UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

"DESENVOLVIMENTO DE UM SISTEMA MICROCOMPUTADOR
TOLERANTE A FALHAS COM ARQUITETURA EM ANEL"

DEISY PIEDADE MUNHOZ FISCHER

ORIENTADOR:

PROF. DR. VALENTIN OSAC RODA

USP / IFQSC / SPM
[REDACTED]
0-2-66667

PC

Tese de Doutorado apresentada ao Instituto de Física e
Química de São Carlos, USP, para a obtenção do Título de
Doutor em Ciências "Física Aplicada"

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

DEPARTAMENTO DE FÍSICA E CIÊNCIA DOS MATERIAIS

SÃO CARLOS - 1990

UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE DEISY PIEDADE M. FISCHER APRESENTADA AO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 26.10.90

COMISSÃO JULGADORA:

Prof. Dr. Valentin Obac Roda

Prof. Dr. Alberto Tannus

Prof. Dr. Carlos Antonio Ruggiero

Prof. Dr. Caetano Traina Junior

Prof. Dr. Boaventura Freire dos Reis

*Dedico este trabalho, a todas as
pessoas que me são muito queridas, e
em especial ao Isaac, meu filho.*

*Agradeço ao Senhor Deus, Criador do
Universo, por ter me dado condições,
de desenvolver este trabalho e por
ter atendido as minhas orações nos
momentos mais difíceis.*

Donde, pois, vem a sabedoria e onde está o lugar da inteligência?

Deus entende o seu caminho, e ele sabe o seu lugar.

*Porque ele vê as extremidades da terra, e vê tudo, o que há
debaixo dos céus.*

Quando deu peso ao vento, e tomou a medida das águas.

*Quando prescreveu uma lei para a chuva e caminho para o
relâmpago dos trovões.*

*Então a viu e a manifestou; estabeleceu-a, e também a
esquadrinhou.*

*Mas disse ao homem: Eis que o temor do Senhor é a sabedoria,
apartar-se do mal é a inteligência.*

Livro de Jó (cap. 28; v. 20 e 23 a 28)

AGRADECIMENTOS

Ao Prof. Dr. Valentin Obac Roda, pela orientação, amizade, incentivo, e empenho para que este trabalho fosse realizado com sucesso.

Ao Prof. Dr. Caetano Traina, pelo auxílio no desenvolvimento deste trabalho e pela gentileza em ceder os programas fontes, referentes ao seu trabalho de doutorado.

Ao Benedito René Fischer, meu esposo, pelo companheirismo e pela colaboração, que prestou no desenvolvimento deste trabalho.

Aos técnicos de Laboratório de Instrumentação Eletrônica do IFQSC, em especial à Ivanilda, pelo auxílio na implementação do sistema.

Ao Gilberto, desenhista do Departamento de Física de Rio Claro, pela paciente elaboração dos desenhos.

A Liliane, técnica do Laboratório de Computação do DEMAC, pelo trabalho de edição do texto.

A todos meus amigos, que incentivaram, colaboraram e desejaram a concretização desta Tese.

A UNESP, em especial ao Departamento de Física de Rio Claro, que permitiu e deu condições para o desenvolvimento desta pesquisa.

ABSTRACT

A fault-tolerant, tri-module redundant (TMR), microcomputer System is presented. This system is characterized by a Ring Architecture implemented with three processor modules. The ring structure is a loop type architecture in which adjacent modules are connected by single communication links. The modules receive data from one or more sources (depending on whether these sources are replicated or not). This information is then processed and made ready for voting. The data is passed between the adjacent modules over the connecting links. Fault-tolerance is achieved by each of the three processors being able to examine computational results from its neighbour. Thus, each module process receives two versions of each calculation: one from its own calculation and one received from the other processor. Each module then performs the voting by software, with voting on partial data strategy. If no fault has occurred, it can be expected that all the three modules will produce the same result. The result of the voting (comparison) is indicated in each module by an error condition flag. In the event of a fault in one of the module/processors, then this will be recognized by the software voting and an error will be reported, the system will continue proper operation in spite of the failure of a single module. Triple Modular Redundant Ring System can tolerate a single fault in one of the modules. The calculations are

not carried out in a tightly synchronized manner, but the processors are loosely synchronized by software.

The system was implemented using three Z-80 based microcomputer boards. Each microcomputer board has its own disk-controller board. The system access a single video terminal. The software monitor is comprised of three identical modules, one for each three microcomputer. Each software monitor module resides in the respective local memory of its microcomputers. The application software performs under CP/M Operational System. Programs from non-redundant versions will be executed in a fault tolerant manner without modification. Through this, our objective was to develop a system of general application, with high availability.

RESUMO

Neste trabalho é apresentado um sistema microcomputador tolerante a falhas, com redundância modular tripla (TMR). Este sistema é caracterizado por uma Arquitetura em Anel implementada com três módulos processadores. A estrutura em Anel é uma arquitetura onde os módulos adjacentes são conectados por um canal de comunicação, formando um laço. Os módulos recebem dados de uma ou mais fontes (dependendo se as fontes são replicadas ou não). Esta informação é então processada e um dado é preparado para votação. O dado é transmitido aos módulos adjacentes, através do canal de comunicação. A tolerância à falhas é obtida, pela capacidade que os três processadores têm de examinar os resultados do processamento de seus vizinhos. Assim, cada processador recebe duas versões de cada processamento: o seu próprio resultado e o resultado do seu vizinho. Cada módulo, então executa a votação por programação, através da estratégia de votação sobre um número parcial de dados. Se nenhuma falha ocorreu, os três módulos irão produzir o mesmo resultado. O resultado da votação (comparação) é indicado em cada módulo por um sinalizador de erro. Quando ocorre uma falha em um módulo, o esquema de votação por programação identifica a ocorrência desta falha, mas o sistema irá continuar a operar corretamente, apesar da falha em um módulo. O sistema em Anel com Redundância Tripla, pode tolerar uma falha em um dos módulos. Estes

cálculos não são executados de uma maneira fortemente sincronizada, mas os processadores são sincronizados de uma forma mais flexível, através de programação. O sistema foi implementado usando três módulos microcomputadores. Cada microcomputador tem um controlador de disco. O sistema acessa um único terminal de vídeo. O programa monitor é composto de três módulos idênticos, para os três microcomputadores. Cada módulo reside na memória local de cada microcomputador. O sistema executa o Sistema Operacional CP/M. Os programas para este sistema operacional serão executados de uma forma tolerante à falhas sem necessidade de modificações. O objetivo deste trabalho foi desenvolver um sistema de uso geral com alta disponibilidade.

ÍNDICE

CAPÍTULO 1 : INTRODUÇÃO	1
1.1 COMPUTADORES TOLERANTES À FALHAS	1
1.2 O OBJETIVO DESTE TRABALHO	6
1.3 APRESENTAÇÃO	9
CAPÍTULO 2 : A TÉCNICA DE TOLERÂNCIA À FALHAS	11
2.1 INTRODUÇÃO	11
2.2 TOLERÂNCIA À FALHAS	13
2.2.1. CONFIABILIDADE E DISPONIBILIDADE	13
2.3 TÉCNICAS DE TOLERÂNCIA À FALHAS	18
2.3.1. CARACTERIZAÇÃO DE ERROS	18
2.3.2. DETECÇÃO DE ERROS	21
2.3.3. TÉCNICAS DE REDUNDÂNCIA	25
2.4 REDUNDÂNCIA NA INFORMAÇÃO	26
2.5 REDUNDÂNCIA TEMPORAL	27
2.6 REDUNDÂNCIA POR CIRCUITOS	28
2.6.1. REDUNDÂNCIA ESTATICA POR CIRCUITOS	29
2.6.2. REDUNDÂNCIA DINÂMICA POR CIRCUITOS	31
2.7 REDUNDÂNCIA POR PROGRAMAÇÃO	37
2.7.1. REPLICAÇÃO DE RECURSOS DE PROGRAMAÇÃO	38
2.7.2. VOTAÇÃO POR PROGRAMAÇÃO	39
2.8 CONCLUSÃO	42
CAPÍTULO 3 : IMPLEMENTAÇÃO DA ARQUITETURA EM ANEL TOLERANTE À FALHAS	44
3.1 INTRODUÇÃO	44
3.2 A ARQUITETURA EM ANEL	45
3.2.1. CONSIDERAÇÕES SOBRE A VOTAÇÃO POR PROGRAMA ÇÃO NA ARQUITETURA EM ANEL	47
3.3 IMPLEMENTAÇÃO DA ARQUITETURA EM ANEL	62
3.3.1. CARACTERÍSTICAS DO SISTEMA DESENVOLVIDO ..	62

3.3.2.	CONFIGURAÇÃO DO SISTEMA	64
3.4	A ARQUITETURA DE UM MÓDULO	65
3.4.1.	O SUBSISTEMA DE PROCESSAMENTO	66
3.4.2.	O SUBSISTEMA DE ENTRADA E SAÍDA	72
3.4.3.	O SUBSISTEMA DE COMUNICAÇÃO NO ANEL	74
3.5	PROGRAMAS DE GERENCIAMENTO DO SISTEMA	86
3.5.1.	SISTEMA DE ARMAZENAGEM EM DISCO FLEXÍVEL ..	86
3.5.2.	DESCRIÇÃO DOS PROGRAMAS DE GERENCIAMENTO ..	93
CAPÍTULO 4 : SINCRONIZAÇÃO EM SISTEMAS TOLERANTES À FALHAS COM VOTAÇÃO		103
4.1	INTRODUÇÃO	103
4.2	SINCRONIZAÇÃO	104
4.3	SINCRONIZAÇÃO COM RELÓGIOS	106
4.3.1.	RELÓGIOS INDEPENDENTES	108
4.3.2.	RELÓGIO EXTERNO	108
4.3.3.	REALIMENTAÇÃO MÚTUA	114
4.4	PONTO COMUM DE ENCONTRO	127
4.4.1.	A SINCRONIZAÇÃO NA ARQUITETURA EM ANEL ..	132
4.5	ANÁLISE	136
CAPÍTULO 5 : ANÁLISE DO SISTEMA		140
5.1	ANÁLISE DA TOLERÂNCIA À FALHAS	140
5.2	ANÁLISE DA ARQUITETURA	141
5.2.1.	ANÁLISE DA COMUNICAÇÃO	141
5.2.2.	ANÁLISE DA VOTAÇÃO PARCIAL	149
5.2.3.	ANÁLISE DA DISPONIBILIDADE	159
CAPÍTULO 6 : CONCLUSÃO E PERPECTIVAS		161
BIBLIOGRAFIA :		166
APÊNDICE: LISTAGEM DAS ROTINAS DE GERENCIAMENTO....		176

CAPÍTULO 1

INTRODUÇÃO

1.1 COMPUTADORES TOLERANTES A FALHAS

Lei de Murphy : "Mais dia, menos dia, um sistema irá falhar, e geralmente na pior hora possível".

A pesquisa em computadores tolerantes a falhas, tem por objetivo tornar a afirmação de Murphy menos fatalista. Espera-se que os computadores tolerantes a falhas, continuem a execução correta de seus programas, mesmo na ocorrência de falhas no sistema, sejam estas falhas nos circuitos, ou nos programas [AVIZIENIS, 1975]. Um computador tolerante a falhas, pode sobreviver a um conjunto de falhas, que fariam com que um computador convencional, parasse suas operações.

