSO

23.11.87 DATA Eduardo CHEFE DEPARTAMENTO

REVISÃO DE LINGUAGEM

Nº: 229 PRIORIDADE: 2

DATA: 11-9-87

REVISADO ☐ COM ☐ SEM CORREÇÕES ☐ VER VERSO

POR: Paulina Prado de Carvalho

16.9.87 DATA Paulina P. Carvalho ASSINATURA

DATILOGRAFIA

O(S) AUTOR(ES) DEVE(M) MENCIONAR NO VERSO, OU ANEXAR NORMAS E/OU INSTRUÇÕES ESPECIAIS

RECEBIDO EM: _____

CONCLUÍDO EM: _____

DATILÓGRAFA: Cidinha/R.388

08.09.87 ASSINATURA

PARECER

FAVORÁVEL: ☐ SIM ☐ NÃO ☐ VER VERSO

_____ DATA _____ RESPONSÁVEL/PROGRAMA

EM CONDIÇÕES DE PUBLICAÇÃO EM: _____ Alderico
AUTOR RESPONSÁVEL

AUTORIZO A PUBLICAÇÃO: ☐ SIM ☐ NÃO

DIVULGAÇÃO ☐ INTERNA ☐ EXTERNA MEIO: _____

OBSERVAÇÕES: _____

_____ DATA _____ DIRETOR

SEC

PUBLICAÇÃO: _____ PÁGINAS: _____ ÚLTIMA PÁGINA: _____

CÓPIAS: _____ TIPO: _____ PREÇO: _____