Atualmente os computadores estão sendo usados nas mais diversas atividades humanas. Estas atividades envolvem aplicações militares, médicas, econômicas, transportes coletivos, controle de processos, comunicações, satélites e naves espaciais. O grau de dependência humana, em relação aos computadores, vem se tornando cada dia maior. É, portanto, necessário que os sistemas de computação sejam construídos, de forma que se possa confiar nos serviços fornecidos pelos mesmos. Com

isto, no desenvolvimento dos sistemas de computação, é interessante introduzir o conceito de **segurança de funcionamento** ("dependability"). "A segurança de funcionamento pode ser definida como a qualidade do serviço fornecido, por um sistema de computação, tal que confiança possa ser justificadamente depositada neste serviço" [LAPRIE, 1982, 1985] [PAULA JR., 1989]. Desta forma, a **segurança de funcionamento** está relacionada com a **qualidade do serviço fornecido**, de modo que se possa ter confiança neste serviço. O serviço fornecido por um sistema, é o comportamento desse sistema, conforme percebido por outro sistema (ou usuários), interagindo com o sistema em questão [FRAGA, 1987]. As duas formas utilizadas para quantificar a qualidade do serviço fornecido, são [PAULA JR., 1989] [LAPRIE, 1985]:

Confiabilidade- é a medida do serviço continuamente realizado a partir de uma referência de tempo inicial. É a probabilidade do sistema operar sem falhas até o tempo t , dado que ele estava operando corretamente no início da operação ($t=0$) [MINI CURSO, 1987].

Disponibilidade-é a medida do serviço realizado em relação a alternância de realizações e interrupções.

Um meio de se obter segurança de funcionamento em um sistema de computação, é através da construção de sistema de computação tolerante a falhas. Outro meio, é através do uso da técnica de intolerância a falhas, ou prevenção de falhas. Neste último, a segurança de funcionamento é garantida por construção, com a eliminação à "priori" das possíveis causas de falhas. Busca-se já no projeto, e na implementação do sistema, eliminar as fontes de falhas, através de repetidos testes, análises, seleção de circuitos e programas.

Nos sistemas de computação tolerantes a falhas, não se evita por construção, que as falhas ocorram. Contudo, são providenciados meios de proteção, para que a ocorrência de falhas, não venha alterar o funcionamento correto do sistema, como um todo. O principal meio de proteção empregado, é o uso de redundância de recursos. A redundância de recursos, significa o uso de recursos suplementares, que seriam desnecessários, em um sistema livre de falhas, para a sua operação normal. Estes recursos podem ser: materiais (circuitos, módulos, processadores), lógicos (programas, informação) e tempo (repetição de tarefas).

A pesquisa em sistemas de computação tolerantes a falhas, é voltada para as diferentes formas de implementação da redundância de recursos, e gerenciamento destes recursos. Na implementação, a escolha da arquitetura tolerante a falhas dependerá da aplicação destinada ao sistema. Há quatro categorias de aplicações [RENNELS, 1980] [MAXION, 1987] : situações que exigem alta disponibilidade, execução de tarefas críticas, adiamento

da manutenção, e longevidade do sistema.

Atualmente com a informatização, principalmente das atividades comerciais e bancárias, a utilização de sistemas de computação com alta disponibilidade vem se tornando um requisito fundamental. A disponibilidade é definida também, como a probabilidade que o sistema irá fornecer o serviço requisitado, quando solicitado [LADUZINSKY, 1985] [SIEWIOREK, 1982]. Sistemas com alta disponibilidade usam a redundância de recursos de forma a garantir a operação do sistema, enquanto está sendo requisitado o seu serviço. Na ocorrência da falha, é necessário que haja facilidade para reparo e recuperação ou reconfiguração do sistema, de maneira a mantê-lo disponível. Como exemplos de sistemas tolerantes a falhas com alta disponibilidade, tem-se os sistemas comerciais TANDEM [KATZMAN, 1977] [BARTLET, 1977] e o STRATUS [HEDRIE, 1983].

A execução de tarefas críticas, exige mais cuidado no tratamento de falhas. Na execução destas tarefas, atrasos e falhas não são tolerados. Para a execução das tarefas, as arquiteturas podem ser implementadas usando duas unidades totalmente auto-testáveis, ou mais unidades de processamento. Com duas unidades, se ocorrer uma falha, a operação pode seguir sem atraso com a outra unidade. Enquanto isto, a unidade com problemas pode ser reparada. Com mais unidades, por exemplo três, elas devem executar o mesmo programa, e os resultados devem ser votados ou comparados. No caso de falhas em uma unidade, as outras duas devem garantir a operação do sistema, enquanto a unidade defeituosa é

reparada ou substituída. As unidades com falhas devem ser rapidamente reparadas, para que uma segunda falha não interrompa o serviço. O controle de processos em tempo real, por exemplo o controle de aeronaves e o controle de processos industriais, são aplicações críticas, para as quais se exige que os sistemas de computação tenham alta confiabilidade, e possam ser rapidamente reparados. O sistema August série 300, é um exemplo de sistema comercial para aplicações críticas [WENSLEY, 1981].

As aplicações onde é necessário o adiamento da manutenção, exige que a arquitetura escolhida, incorpore meios para tolerar falhas até que o sistema possa ser recuperado. Neste caso, o uso de unidades reservas pode ser uma solução. Outra solução é o uso de várias unidades de processamento, como no caso das aplicações críticas. Nestas soluções, o sistema vai se reconfigurando, até que o reparo possa ser executado. As estações remotas de processamento/coleta de dados, o controle de transporte coletivos tipo Metrô, são alguns exemplos destas aplicações.

Sondas e satélites espaciais são exemplos de aplicações, onde se exige longevidade do sistema de computação. Nas missões não tripuladas, geralmente o reparo é impossível após o início de missão e o sistema deve tolerar falhas, por períodos tão longos como 10 anos ou mais. Estes sistemas são altamente redundantes, equipados com várias unidades reservas. O número de unidades reservas vai depender da duração da missão. O gerenciamento do sistema, por exemplo a ativação das unidades reservas, poderá ser executado automaticamente

(pelo próprio sistema) ou remotamente através de estações da Terra. A sonda VOYAGER é um exemplo desta aplicação [JONES, 1982].

1.2 O OBJETIVO DESTE TRABALHO

Do que foi apresentado no item anterior, verifica-se que a pesquisa em arquiteturas tolerantes a falhas é um campo fértil, com importantes aplicações nas várias atividades humanas. Vários sistemas foram, e estão sendo desenvolvidos. Alguns já são consagrados comercialmente como o TANDEM, usado principalmente em transações bancárias e o AUGUST-300, voltado para o controle de processos.

O objetivo deste trabalho, foi estudar e implementar um sistema de computação tolerante a falhas utilizando uma arquitetura diferente e operando com o Sistema Operacional CP/M. A arquitetura pesquisada é a Arquitetura em Anel. Esta arquitetura pode utilizar três ou mais módulos processadores, sendo estes módulos interligados na forma de um anel. O sistema implementado utiliza três módulos processadores, tendo uma estrutura modular tripla (TMR - "Triple Modular Redundancy"). A Tolerância a Falhas é implementada com o uso de programas idênticos, processados em três módulos idênticos e o resultado deste processamento submetido a votação. O objetivo foi desenvolver um sistema de uso geral com alta disponibilidade e confiabilidade. O sistema é transparente

ao usuário, no sentido que o usuário do sistema, não precisa de conhecimentos especiais para utilizá-lo.

Os sistemas TMR são geralmente utilizados nas aplicações que requerem alta confiabilidade de operações. Os sistemas convencionais consistem de três módulos processadores (atualmente microcomputadores) [WAKERLY, 1976], projetados para operarem sincronamente, e a seleção da saída do sistema, é obtida através do princípio de votação por maioria [VON NEUMAN, 1956]. Estas arquiteturas, toleram um módulo com defeito. A falha é mascarada e não afeta os serviços oferecidos pelo sistema. Recursos podem ser adicionados, para permitir também a detecção da falha e indicação do processador com defeito. A votação pode ser feita através de circuitos, ou através de programação, e os elementos votadores podem ser triplicados ou não.

O sistema implementado nesta pesquisa, utiliza três módulos de processamento, que são microcomputadores baseados no microprocessador Z-80 e ambiente operacional CP/M. Cada módulo tem o seu próprio controlador de disco flexível, e todos tem acesso a um único terminal de vídeo. A votação é efetuada, todas as vezes que o sistema apronta um dado para enviar ao terminal de vídeo. Quando um módulo termina o processamento de um dado, ele o envia ao módulo adjacente. Este compara o dado recebido, com o resultante de seu próprio processamento, e sinaliza o resultado da comparação. Os sinalizadores são usados para selecionar um módulo, funcionando corretamente, para alimentar a saída do sistema. Tanto o processo de votação, como o processo de seleção de saída, são implementados através de

algoritmos. O algoritmo de votação, difere dos sistemas TMR convencionais: aqui é utilizado um esquema de votação parcial, e cada módulo executa o processo de votação. Também os 3 módulos fazem a seleção de saída. Tudo se processa como se as unidades de votação e de seleção fossem triplicadas. Todos os módulos executam estas operações conjuntamente. Foi criado um sistema de "linha bastão", para permitir que apenas um módulo, tenha acesso a um periférico de entrada ou de saída. O mecanismo utilizado para a transmissão de dados é o de comunicação com reconhecimento ("handshake"). Isto facilitou a operação de sincronização dos módulos, evitou-se uma base de tempo comum, e tornou flexível este procedimento. Os processadores não precisam executar suas tarefas em intervalos rigidamente iguais, e a sincronização é obtida através de um ponto comum de encontro ("rendez-vous"). Assim, todo o gerenciamento da tolerância a falhas é executado através de programação. Isto torna esta arquitetura promissora para diversas aplicações, onde seja necessário segurança de funcionamento do sistema de computação. Graças à flexibilidade do processo de votação e de sincronização dos módulos, esta arquitetura tanto pode ser usada no tratamento de bases de dados, como no controle de processos.

A Arquitetura em Anel foi proposta pela primeira vez em [OBAC RODA, 1983.a] e implementada em um sistema extremamente simples, usando o microcomputador 8748 da Intel, e executava a leitura de chaves digitais. O ciclo de processamento era controlado por um computador mestre, e a seleção de saída feita através de circuitos lógicos. O

sistema implementado aqui, faz o seu próprio gerenciamento, sem a utilização de computador mestre e executa tarefas mais complexas como o processamento do CP/M.

Em toda a literatura pesquisada, nenhum sistema usa uma arquitetura semelhante à implementada neste trabalho. Também o procedimento de sincronização é praticamente inexplorado em arquitetura com redundância modular. Um exemplo encontrado, usa um sistema TMR que utiliza uma memória não tolerante a falhas para efetuar a comunicação no processo de sincronização [CHANDE, 1989].

1.3 APRESENTAÇÃO

No capítulo 2 é apresentada uma revisão das definições e procedimentos relacionados com a Técnica de Tolerância a Falhas. É feito um resumo das formas de caracterização e detecção de erros e dos vários tipos de redundância utilizados nos Sistemas Tolerantes a Falhas.

A implementação do sistema do ponto de vista de circuitos, bem como do gerenciamento da Tolerância a Falhas, é apresentada no capítulo 3. Neste capítulo é feita uma análise do grau de tolerância a falhas proporcionado pelo uso do esquema de votação parcial na Arquitetura em Anel. Para efeito de comparação é analisado o esquema de votação majoritária. Isto é feito, através das expressões de confiabilidade deduzidas para os dois

esquemas.

A sincronização dos processadores é o ponto vital para o funcionamento de qualquer arquitetura com redundância modular e votação. No capítulo 4, é feita uma revisão das diversas formas de sincronização, utilizadas nestas arquiteturas. Também é analisada a sincronização implementada na Arquitetura em Anel, comparando-a com outras implementações.

No capítulo 5, é apresentada uma análise global do sistema implementado, abordando tanto os aspectos vantajosos desta arquitetura, o desempenho, e os pontos críticos.

As conclusões, as propostas para a continuidade do trabalho e sugestões para a aplicação desta arquitetura estão no capítulo 6.

CAPÍTULO 2

A TÉCNICA DE TOLERÂNCIA A FALHAS

2.1 - INTRODUÇÃO

O objetivo deste capítulo é apresentar um resumo da teoria de Tolerância a Falhas nos aspectos mais relevantes para a pesquisa desenvolvida.

A utilização cada vez maior do computador e dos microcomputadores em todos os ramos da atividade humana, faz que se busque de forma crescente, sistemas de computação que apresentem operação correta, mesmo na presença de falhas. Sistemas com esta característica são chamados de Sistemas Tolerantes a Falhas. Com estes sistemas de computação, procura-se manter a execução correta de programas e operações de entrada e saída, livres de erro, e o tempo de execução das tarefas dentro de limites especificados.

Para evitar os transtornos que ocorrem quando um sistema falha, existem basicamente duas técnicas [SIEWIOREK, 1982] [MINI CURSO, 1987] :

- (1) - técnica de intolerância a falhas
(prevenção);
- (2) - técnica de tolerância a falhas
(recuperação e redundância).

Com a primeira técnica, são construídos equipamentos com componentes extremamente confiáveis e são feitos testes exaustivos de circuitos e programas. Tenta-se eliminar por construção, as possíveis causas de falhas. Como é impossível eliminar todas as causas de falhas, o sistema irá funcionar adequadamente, até que a primeira falha ocorra. É impossível conseguir 100% de confiabilidade nos circuitos. Embora seja possível prever a confiabilidade de um lote de componentes, não é possível assegurar a confiabilidade de um componente individual. Na impossibilidade de garantir 100% de não ocorrência de falhas, os projetos que utilizam esta técnica, procuram reduzir a taxa de ocorrência de falhas a níveis aceitáveis [FRAGA, 1987]. Esta técnica tem custos elevados de implementação.

Na técnica de tolerância a falhas [SIEWIOREK, 1982] [AVIZIENIS, 1975] [MAXION, 1987], parte-se do princípio que falhas, erros ou defeitos, não podem ser evitados. Os sistemas são projetados e construídos usando-se meios que lhes permitam continuar operando corretamente, mesmo na presença de falhas. Esta técnica tem a vantagem de permitir a utilização de componentes com custo menor do que os usados na primeira técnica (intolerância a falhas) [MAXION, 1987]. A seguir estes conceitos são apresentados com mais detalhes.

2.2 - TOLERÂNCIA A FALHAS

"A função da tolerância a falhas, é reagir à ação da falha na máquina física, permitindo à máquina lógica continuar funcionando corretamente" [LEITE, 1987]. Um computador tolerante a falhas deverá continuar a execução correta de um programa, mesmo na presença de falhas. A tolerância a falhas tem se constituído em uma importante técnica, para se obter altos índices de confiabilidade e (ou) de disponibilidade [SIEWIDREK, 1982] em sistemas digitais.

2.2.1 - CONFIABILIDADE E DISPONIBILIDADE

O estudo da confiabilidade e da disponibilidade surgiu da necessidade de garantir o funcionamento correto de um sistema, equipamento ou componente, por um tempo de vida determinado. Este estudo é importante no desenvolvimento de instrumentação automática, nas áreas onde o fator segurança é requisito indispensável.

A Confiabilidade de um componente ou um sistema, é função do tempo, sendo definida como a probabilidade que o componente ou sistema não falhe no instante t , dado que ele estivesse operacional no instante $t = 0$. [SIEWIDREK, 1982]. Ela é determinada, pela capacidade do sistema manter-se em operação correta, apesar de falhas

que possam ocorrer nos circuitos e (ou) na programação [MAXION, 1987].

A expressão para a confiabilidade (R) é dada em função da taxa de falhas pela expressão [SIEWIOREK, 1982] [MINI CURSO, 1987] :

$$R(t) = \exp \left(- \int_0^t \lambda(t) d(t) \right) \quad (2.1)$$

Onde $\lambda(t)$ é a taxa de falhas no intervalo de tempo Δt . A função $\lambda(t)$ fornece uma avaliação quantitativa, do desempenho de componentes ou sistemas. No caso de componentes, a taxa de falhas é determinada pelo fabricante, submetendo os componentes a ensaios de laboratório e verificando quantos deles sobreviveram (mantiveram a operação correta), no intervalo de testes. A taxa de falhas é normalmente expressa em FITS (uma falha por 10^9 horas) [SIEWIOREK, 1982]. A taxa de falhas em função do tempo de vida de um componente ou sistema, segue a forma apresentada na figura 2.1, onde se tem 3 regiões distintas:

- período de falhas pré-maturas (iniciais), com taxa de falhas decrescente, "mortalidade infantil";
- período em que a taxa de falhas é constante;
- período de desgaste, quando a taxa de falha é crescente.

- I - Região de altas taxas de falhas : os elementos apresentam defeitos logo que postos em funcionamento - "mortalidade infantil".
- II - Período de operação normal : taxa de falhas aproximadamente constante .
- III - Região de desgaste : degradação dos elementos ,há um crescimento da taxa de falhas .

Fig. 2.1: VARIAÇÃO DA TAXA DE FALHAS

Para componentes eletrônicos e durante o período de vida útil, a taxa de falhas pode ser considerada constante [MINI CURSO, 1987]. E a confiabilidade dos componentes individuais, no período II (fig. 2.1), tem a distribuição de Poisson:

$$R(t) = e^{-\lambda t} \quad (2.2)$$

Circuitos com baixa escala de integração (SSI: por exemplo, tipo TTL), têm taxas de falhas de 10^3 a 10^4 FITS, enquanto circuitos com larga escala de integração (LSI: por exemplo, memórias de 1024 bit) têm taxa de falhas da ordem de 10^4 a 10^3 FITS.

Outros coeficientes relacionados com a confiabilidade do sistema são: Tempo Médio Entre Falhas (MTBF - "Mean Time Between Failure") ; Tempo Médio para Ocorrência de Falhas (MTTF - "Mean Time To Failure") e Tempo Médio para Conserto (MTTR - "Mean Time To Repair"). Estes parâmetros são parâmetros estatísticos, calculados sobre um conjunto de sistemas ou dispositivos idênticos, e são usados para comparação e avaliação de diferentes sistemas.

O MTBF traduz o tempo médio de bom funcionamento de um sistema reparável [FORBES, 1985] entre sucessivas falhas. O MTTF indica, nos sistemas não reparáveis, o tempo médio até a ocorrência da primeira falha [SIEWIOREK, 1982] [MINI CURSO, 1987]. E o MTTR é usado para medir o grau de recuperação (de conserto do sistema). O tempo

médio entre falhas é igual ao tempo médio para a ocorrência de falhas mais o tempo de reparo:

$$MTBF = MTTF + MTTR \quad (2.2)$$

A Disponibilidade é a probabilidade que o sistema irá fornecer o serviço requisitado, quando solicitado [SIEWIOREK, 1982] [MAXION, 1987]. Este conceito é diferente da confiabilidade e está relacionado com o tempo de reparo. A disponibilidade é uma medida de quão frequentemente um sistema falha e quão rapidamente ele é restaurado. Um sistema que falha frequentemente, mas pode ser rapidamente restaurado, terá baixa confiabilidade e alta disponibilidade.

A disponibilidade (A) pode ser definida como [SIEWIOREK, 1982] [MINI CURSO, 1987] [OBAC RODA, 1988.a] :

$$A = \frac{\text{tempo de operação}}{\text{tempo de operação} + \text{tempo inoperante}} \quad (2.3)$$

A disponibilidade é um fator importante, em sistemas que possam ser interrompidos para manutenção e reparo, como os sistemas bancários e estações telefônicas. Já a confiabilidade é um parâmetro importante em sistemas onde o reparo nem sempre é possível, como nos satélites espaciais ou onde há risco à vida humana.

2.3 - TÉCNICAS DE TOLERÂNCIA A FALHAS

A implementação da técnica de tolerância a falhas, envolve duas etapas principais [SIEWIOREK, 1982] :

1. detecção de erros ou falhas nos sistemas;
2. recuperação do sistema com substituição ou isolamento do defeito (reconfiguração), para que o conjunto retorne a sua operação normal.

A detecção de erro é o ponto de partida para o uso da técnica de tolerância a falhas. Tanto a detecção de erros, como a recuperação do sistema, envolve o uso da técnica de redundância. A técnica de redundância consiste na adição ao sistema de informações e recursos extras, além do necessário, para a sua operação normal [SIEWIOREK, 1982] [MAXION, 1987]. O uso de redundância, é a base da tolerância a falhas.

2.3.1 - CARACTERIZAÇÃO DE ERROS

A detecção de erros é a habilidade que o sistema tem de reconhecer, que está em um estado errôneo [JOHNSON, 1984]. Uma falha é a causa direta de um erro [LEITE, 1987] [FRAGA, 1987]. Há falhas de projeto, falhas de

implementação, falhas em componentes, falhas que ocorrem devido a distúrbios externos (como ruídos eletromagnéticos, temperatura, sobretensão, humidade), e falhas na interação do homem com a máquina.

Tanto o erro, como a falha possuem algumas características [JOHNSON, 1987] [MAXION, 1987] :

1. causa : física, humana ou do ambiente
2. natureza: circuitos ou programas, digital ou analógico.
3. duração : permanente, transiente ou intermitente
4. extensão: localizada em uma parte do circuito, em uma rotina do programa, ou espalhada pelo sistema.
5. valor : determinado (fixo em um valor), ou indeterminado.
6. comportamento: omissão, queda, falha maliciosa [CRISTIAN, 1989]
[DRUMMOND, 1987]

Destas características, a duração é um problema que precisa ser considerado com cuidado no projeto de Sistemas Tolerantes a Falhas. Como ilustração: ao se detectar um erro, há sistemas que podem isolar a parte com defeito e continuar operando reconfigurados, ou substituí-la por outra sem defeito. No caso de um erro (ou falha) transiente, a substituição ou desligamento de um módulo, poderia implicar em desperdícios de recursos, pois este

módulo teria condições de continuar a operar corretamente, assim que cessasse a causa do transiente.

A literatura não faz distinção rigorosa entre falhas/erros transientes e intermitentes. Os erros transientes e intermitentes são responsáveis por 80% a 90% das falhas eletrônicas em computadores [MAXION, 1987], aparecendo também em microprocessadores [BRODSKY, 1980]. Estes erros tornam-se problemáticos com o aumento da velocidade de operação, aumento da escala de integração dos dispositivos e diminuição dos valores de tensão para indicar níveis lógicos.

As falhas no comportamento merecem atenção especial, quando se trabalha na implementação de arquiteturas tolerante a falhas. Estas falhas estão relacionadas com o tipo de saída dada por um sistema. Este sistema pode ser um simples componente, ou pode ser um sistema de processamento complexo. A saída pode ser um sinal eletrônico, um dado, ou uma mensagem. Se diz que um sistema não está se comportando corretamente, se o seu comportamento não é consistente com as suas especificações. A especificação descreve a saída que deverá ocorrer, em resposta a uma sequência de eventos de entrada, e o intervalo de tempo dentro do qual esta saída deverá acontecer [CRISTIAN, 1989]. Na falha de comportamento por omissão, o sistema deixa de enviar algumas saídas, em resposta a algumas sequências de eventos de entrada. Na falha por queda, o sistema deixa de dar qualquer saída, a partir de um determinado instante. Na falha maliciosa, o sistema tem um comportamento

arbitrário. Ele pode omitir alguns sinais, pode enviar saídas fora da especificação, ou pode enviar saídas conflitantes. Neste caso, por exemplo, o sistema pode enviar informações conflitantes por rotas independentes, para processadores distintos [DRUMMOND, 1987]. Ele pode enviar para um sistema o sinal lógico "1" e para outro, um sinal lógico "0", ambos correspondentes a uma mesma saída.

Estas falhas de comportamento são críticas nos sistemas de computação tolerantes a falhas. Estes sistemas, como será visto mais adiante, fazem a comparação/votação de dados, para fornecerem uma saída correta. Além disso, estes sistemas precisam executar suas tarefas de forma síncrona. Em alguns casos, a sincronização depende da troca de informação entre os módulos do sistema. Quando as falhas são do tipo maliciosa, fica difícil para o sistema, tomar uma decisão sobre qual é o dado correto para ser enviado a saída. Também a sincronização pode ficar comprometida, se ocorrerem falhas de comportamento no envio de sinais para sincronização.

2.3.2. DETECÇÃO DE ERROS

Os mecanismos de detecção de erro, podem ser aplicados em partes específicas do sistema, por exemplo: detecção nas memórias, detecção no barramento entre o processador e a memória, detecção a nível de transferência

de registros. Podem também ser realizados após todos os resultados terem sido obtidos, mas antes que sejam fornecidos como saída: detecção ao término de uma tarefa, detecção no barramento de E/S do computador; chamados neste caso de "testes de último momento" [LEITE, 1987]. Os "testes de último momento", têm a vantagem de testar todas as atividades do sistema.

As técnicas de detecção de erro mais usadas são [SIEWIOREK, 1982] [AVIZIENIS, 1975] [LEITE, 1987] [OBAC RODA, 1988.a] [JOHNSON, 1984] : uso de cópias dos sistemas; testes de limite de tempo; testes de coerência; e códigos detectores de erro.

a) Uso de Cópias dos Sistema ("replication")

Esta técnica pode usar duas cópias idênticas do sistema (duplicação com comparação), ou pode usar mais cópias com votação [LEITE, 1987]. A técnica de duplicação é a mais simples de ser aplicada, onde a saída dos 2 sistemas são comparadas. Quando ocorre uma falha, as saídas deixam de ser idênticas e isto é detectado pelo elemento comparador. Com o uso de mais cópias, a comparação geralmente é feita em um elemento votador, e a igualdade ou não das comparações são indicadas pela adição de um elemento "detector de discordância" [SIEWIOREK, 1978].

Na aplicação desta técnica, deve-se tomar cuidado com o sincronismo entre as cópias, as comparações deverão ser realizadas em tempos bem determinados. Também

o elemento comparador/votador pode-se constituir em um ponto crítico de falha, um ponto vulnerável do sistema ("hardcore").

b) Testes de Limite de Tempo

Esta técnica se baseia no princípio que, algumas operações no sistema devem ser completadas dentro de um intervalo de tempo limitado. O teste de tempo esgotado ("time-out"), é um exemplo desta técnica [SIEWIOREK, 1982]. Estabelece-se um limite de tempo para um processo, se este não for completado neste intervalo, considera-se que há falhas no processo. Por exemplo, pode limitar-se o tempo de acesso a uma memória.

Outro teste conhecido, é o teste com o "relógio cão-de-guarda" ("Watchdog timer"). O relógio é mantido como um processo separado daquele que ele verifica. Quando o sistema está operando corretamente, ele envia um sinal ao relógio dentro de um intervalo de tempo especificado. Este sinal restabelece no relógio o seu valor original. Se este valor não é restabelecido dentro do intervalo de tempo previsto, considera-se a ocorrência de alguma falha no processo sob verificação.

c) Testes de Coerência ("reasonableness checking")

Estes testes verificam se os resultados de uma operação são consistentes com a faixa esperada de valores,

ou com uma função estabelecida a priori. Por exemplo, se a temperatura de um sistema está dentro do intervalo estabelecido. Ao nível de programação, tem-se a verificação de erros na compilação de um programa.

d) Códigos Detectores de Erros

Estes códigos se constituem na adição de informação extra a uma estrutura de dados básica. Com códigos detectores de erro é possível detectar erro na informação digital, utilizando um certo número adicional de bits.

Um dos códigos mais simples é a verificação da paridade. Como os computadores trabalham com dígitos binários, pode-se estabelecer uma forma na qual as palavras armazenadas ou transmitidas tenham sempre um número par (ou ímpar) de dígitos "1".

Existem outros tipos de códigos, como o da verificação da palavra soma ("Checksum"). Este teste é usado quando blocos de dados são transmitidos de um ponto a outro, ou quando são armazenados. É gerada uma palavra soma, somando todas as palavras de um bloco de dados. Na transmissão ou armazenamento de um bloco de dados a palavra é gerada e enviada (ou armazenada), geralmente no final do bloco. Na separação (ou leitura) do bloco, é gerada uma outra palavra com base nos dados recebidos. Após isto, as palavras são comparadas e se há discrepância, o sistema detecta uma condição de erro. Este

processo é útil quando se manuseia grandes blocos de dados, por exemplo no armazenamento em disco.

Há também os códigos que verificam se há erros e restauram a palavra com erro, como é o caso do código "Hamming" [SIEWIOREK, 1982] [JOHNSON, 1984]. Este código é usado principalmente para detectar erros de armazenamento em memórias tipo RAM ("Random Access Memory"). Estas memórias contribuem com 60% a 70% das falhas nos sistemas digitais, sendo propensas a sofrer falhas transientes [MAXION, 1987].

Um outro tipo de código é o CRC ("Cyclic Redundancy Code"), usado principalmente quando se transmite dados serialmente. Ele consiste na adição de palavra de verificação, ao bloco a ser transmitido. Esta palavra é construída através de um polinômio, que é múltiplo de um polinômio gerador padrão. Existem vários tipos de polinômios padrões, como o CRC-16 e o SDLC-IBM [RAMABADRAN, 1988].

2.3.3 - TÉCNICAS DE REDUNDÂNCIA

A base do conceito de tolerância a falhas, é o uso de cópias do sistema, de partes críticas do sistema, ou de circuitos e de programas. Estes recursos extras, não são estritamente necessários para a operação básica do sistema. A redundância é usada tanto para detecção e (ou)

maskamento de erros, quanto para substituir partes com defeitos no sistema, permitindo ao mesmo, executar seus processos, programas ou operações de forma correta. As falhas são consideradas acontecimentos reais na vida do sistema, e a redundância é usada como uma proteção automática, contra estes eventos.

A técnica de redundância pode ser implementada nas seguintes formas [JOHNSON, 1984] :

- Redundância na informação;
- Redundância temporal;
- Redundância por circuitos;
- Redundância por programação.

Estas formas não são mutuamente exclusivas, e na maioria das vezes, mais de uma técnica é empregada no desenvolvimento de projetos de computadores tolerantes a falhas. Além disso há em alguns casos interseção de uma técnica com outra.

2.4 - REDUNDÂNCIA NA INFORMAÇÃO

A técnica de redundância na informação, envolve o uso de códigos detectores de erros. Desta forma há a adição de informação extra, na estrutura de dados básica, que não é imprescindível à operação do sistema. Esta

informação permite identificar erros, e em alguns casos recuperar a palavra original, identificando os dígitos incorretos. Estes códigos são chamados códigos de correção de erro ("Error Correcting Codes" - ECC), sendo o código de "Hamming" um dos mais usados [MAXION, 1987] [JOHNSON, 1984] [SIEWIOREK, 1982].

2.5 - REDUNDÂNCIA TEMPORAL

A redundância temporal [SIEWIOREK, 1982], consiste basicamente na repetição de tarefas em diversos níveis. Ela permite fazer a distinção entre falhas transientes e falhas permanentes. O sistema executa a mesma operação mais de uma vez depois de detectado um erro, e se após isto, a condição de erro desaparece, então uma falha transiente é considerada como causa do erro. O sistema deve ter uma "capacidade de reserva temporal", para repetir a execução das tarefas.

Um problema com este tipo de redundância, é assegurar que o processador tenha os mesmos dados para manipular, cada vez que repete a operação. Depois de uma falha transiente, a integridade dos dados pode ficar comprometida.

2.6 - REDUNDÂNCIA POR CIRCUITOS

A redundância por circuitos ("Hardware Redundancy"), baseia-se no uso de cópias, tanto de circuitos a nível de componentes eletrônicos, como na reprodução de módulos inteiros [AVIZIENIS, 1975]. Ela é utilizada para detecção, para mascaramento de erros, e para aumentar o tempo de vida útil do sistema. A redundância por circuitos pode ser dividida em duas classes:

- redundância estática ou passiva;
- dinâmica ou ativa

2.6.1 - REDUNDÂNCIA ESTÁTICA POR CIRCUITOS

A redundância estática por circuitos é também chamada de "redundância mascarável". Ela é mascarável porque nesta técnica, o efeito da falha é dissimulado ou anulado pelos módulos ou unidades não falhas do sistema. É estática, porque a ocorrência de uma falha não é notificada, aos terminais do conjunto, e não há isolamento ou reparo da unidade falha. Todas as unidades executam as mesmas tarefas, estando ativas durante a operação do sistema.

A aplicação mais comum da redundância estática

por circuitos, é o sistema de redundância modular com três módulos (TMR - "Triple Modular Redundancy" [SIEWIOREK, 1982] [VON NEUMAN, 1956]). Nesta configuração, 3 módulos trabalham sincronamente, executando as mesmas tarefas ou programas. Os dados gerados pelos 3 módulos, são submetidos a um circuito votante por maioria (fig. 2.2). O conceito de votação por maioria, foi introduzido por Von Neuman em 1956 [VON NEUMAN, 1956]. Neste sistema, uma saída permanece válida, desde que 2 dos 3 módulos concordem, e o circuito de votação esteja livre de falhas. Os circuitos votantes podem ser construídos com a técnica de "intolerância a falhas", usando-se dispositivos de alta confiabilidade e os votadores submetidos a vários testes antes de entrarem em operação. Ou os circuitos votantes, podem também ser triplicados junto com os módulos [WAKERLY, 1976].

Na redundância estática, a votação é usualmente implementada através de circuitos, e o processo é aplicado votando-se bit a bit. Na fig. 2.3, tem-se um exemplo de um circuito votante. A arquitetura TMR suporta falhas em apenas 1 módulo. A falha em um segundo módulo, leva a perda da integridade do sistema, mesmo existindo ainda uma unidade sem problemas (os dois módulos falhos são maioria). Nestes sistemas há também a perda de 2/3 de capacidade de processamento, pois as 3 unidades executam a mesma tarefa, mesmo quando não há problemas. Há sistemas que tentam minimizar esta perda. Um exemplo é o C.vmp ("Computer, Voted Multi Processor" [SIEWIOREK, 1978]), que pode se reconfigurar e trabalhar com redundância tripla

Fig. 2.2 – REDUNDÂNCIA MODULAR TRIPLA – TMR .

Fig. 2.3 – Exemplo de circuito votante (bit a bit) .

apenas na execução de tarefas críticas, aproveitando em outras ocasiões a capacidade computacional, para operar como um sistema de multiprocessamento independente.

Uma extensão da arquitetura TMR é o sistema com redundância modular múltipla (NMR - "N Modular Redundancy") com votação, que pode suportar falhas em f módulos onde $N = 2f + 1$ [SIEWIOREK, 1982] [MAXION, 1987].

A arquitetura em anel [OBAC RODA, 1983.a, 1985], é outro exemplo de aplicação da redundância NMR. Nesta estrutura, os módulos são arranjados na forma de um "loop" (anel), no qual os módulos adjacentes são conectados por canais de comunicação unidirecionais ou bidirecionais (fig. 2.4). Os módulos recebem os dados de uma ou mais fontes, e a informação é processada e preparada para votação, que pode ser realizada em cada módulo. Neste tipo de arquitetura o elemento de votação, pode ser substituído por votação por programação (item 2.7.2)

2.6.2 - REDUNDÂNCIA DINÂMICA POR CIRCUITOS

A redundância dinâmica (ou ativa) por circuitos, é caracterizada pela "detecção do erro e a subsequente implementação de uma ação corretiva" [LEITE, 1987]. Esta técnica usualmente envolve reconfiguração, e a correção do erro passa por três fases:

Fig. 2.4 – Estrutura em anel com cinco módulos .
circulação horária

Fig.2.5 – Duplicação com comparação.

1. detecção da falha;
2. localização da unidade com defeito;
3. reconfiguração

A reconfiguração pode implicar na troca da unidade defeituosa por uma sem falhas, ou na simples desativação da unidade com defeito.

Entre os mecanismos de redundância dinâmica por circuitos destacam-se os seguintes:

- a. duplicação reconfigurável;
- b. redundância híbrida;
- c. votação adaptativa.

2.6.2 a. - DUPLICAÇÃO RECONFIGURÁVEL

Na redundância por circuitos com duplicação reconfigurável, dois módulos operam sincronamente, executando as mesmas tarefas e programas, mas apenas um deles alimenta a saída do sistema. Um elemento de comparação detecta as possíveis diferenças entre os dois módulos, segundo uma técnica de diagnose (por exemplo: relógio cão de guarda) implementada no projeto [SIEWIOREK, 1982] [ANDERSON, 1981]. Na presença de um erro, o sistema identifica o módulo com falha, reconfigura-se colocando-o fora de operação, e continua com o módulo correto, (fig. 2.5).

2.6.2 b. REDUNDÂNCIA HÍBRIDA

A técnica da redundância híbrida é uma combinação da redundância estática, com a adição da propriedade de reconfiguração, através do uso de um conjunto de módulos de reposição [MATHUR, 1976] [SIEWIOREK, 1982]. Ela tem as seguintes características:

1. a falha ou erro é mascarada;
2. a falha é detectada;
3. é localizada a unidade com defeito;
4. o sistema é reconfigurado.

Um exemplo de redundância híbrida, é o sistema NMR com um conjunto de módulos reservas. Neste caso há N módulos executando as mesmas tarefas, e tendo suas saídas submetidas a um circuito de votação por maioria. O sistema permanece nesta configuração, até o detector de discordância indicar que há um módulo com erro. Identificado este módulo, ele é substituído por um módulo sobressalente e o sistema retorna à configuração NMR com votação por maioria. O processo continua até que se esgotem os módulos reservas. Este mecanismo é ilustrado na figura 2.6.

Este tipo de arquitetura tem um inconveniente: a complexidade dos circuitos de chaveamento dos módulos sobressalentes, que pode comprometer a confiabilidade do sistema. Se houver um aumento excessivo de circuitos, a

Fig. 2.6 – Redundância Híbrida / NMR.

confiabilidade do sistema NMR pode ficar menor que a de um sistema sem redundância, (sistema simplex) [MAXION, 1987]. Um modo de diminuir este problema é a implementação de algumas funções de chaveamento através de programação, envolvendo a técnica de Redundância por Programação ("Software Redundancy" - item 2.7) [OBAC RODA, 1982, 1988.b, 1989]. Com isto, alguns sistemas híbridos têm ganho de confiabilidade sobre outros sistemas tomados como referência. Por exemplo: a arquitetura híbrida em anel, com sistema de circulação incluindo módulos ativos e desviando dos módulos reservas ("bypassing scheme"), proposta por [OBAC RODA, 1988. b, 1989], tem confiabilidade comparada ao TMR/híbrido. E o sistema duplex/híbrido, com o detector de concordância e mecanismos de chaveamento implementados por programação, tem confiabilidade maior que o TMR híbrido [OBAC RODA, 1982].

2.6.2 c. - VOTAÇÃO ADAPTATIVA

Na técnica de votação adaptativa, os módulos enquanto estiverem operando corretamente, fazem parte de um sistema com votação. Quando eles falham, vão sendo desligados do elemento votador, modificando o processo de votação dinamicamente [SIEWIOREK, 1982]. Existem vários esquemas que usam este processo [SIEWIOREK, 1982] [MAXION, 1987]. Um sistema que aplica esta técnica, é o NMR/Simplex [MATHUR, 1975]. De início o sistema é composto de N (N-ímpar) módulos, numa configuração NMR convencional com

votação. Quando um deles falha, ele e um outro são desligados do sistema. A remoção de 2 módulos é feita para evitar empate no processo de votação. Conforme vão se esgotando os módulos, o sistema acaba em um sistema Simplex, um módulo ativo sem redundância.

Uma alternativa neste processo seria a implementação da técnica de duplicação reconfigurável, antes do sistema se reduzir a um único módulo. Isto seria feito utilizando como módulos reservas, os módulos em bom funcionamento, que foram desligados junto com os falhos.

2.7 - REDUNDÂNCIA POR PROGRAMAÇÃO

A Redundância por Programação consiste na adição de técnicas de programação, que ajudam na detecção e recuperação de erros no sistema [AVIZIENIS, 1976]. O termo redundância por programação, significa o uso de recursos de programação para tolerar falhas no sistema [WENSLEY, 1978]. É diferente do conceito redundância em programação, ou equivalentemente tolerância a falhas em programação ("Software Fault Tolerance") [MAXION, 1987]. Tolerância a falhas em programação significa o uso de técnicas de tolerância a falhas (testes, detecção de erros, verificação, redundância), para lidar com falhas e erros que possam ocorrer nos programas ("Software"), por exemplo: programa mal feito, implementação inadequada, especificação imprópria, falhas no sistema operacional [MINI CURSO, 1987] [OZAKI, 1988].

A redundância por programação, a redundância por circuitos e a redundância em programação, são técnicas que se complementam.

A redundância por programação pode ser tratada sob dois aspectos:

- Replicação de recursos de programação;
- Votação por programação.

2.7.1 - REPLICAÇÃO DE RECURSOS DE PROGRAMAÇÃO

A replicação de recursos, inclui simplesmente a adição extra de programas, segmentos de programas, instrução e microprogramas, como mecanismos para tolerar falhas em um sistema. As principais formas são [LEITE, 1987] [JOHNSON, 1984] :

1. cópias múltiplas de programas e dados críticos;
2. programas de testes e diagnose em vários níveis;
3. características de tolerância a falhas no próprio programa
(usando redundância em programação).

Um exemplo desta técnica é o teste de coerência, onde programas são adicionados para verificar se os resultados produzidos pelo sistema estão dentro de um intervalo de valores aceitáveis. Um sistema de controle

por computador tolerante a falhas, pode fazer uso desta técnica, aplicando-a tanto nos atuadores do controle, como nos sensores. Outro caso é o uso de uma mistura de redundância temporal com redundância por programação, que pode consistir em testes periódicos no sistema.

Há ainda o uso de várias versões de um mesmo algoritmo. Estes programas executam as mesmas tarefas, mas podem usar métodos diferentes ou codificação diferente, e podem ser executados sequencialmente em um mesmo processador ou simultaneamente em vários processadores. Os resultados obtidos por estas versões são comparados, visando a detecção de falhas. Este processo é o equivalente em programação da redundância NMR, sendo chamado "programação a N-versões" [CHEN, 1978]. Ele pode ser usado tanto como técnica de redundância por programação como pela redundância em programação.

A vantagem principal da redundância por programação, é a sua flexibilidade, permitindo alteração e refinamento dos programas. É mais fácil alterar a redundância na programação do que a redundância nos circuitos. Além disso, esta técnica pode ser aplicada em computadores comerciais, sem alterações nos circuitos ("hardware").

2.7.2 - VOTAÇÃO POR PROGRAMAÇÃO

A votação por programação consiste no uso de

programas, em vez de circuitos, para a implementação do elemento votador usado na redundância modular múltipla, NMR. Ela pode usar o mesmo conceito de votação por maioria introduzido por von Neuman [VON NEUMAN, 1956]. A votação por programação tem a vantagem de não ter a natureza estática, da votação por circuitos, podendo-se modificar os algoritmos utilizados na votação. Com o uso desta técnica, pode-se simplificar os mecanismos de chaveamento das unidades sobressalentes usadas na aplicação da redundância híbrida [OBAC RODA, 1982], tanto nos aspectos do uso de circuitos, como no processo de controle destes circuitos. A votação por programação apresenta as seguintes características [LOMBARDI, 1982] [OBAC RODA, 1985] :

1. Os programas não sofrem desgaste como os circuitos
2. Fornece informação mais completa sobre o sistema, uma vez que o fluxo do programa será determinado pelo comportamento correto ou incorreto do módulo
3. Tem flexibilidade para variar o grau de proteção a falhas [WENSLEY, 1978]. Sendo portanto, adequada para sistemas com redundância dinâmica, tais como os sistemas híbridos [OBAC RODA, 1982] e aqueles com votação adaptativa
4. Permite a sua adaptação para diversas aplicações, sem grandes alterações na operação do sistema. A votação por programação pode ser usada na votação de

sinais contínuos e sinais em tempo real
[OBAC RODA, 1982, 1988.b]

5. As exigências de sincronização são menos críticas [WENSLEY, 1978]. Com isto, pode-se optar por arquiteturas redundantes fracamente conectadas [MINI CURSO, 1987], que operam com circuitos de relógio independentes. Nos circuitos fortemente conectados, as unidades operam sincronamente executando o mesmo programa, e a temporização fica a cargo de um único relógio, que neste caso pode se constituir em um ponto vulnerável na arquitetura

6. Há um aumento no tempo de processamento. Isto se constitui em um ponto negativo na votação por programação. É necessário considerar na aplicação desta técnica, o fator tempo, que pode impor restrições a sua aplicação. O tempo adicional usado na votação será função do grau de tolerância a falhas e da topologia do sistema [LOMBARDI, 1982]

Técnicas para Implementação de Votação por Programação

1. Executar a tarefa 3 vezes no mesmo processador e votar os resultados. Algoritmos diferentes podem ser usados em cada tarefa. Este procedimento também

é utilizado pela técnica de redundância em programação, para tolerar falhas em programas.

2. Executar a mesma tarefa em módulos diferentes e usar uma unidade inteligente para votar [OBAC RODA, 1983.a]. Esta unidade executaria a votação por programação, de acordo com um algoritmo pré-estabelecido. Ele somente é vantajoso, se a unidade inteligente tiver confiabilidade muito maior que os módulos processadores.

3. A mesma tarefa pode ser executada em diferentes módulos e a votação realizada por programação dentro de cada unidade. Isto requer o intercâmbio de dados entre os módulos e a seleção de um dos módulos (com operação correta) para dar saída ao sistema.

2.8 - CONCLUSÃO

Foram apresentadas várias técnicas para tolerância a falhas em sistemas de computação.

Não é possível uma classificação rigorosa de cada técnica em uma determinada classe, o que predomina é a cooperação e a intersecção entre as metodologias. Um grande campo de pesquisa que se tem aberto, é o desenvolvimento de projetos tolerantes a falhas usando os circuitos com Grande Escala de Integração (VLSI). Estes circuitos são mais suscetíveis a falhas transientes e a interferência de fatores externos por causa do alto grau

de integração e dos baixos níveis de tensão elétrica utilizados. A pesquisa em Tolerância a Falhas neste campo, envolve vários níveis, desde o projeto do circuito em si, até o desenvolvimento de sistemas de computação completos, sempre buscando um equilíbrio entre confiabilidade, disponibilidade, desempenho e custo. É neste sentido que este trabalho busca desenvolver uma nova tecnologia, que é o uso da arquitetura em anel, com votação por programação, em um sistema de uso geral, baseado no microprocessador Z-80 e com sistema operacional CP/M.

CAPÍTULO 3

IMPLEMENTAÇÃO DA ARQUITETURA EM ANEL TOLERANTE A FALHAS

3.1 - INTRODUÇÃO

A Arquitetura em Anel introduzida no capítulo anterior, quando se tratou das arquiteturas com redundância, tem a grande vantagem e flexibilidade de permitir a votação através de programação. Neste capítulo, são discutidas as características desta arquitetura e as estratégias básicas que podem ser usadas, para a execução da votação: a votação majoritária e a votação parcial. É também analisado o grau de tolerância a falhas, que estas estratégias de votação proporcionam ao sistema. Isto é feito através das expressões de confiabilidade deduzidas para ambas.

A implementação do Sistema em Anel com 3 módulos também é apresentada, tanto do ponto de vista dos circuitos utilizados ("hardware"), como do sistema de gerenciamento do esquema de tolerância a falhas ("software"). Cada módulo é dividido em 3 subsistemas: de processamento, de entrada e saída e de comunicação no anel. Estes subsistemas são discutidos separadamente.

Na parte de gerenciamento da tolerância a falhas, são apresentadas as rotinas desenvolvidas para

votação, comunicação no anel, e acesso ao barramento de entrada e saída. Os mecanismos de votação e comunicação no anel, foram projetados de forma a manter o sistema sincronizado, pois cada módulo tem o seu próprio relógio não estando sincronizado com os demais. Neste sistema, a votação é na saída dos dados, e ele deve estar sincronizado quando esta saída é produzida.

3.2 - A ARQUITETURA EM ANEL

A Arquitetura em Anel é um tipo de estrutura NMR, onde os módulos são arranjados em um "loop" (anel), e não há um circuito votante comum [OBAC RODA, 1983.a]. Na figura 3.1 temos o esquema desta arquitetura com 3 módulos. Cada módulo recebe os dados de uma fonte, estes dados são processados e o resultado do processamento é passado para os outros módulos através de um canal de comunicação, havendo circulação dos dados através do anel. A circulação pode ser unidirecional ou bidirecional, podendo ainda usar canais redundantes. Sendo unidirecional a circulação pode ser escolhida no sentido horário ou anti-horário. Quando os módulos têm dados suficientes, cada módulo executa a votação por programação, de acordo com um determinado algoritmo. O resultado da votação é indicado por sinalizadores de erro, cujo estado lógico é usado para a seleção de apenas um módulo (sem falhas), para dar saída ao sistema. Os sistemas periféricos podem ou não ser triplicados. Para a votação dos dados podem

Fig. 3.1 : Estrutura em anel com 3 módulos
circulação unidirecional horária

ser usadas duas estratégias:

1. Votação Majoritária: sobre todos os dados
2. Votação sobre um número parcial de dados

No primeiro caso, cada módulo recebe os dados de todos ou outros módulos para votação. No segundo, é feita a votação em cada módulo sobre um número menor de dados. O limite é a comparação de dados de dois módulos adjacentes.

3.2.1 - CONSIDERAÇÕES SOBRE A VOTAÇÃO POR PROGRAMAÇÃO NA ARQUITETURA EM ANEL

As duas estratégias (votação majoritária e votação parcial) utilizadas na votação por programação na Arquitetura em Anel, dependem da comunicação entre os módulos no anel e vão fornecer diferentes graus de tolerância a falhas para o sistema.

3.2.1a - VOTAÇÃO MAJORITÁRIA

A votação majoritária consiste na implementação do conceito de votação de Von Neuman [VON NEUMAN, 1956], onde o sistema tem N módulos ($N = 2n + 1$, n inteiro) e pode tolerar n unidades com defeitos. O resultado correto de um processo é selecionado como aquele que tem o maior número ($n + 1$) de resultados coincidentes. Com esta estratégia, na Arquitetura em Anel, todos os módulos

recebem os dados de todos os outros módulos, e cada módulo, através de programação, determina qual dado tem o maior número de coincidências, sendo considerado como resultado correto. Após isto, um dos módulos é selecionado para dar saída ao sistema. Esta seleção pode ser efetuada também por programação, ou pode ser efetuada por um sistema de seleção, implementado por circuitos, com base em indicadores de erro fornecidos por cada um dos módulos [OBAC RODA, 1983.a]. Diferentes algoritmos de votação podem ser implementados em cada módulo, adicionando-se também um certo grau de proteção a falhas em programação.

No esquema de votação por maioria, $n + 1$ módulos devem estar em operação correta e serem capazes de se comunicar, de forma a ter em pelo menos um módulo, a maioria de dados corretos. Considerando a circulação unidirecional são necessários $N - 1$ circulações de dados, para que isto ocorra. A figura 3.2 ilustra a circulação de dados.

Este esquema pode tolerar falhas em 2 módulos adjacentes ($N > 3$), mas quando os módulos falhos não são adjacentes, dependendo do tipo de falhas o sistema pode entrar em colapso, qualquer que seja N .

Nas figuras 3.3 e 3.4 estão ilustradas as influências das falhas, em módulos adjacentes e não adjacentes, na circulação dos dados. No caso (a), embora se tenha falhas em 2 módulos, estes não bloqueiam a circulação, porque são módulos adjacentes. Há 3 módulos (maioria) funcionando corretamente e adjacentes de maneira que um deles tem maioria de dados corretos. No exemplo, o

Sistema em Anel com 5 módulos
circulação unidirecional

Subsistema de comunicação

Subsistema de
entrada e saída

Esquema de um módulo do Anel

MÓDULO		M1	M2	M3	M4	M5
DADO APÓS PROCESSAMENTO		D_1^1	D_2^2	D_3^3	D_4^4	D_5^5
CIRCULAÇÃO	1º	D_5^5	D_1^1	D_2^2	D_3^3	D_4^4
	2º	D_5^4	D_1^5	D_2^1	D_3^2	D_4^3
	3º	D_5^3	D_1^4	D_2^5	D_3^1	D_4^2
	4º	D_5^2	D_1^3	D_2^4	D_3^5	D_4^1

Dados para votação no
Módulo 1 após 4 circulações .

D_j^i - é o resultado do processamento do
módulo i transmitido pelo módulo j .

D_j^j - é o resultado do próprio processa-
mento .

Fig. 3.2 : Circulação dos dados unidirecional em um anel com 5 módulos e votação majoritária .

módulo 1 tem maioria de dados corretos. As falhas mostradas no caso (c), implicam no bloqueio da circulação de dados através do anel, de forma que após $(N - 1)$ circulações nenhum dos módulos têm maioria de dados corretos. Já uma falha do tipo (b), onde apenas parte do subsistema de comunicação está comprometido, ainda permite que se tenha um módulo com maioria de dados corretos. Este módulo poderia dar saída ao sistema.

O grau de tolerância a falhas nos sistemas NMR em Anel com votação majoritária, é diferente daquele para o NMR com votação por circuitos (NMR-mascarável). Este último pode tolerar n módulos falhos ($n + 1$ em operação correta, $N = 2n + 1$).

O NMR em Anel, pode tolerar falhas em 1 módulo com 100% de segurança. A falha em um segundo módulo vai depender do tipo de falha e da localização do módulo, se é ou não adjacente a um módulo já com defeito. Por outro lado, no sistema NMR mascarável, o elemento votador implementado por circuitos é um ponto vulnerável da arquitetura, sendo necessário usar redundância também nas unidades de votação. Dependendo do número de módulos (N), e do número de linhas submetidas a votação (a votação é bit a bit), a confiabilidade do sistema como um todo (módulos + votadores), pode ficar comprometida [WAKERLY, 1976].

No Sistema em Anel, com a votação sendo feita por programação, a confiabilidade do votador está embutida na confiabilidade do módulo, não importando o número de módulos. O ponto vulnerável é o bloqueio da circulação de

a) Falhas nos módulos 2 e 3 adjacentes.

b) Falha parcial no subsistema de comunicação do módulo 2 e falha total no módulo 4. Os módulos 2 e 4 não são adjacentes.

c) Falhas nos módulos 2 e 4 não adjacentes.
Bloqueio da circulação.

Fig. 3.3 : Sistema com 5 módulos, mostrando falhas em 2 módulos.
Circulação unidirecional

MÓDULO		M1	M2	M3	M4	M5
DADOS APÓS PROCESSAMENTO		D ₁ ¹	D ₂ ^X	D ₃ ^X	D ₄ ⁴	D ₅ ⁵
CIRCULAÇÃO	1º	D ₅ ⁵	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ⁴
	2º	D ₅ ⁴	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	3º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	4º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X

X - Dado corrompido devido a falhas .

O módulo 1 tem maioria de dados corretos .

(a)

MÓDULO		M1	M2	M3	M4	M5
DADOS APÓS PROCESSAMENTO		D ₁ ¹	D ₂ ²	D ₃ ³	D ₄ ^X	D ₅ ⁵
CIRCULAÇÃO	1º	D ₅ ⁵	D ₁ ²	D ₂ ^X	D ₃ ^X	D ₄ ^X
	2º	D ₅ ^X	D ₁ ⁵	D ₂ ^X	D ₃ ^X	D ₄ ^X
	3º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	4º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X

O módulo 2 tem maioria de dados corretos .

(b)

MÓDULO		M1	M2	M3	M4	M5
DADOS APÓS PROCESSAMENTO		D ₁ ¹	D ₂ ^X	D ₃ ³	D ₄ ^X	D ₅ ⁵
CIRCULAÇÃO	1º	D ₅ ⁵	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	2º	D ₅ ^X	D ₁ ⁵	D ₂ ^X	D ₃ ^X	D ₄ ^X
	3º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	4º	D ₅ ^X	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X

Nenhum dos 5 módulos tem maioria de dados corretos .

(c)

Fig. 3.4 : Circulação dos dados para os casos da figura 3.3 .

dados corretos, como mostrado nas figuras 3.3 e 3.4 (c).

Para minimizar o efeito do bloqueio na circulação dos dados, e permitir que os módulos em operação correta recebam a maioria de dados corretos, pode-se introduzir um esquema em que os dados circulam n vezes em cada direção. A figura 3.5 mostra este processo. Após 4 circulações (2 em cada direção) os módulos em operação correta, receberam a maioria dos dados corretos. Para um sistema com 3 módulos, este esquema de circulação pode permitir a identificação do tipo de falha, se no subsistema de comunicação ou no processamento. A figura 3.6 ilustra o que ocorre.

Sendo a falha só em parte do subsistema de comunicação, todos os módulos têm a maioria de dados corretos. Além disso o módulo M1 tem todos os dados corretos. Já com a falha no processamento, apenas M1 e M3 têm a maioria de dados corretos, mas nenhum tem todos. Como a votação é por programação, podem ser implementadas rotinas de verificação, que de acordo com a configuração dos dados dentro de um módulo, permitam identificar onde ocorreu a falha.

Na referência [OBAC-RODA, 1983] foi calculada a expressão para a confiabilidade para o Sistema NMR em Anel empregando a votação majoritária:

$$R_{ANEL} = R_m^N + N \cdot \left[\sum_{k=1}^n \binom{n-1}{k-1} \cdot R_m^{N-k} (1 - R_m)^k \right] \quad (3.1)$$

onde

$$N = 2n + L$$

$$R_m = \text{confiabilidade do módulo}$$

Falhas nos módulos 2 e 3 .

MÓDULO			M1	M2	M3	M4	M5
DADO APÓS PROCESSAMENTO			D ₁ ¹ D _{1*}	D ₂ ^X	D ₃ ^X	D ₄ ⁴ D _{4*}	D ₅ ⁵ D _{5*}
CIRCULAÇÃO		1º	D _{5*} ⁵	D ₁ ^X	D ₂ ^X	D ₃ ^X	D _{4*} ⁴
	2º	D _{5*} ⁴	D ₁ ^X	D ₂ ^X	D ₃ ^X	D ₄ ^X
	3º	D ₂ ^X	D ₃ ^X	D ₄ ^X	D _{5*} ⁵	D _{1*} ¹
	4º	D ₂ ^X	D ₃ ^X	D ₄ ^X	D _{5*} ¹	D ₁ ^X

* Dados dos módulos em operação correta .

Fig. 3.5 : Circulação bidirecional dos dados .

MÓDULO		M1	M2	M3
PROCESSAMENTO		D_1^1	D_2^2	D_3^3
CIRCULAÇÃO		D_3^3	D_1^1	D_2^X
	D_2^2	D_3^X	D_1^1

a) Falha no subsistema de comunicação.

MÓDULO		M1	M2	M3
PROCESSAMENTO		D_1^1	D_2^X	D_3^3
CIRCULAÇÃO		D_3^3	D_1^X	D_2^X
	D_2^X	D_3^X	D_1^1

b) Falha no sistema de processamento.

Fig. 3.6: Ilustração da ocorrência das falhas no Sistema em Anel com 3 módulos.

Para a dedução desta equação foi considerado que:

1. Cada módulo tem taxa de falha λ_m constante;
2. A confiabilidade do sistema em $t = 0$ é 1;
3. As falhas nos módulos são estatisticamente independentes;
4. Há apenas 2 estados operacionais para cada módulo: estado em operação, ou estado errôneo;
5. Uma falha no sistema de comunicação é considerada uma falha do módulo. Neste caso o módulo corrompe os dados que chegam dos outros módulos;
6. Não foi considerado na dedução desta equação, os casos de falhas apenas no subsistema de comunicação, como o exemplo mostrado na figura 3.3b. Falha deste tipo está sendo considerada falha do módulo.

A expressão 3.1 permite fazer a comparação do sistema NMR-Mascarável com o sistema NMR-Anel. A expressão para a confiabilidade do sistema NMR-Mascarável, é [SIEWIOREK, 1982] :

$$R_{NMR} = R_m^N + \sum_{k=1}^n \binom{N}{k} R_m^{N-k} (1 - R_m)^k$$

com $N = 2n + 1$

(3.2)

Onde foi considerado, que a confiabilidade do elemento votador é igual a 1 ($R_v = 1$). Para valores diferentes R_m é substituído por $(R_m R_v)$, [WAKERLY, 1976].

Na figura 3.7 está traçada a confiabilidade do sistema em função da confiabilidade de um módulo, para alguns valores de N, tanto para a Arquitetura em Anel, como para o sistema NMR-Mascarável. A confiabilidade do sistema NMR-Mascarável é maior do que a confiabilidade do sistema em Anel, sendo igual somente para $N=3$. Ao contrário do sistema NMR-Mascarável,

na Arquitetura em Anel a confiabilidade diminui conforme aumenta o número de módulos. Isto acontece [DOBAC RODA, 1983.a, 1983.b] devido a intolerância do meio de transmissão, com a ocorrência de situações de bloqueio do canal de comunicação conforme mostrado nas figuras 3.3 e 3.4. Para que um módulo tenha a maioria de dados corretos, é necessário que pelo menos $n+1$ módulos adjacentes estejam em operação correta. No Anel, aumentando o grau de redundância, aumenta a probabilidade de falhar mais módulos, bloqueando a circulação, e com isto a confiabilidade do sistema diminui.

3.2.1b - VOTAÇÃO PARCIAL

Na votação parcial, um número menor de dados é votado em cada módulo. Não há a necessidade de cada módulo receber os dados de todos os outros módulos. A votação é executada comparando-se os dados recebidos e o número mínimo de dados a serem votados é limitada pela condição de existência de 2 módulos adjacentes em bom funcionamento. Neste esquema, cada módulo recebe J dados, sendo habilitado para dar saída ao sistema o módulo que tiver J dados coincidentes corretos. Para isto, é necessário que haja no sistema de N módulos, pelo menos J módulos adjacentes funcionando corretamente.

Para a votação parcial são necessárias as seguintes condições [DOBAC RODA, 1983.a, 1985]:

1. O módulo deve ser capaz de detectar a real concordância dos dados recebidos. Não pode indicar concordância entre um processamento incorreto e os dados recebidos de outro módulo com falha no subsistema de comunicação. Este tipo de situação pode ser evitada, através de rotinas que testem periodicamente o subsistema de comunicação.

2. O mecanismo de sinalização de erros (não concordância entre os J dados), deve indicar corretamente o resultado da votação.

A expressão para a confiabilidade nesta estratégia de votação, é determinada usando: a definição para a votação parcial; as 2 condições acima e também as 6 considerações listadas para a votação total. A expressão obtida é:

$$R_{\text{ANEL PARCIAL}} = R_m^N + \sum_{k=1}^{N-j-2} \binom{N}{k} R_m^k (1-R_m)^{N-k} + N \sum_{\ell=N-j-1}^{N-j} \binom{N-j-1}{N-j-\ell} R_m^{N-\ell} (1-R_m)^\ell$$

(3.2)

Com

$$N = 2n + 1$$

$$J \geq 2$$

A confiabilidade do sistema com votação parcial, para alguns valores de N e de J, está mostrada na figura 3.8. Da figura, pode-se observar que a confiabilidade do sistema aumenta, conforme diminui o número de dados a serem votados. Por exemplo: N = 7, J = 2 e 3 ; N = 5, J = 2 e 3. Para N = 5 e J = 3 tem-se o caso de votação majoritária, e a expressão obtida para a confiabilidade do sistema é a mesma tanto a partir da equação 3.2 como da equação 3.1. O mesmo ocorre com N = 7 e J = 5. Para N = 3 o Sistema em Anel com votação parcial, tem os mesmos valores de confiabilidade, que o sistema com votação total e que o sistema TMR.

Na votação parcial a melhora de desempenho do sistema com a diminuição dos dados votados, se deve a exigência de um número menor de módulos adjacentes, operando corretamente. Na figura 3.3, o sistema com uma falha do tipo mostrado em (c) e votação majoritária, entra em colapso e nenhum dos módulos tem condições de dar saída ao resultado do processamento ocorrido nos módulos, embora

Fig. 3.7 : Confiabilidade do sistema em função da confiabilidade de um único módulo . Sistema em Anel com votação majoritária e Sistema NMR mascarável .

Fig. 3.8 : Confiabilidade do sistema em função da confiabilidade de um único módulo , para o Anel com votação parcial .

haja 3 módulos operando corretamente. Já com o sistema de votação parcial e sobre 2 dados, há dois módulos adjacentes em operação correta (módulos 1 e 5) e o módulo 1 pode dar saída ao sistema.

Neste esquema de votação podem ser toleradas falhas em até $n + 1$ módulos ($N = 2n + 1$), desde que haja $J = n$ módulos adjacentes operando corretamente.

Na figura 3.9 estão mostrados alguns exemplos de falhas que a Arquitetura em Anel com votação parcial de dados pode tolerar, para $N = 5$ e $J = 2$.

A limitação no grau de tolerância a falhas neste esquema de votação, fica sujeita a: implementação dos mecanismos indicadores de erro; ao processo de seleção para garantir que o módulo com J dados corretos alimente a saída do sistema; e ao processo usado para sincronizar a operação de votação.

3.2.1c - ANÁLISE

Considerando as duas estratégias de votação, aquela sobre um número parcial de dados é a que apresenta melhores resultados. O elemento crítico na Arquitetura em Anel é o subsistema de comunicação. O desempenho desta arquitetura pode ser melhorado, se forem usados subsistemas que garantam a circulação dos dados mesmo em caso de falha no módulo correspondente, evitando o bloqueio do canal de comunicação.

Fig. 3.9 : Falhas toleradas pelo sistema em Anel com $N=5$ e $J=2$. (votação parcial).

3.3 - IMPLEMENTAÇÃO DA ARQUITETURA EM ANEL

Foi apresentado no item anterior, que o fator crítico na Arquitetura em Anel é o subsistema de comunicação. Porém, só um subsistema eficiente não basta, é necessário que as votações dos dados trocados sejam efetuadas todas ao mesmo tempo, ou no mesmo "ciclo de votação", [DAVIES, 1978]. A votação é realizada sobre a saída do processo, sendo necessário que estas saídas estejam sincronizadas.

Além da sincronização e comunicação é necessário que cada módulo indique e sinalize corretamente o resultado da votação, e que seja habilitado um módulo para dar saída ao sistema.

Como implementar estes requisitos em um sistema real, de uso geral, foi o objetivo deste trabalho.

Para isto escolheu-se trabalhar com uma Arquitetura em Anel com 3 módulos, que aceita o sistema operacional CP/M [HOGAN, 1983], sendo portanto de uso geral. Com 3 módulos a confiabilidade do sistema é a mesma, seja com votação majoritária ou com votação parcial, sendo também igual a do sistema TMR-Mascarável.

3.3.1 - CARACTERÍSTICAS DO SISTEMA DESENVOLVIDO

O Sistema em Anel proposto tem as seguintes características [FISCHER, 1989]:

1. O sistema tem 3 módulos (fig. 3.1) e pode

- tolerar falhas gerais em 1 módulo;
2. O projeto é modular e pode operar sem um dos módulos ativados. Isto facilita a manutenção e aumenta a disponibilidade do sistema;
 3. É transparente ao usuário. Este não precisa saber que está programando um computador tolerante a falhas;
 4. O sistema operacional escolhido é o CP/M [HOGAN, 1983], adaptado para a Arquitetura em Anel. Desta forma todos os programas existentes para o CP/M poderão ser usados em uma máquina tolerante a falhas.
 5. Os módulos são conectados por dutos unidirecionais com circulação no sentido horário.
 6. Cada resultado de processamento interno é enviado aos outros módulos. Cada módulo compara este dado com o resultado de seu próprio processamento.
 7. A comparação dos dados entre os módulos é implementada por programação. A estratégia de votação é a de votação parcial.
 8. O resultado da comparação é indicado por um sinalizador de erro, que irá controlar a saída do sistema.
 9. Somente um módulo alimenta a saída do sistema. O que tiver pelo menos 2 dados coincidentes.
 10. É considerado que o módulo falho transmite dados incorretos, ou é incapaz de enviar

dados.

11. Cada módulo tem sincronização independente, não há uma referência de tempo comum.

3.3.2 - CONFIGURAÇÃO DO SISTEMA

Na Arquitetura em Anel desenvolvida, cada módulo é constituído por um microprocessador Z-80, 1 bloco de memória ROM com 32K, 1 bloco de memória RAM com 64K, dividido em 2 páginas de 32K. Quando não está sendo usada, a página ROM pode ser substituída por uma das páginas RAM. Na implementação atual, cada módulo tem o seu próprio controlador de disco flexível. Os 3 módulos se comunicam no anel através de portos paralelos. Todos acessam um único terminal de vídeo que é considerado a entrada/saída do sistema. Para isto todos os módulos têm linhas comum de acesso ao dispositivo. O barramento de E/S foi projetado de forma que o sistema possa vir a acessar um único periférico, qualquer que seja ele (controlador de disco flexível, impressora, conversor A/D, terminal de vídeo). Para evitar o acesso de 2 módulos ao mesmo tempo, a um mesmo periférico, foi projetado um esquema de seleção através de uma "linha bastão". O módulo que ganha o acesso, sinaliza a "linha bastão" e até que ele retire o sinal, os outros módulos não podem utilizar o barramento de E/S. Este procedimento pode ser usado em todas operações de leitura de periféricos e nas operações de saída, para maior segurança.

Cada módulo tem um Sistema Monitor, residente em

ROM, assim como o BIOS (Sistema Básico de Entrada e Saída- [HOGAN, 1983]) do CP/M. Foi usado parte do Sistema Monitor desenvolvido para o microcomputador LIE [TRAINA, 1986], assim como o módulo básico desta máquina tal como: esquema de relógio ("Clock"), configuração dos blocos de memória, endereçamento de periféricos. Este microcomputador foi desenvolvido no Laboratório de Instrumentação Eletrônica do Departamento de Física e Ciências dos Materiais e está descrito nas referências [TRAINA, 1986] e [MACHADO TRAINA, 1987]. Usando o núcleo básico foi projetada a Arquitetura em Anel, adicionando os portos paralelos para comunicação, o sistema de seleção para acesso ao barramento de entrada e saída, o sistema para sincronização dos módulos, e o esquema para sinalização do resultado da votação. Foram desenvolvidas as rotinas de comunicação no Anel, as rotinas de votação de dados, e o sistema de gerenciamento da Arquitetura Tolerante a Falha em Anel.

Para alimentação do sistema podem ser usadas fontes de tensão independentes.

3.4 - A ARQUITETURA DE UM MÓDULO

Cada módulo pode ser dividido em 3 subsistemas básicos: o subsistema de processamento, o subsistema de comunicação no Anel, e o subsistema de acesso ao barramento de entrada e saída. Todos os processos ocorrem sob o gerenciamento do microprocessador Z-80. A opção por este microprocessador se deu em virtude de ser um

microprocessador com ampla bibliografia tanto a nível de componentes periféricos como a nível de programação.

Além disso, como o projeto é uma máquina experimental, a disponibilidade de recursos para implementação é fundamental. O microprocessador Z-80 é ainda um dispositivo útil, para o desenvolvimento de projetos de controle de processos dedicados [CHANDE, 1989], sendo justamente nesta área que as técnicas de Tolerância a Falhas podem contribuir para maior segurança de funcionamento.

A seguir serão descritos cada um dos subsistemas.

3.4.1 - O SUBSISTEMA DE PROCESSAMENTO

O diagrama de blocos do subsistema de Processamento está mostrado na figura 3.10. O circuito de temporização fornece duas saídas de 2 MHz e 4MHz, multiplexadas. O microprocessador trabalha com uma das duas frequências sendo a seleção feita através do programa monitor. Nas atividades de acesso a disco, o microprocessador opera com 4 MHz, pois é feita uma rápida transferência de setores baseada na linha de interrupção não mascarável (NMI) do Z-80 e na geração de sinais de pedido de dados (DRQ) e pedido de finalização de transferência (INTRQ) do circuito de interface da controladora de disco (2793). Este sistema foi implementado por [TRAINA 1986], baseado na referência

Fig. 3.10: Arquitetura de um módulo do anel – Detalhe do subsistema de processamento .

[MARSHALL, 1983]. Nas demais atividades o Z-80 também opera com 4 MHZ. O sistema usa 2 MHZ, quando são enviados os sinais de controle, para a programação dos circuitos da interface para comunicação serial com o terminal de vídeo.

O circuito de "Reset", está ligado a uma entrada no barramento E/S, quando o botão de reset é acionado, o Reset é ativado aos 3 módulos ao mesmo tempo.

A interface serial é usada para comunicação com o terminal de vídeo. É composta por 2 circuitos integrados: o 8251 (INTEL) que é uma interface de comunicação serial programável e o contador programável 8253 (INTEL), usado para temporização dos sinais transmitidos e recebidos pela 8251 [INTEL, 1986]. Cada módulo possui sua própria interface serial.

O seletor de dispositivos e endereços, é usado para habilitar tanto os dispositivos internos ao subsistema de processamento, como para selecionar os blocos ROM e RAM, além de habilitar os subsistemas de comunicação no anel e o de entrada/saída. Desta forma, estes subsistemas não são completamente independentes do subsistemas de processamento. Se o subsistema de comunicação for construído de forma que a recepção seja independente da transmissão, como nos esquemas mostrados nas figuras 3.2, uma falha em parte do subsistema de comunicação, por exemplo somente na transmissão, pode permitir que o módulo continue ativo no anel. Nesta implementação, isto não foi feito, e uma falha no subsistema de comunicação é considerada uma falha do

módulo. A separação em subsistemas foi feita para facilitar a descrição das atividades de cada módulo.

O esquema de seleção de páginas de memória, permitiu que fossem implementados 64 Kbytes de memória RAM e 32 Kbytes de ROM. Com isto, foi criado no sistema um ambiente de programação favorável para a execução do Sistema Operacional CP/M de 64K, oferecendo para os programas de aplicação a maior quantidade de memória disponível em um ambiente CP/M. [TRAINA, 1986]. Toda a bibliografia de programas existente para o CP/M poderá ser usada em uma máquina tolerante a falhas, mas transparente ao usuário. Também todos os programas utilitários tais como compiladores, editores, programas montadores, poderão ser executados nesta máquina tolerante a falhas, sob o Sistema Operacional CP/M.

Outra vantagem, é que usando um Sistema Operacional já bem estabelecido e aceito, como é o CP/M, se evita falhas em programação ("software faults") que poderiam ocorrer no desenvolvimento do sistema, por exemplo: especificação imprópria, erros no programa, ou implementação inadequada, além da considerável economia de tempo de projeto.

As páginas de memória estão implementadas na forma descrita abaixo (ver figura 3.11):

2 páginas de memória RAM 0 e RAM 1 com 32 Kbytes cada [TRAINA, 1986]. Estas memórias foram implementadas com os circuitos 62256, que fornecem os 32 K em uma pastilha. Estes

circuitos são memórias estáticas e com o seu uso simplificou-se o projeto.

1 página ROM acessada nos endereços de 0 a 32 K. Esta página ROM é subdividida em [TRAINA, 1986] [MACHADO TRAINA, 1987]:

ROM p(8 K) - foi usado um CI 2764 e nesta memória ficam as rotinas do monitor de acesso a disco e impressora; as rotinas para comunicação com o terminal de vídeo; e foram incluídas as rotinas de votação parcial; as de comunicação no Anel; e as rotinas de gerenciamento do Sistema Tolerante a Falhas.

ROM e - implementados 2 K (com o C.I. 6116); na verdade trata-se de uma memória de escrita e leitura, tendo este espaço sido reservado na página ROM, porque nela ficam armazenados os dados necessários para a execução das funções do bloco ROM - p. Este espaço fica localizado a partir do endereço 7000H. O espaço localizado entre 2000H e 6 FFFH fica reservado para expansões futuras.

O uso deste esquema de chaveamento de memória se processa da seguinte forma [TRAINA, 1986]:

1. O sistema é inicializado executando uma rotina de inicialização da página ROM. Esta rotina coloca na 2ª metade do espaço de endereçamento a página RAM 1 e carrega nos endereços mais altos o BIOS do CP/M.

Fig. 3.11 : Esquema de Memória do subsistema de processamento.

2. A seguir o controle passa para a rotina de inicialização CBOOT do BIOS.

3. A rotina CBOOT do BIOS carrega o restante do Sistema CP/M, tal como BDOS e CCP [HOGAN, 1983]: CCP → Processador de Comandos de Controle; BDOS → Sistema Operacional Básico em Disco.

4. Completado o carregamento, esta rotina desativa a página ROM e coloca em seu lugar a página RAM 0. O controle passa ao CCP e o CP/M fica ativo.

5. O BIOS foi escrito [TRAINA, 1986] de forma que todas as suas funções são executadas através da ativação da página ROM. Sempre que for solicitada uma operação de leitura ou escrita em disco, a rotina correspondente do BIOS reativa a página ROM e passa o controle à rotina correspondente desta página.

Este esquema, em que as operações de Entrada e Saída são feitas através de ativação da página ROM, permitiu que o Sistema de Gerenciamento de Anel e da votação ficasse residente em ROM. No item 3.5 serão descritos estes processos.

3.4.2. - O SUBSISTEMA DE ENTRADA E SAÍDA

O Subsistema de Entrada e Saída é constituído de circuitos (74LS245) de acoplamento com o barramento de E/S e de um sistema chamado "linha bastão" que controla o

acesso ao barramento do módulo.

A "linha bastão" pode ser usada para evitar o acesso de mais de um módulo, ao mesmo tempo, a um mesmo periférico, em uma operação de leitura.

O ganho de acesso é feito através de seleção por programação. Cada processador gera um número aleatório e vai decrementando este número. Aquele que chegar primeiro no zero, ganha o acesso ao barramento. Este módulo sinaliza a "linha bastão", com nível lógico baixo, e até que ele retire o sinal, os outros módulos não podem acessar o periférico.

O esquema usado para a linha bastão está mostrado na figura 3.12. Nesta figura, o diodo usado em (*), evita que a fonte de alimentação de um módulo, jogue corrente na fonte de outro módulo.

Cada módulo lê a linha bastão antes de abaixá-la para uso, isto é feito através do controle da linha bastão. Se neste instante um outro módulo, que tinha gerado o mesmo número aleatório, ou um valor próximo, abaixar também a linha bastão, o circuito detector de coincidência sinaliza esta ocorrência, e cada módulo reinicializa o processo de acesso ao barramento, gerando outro número aleatório. Se todo o processo correu bem e não há conflito, o módulo habilita os circuitos de acoplamento com o barramento e processa a operação de leitura de periféricos. A partir daí cada módulo continua seu processamento normal, e assim que tiver que dar saída a um dado, este dado é enviado aos outros módulos, submetido a votação e apenas um dos módulos dá saída ao

dado.

Este sistema, usando uma linha bastão para acesso do módulo do Anel, é diferente do sistema de permissão "token", usado em redes de sistemas distribuídos com topologia em anel [TAROUÇO, 1985]. Nestes sistemas distribuídos, cada estação tem que esperar a permissão chegar até ela, para então começar a transmitir. Esta permissão circula de estação em estação. No sistema implementado, a linha bastão é utilizada para evitar conflitos no acesso ao barramento na entrada de dados, e não no canal de comunicação entre os módulos. Estes são procedimentos distintos. O acesso ao canal de comunicação será tratado mais adiante. Neste sistema a linha bastão está disponível a qualquer módulo, e acessa o barramento de entrada/saída quem sinalizar a linha bastão primeiro. Após o término da leitura, o módulo deixa o bastão disponível para os outros módulos que estão disputando o acesso. Caso um dos módulos tenha travado a linha bastão devido a uma falha transiente, o processo de reinicialização pode permitir recuperação do módulo.

3.4.3 - O SUBSISTEMA DE COMUNICAÇÃO NO ANEL

O Subsistema de Comunicação de dados no Anel envolve dois esquemas: a transmissão de dados para a votação e a transmissão dos resultados da votação, através

DC - DETETOR DE COINCIDÊNCIA
 $\overline{CBS}$ - CONTROLE DO BASTÃO PRÓPRIO
 $\overline{CBA}$ - CONTROLE DO BASTÃO MÓDULO ANTERIOR
 $\overline{CBP}$ - CONTROLE DO BASTÃO MÓDULO POSTERIOR
 (*) - É LIGADO NOS OUTROS MÓDULOS EM UM PONTO IGUAL A ESTE

Fig. 3.12 : Esquema da "LINHA BASTÃO".

dos sinalizadores de erro ("flags" de erro). A figura 3.13 ilustra os esquemas de transmissão no Anel. Ela será discutida com detalhes mais adiante. Este subsistema foi projetado com a circulação dos dados unidirecional no sentido horário.

Quando cada módulo tem um dado pronto para ser enviado à saída, antes que isto ocorra, ele é submetido a votação para a detecção de falhas nos módulos ou processos. No esquema de votação parcial, escolhido para este sistema com 3 módulos, há a necessidade de comparar os dados de 2 módulos adjacentes. Desta forma, o módulo 1 envia o seu dado para o módulo 2. O módulo 2 por sua vez, envia o seu dado para o módulo 3 e este para o módulo 1. Quando os três módulos estiverem com o dado recebido do módulo adjacente, cada um compara (vota) o resultado do seu próprio processamento com o dado recebido, e ativa o sinalizador de erro. Esta comparação deve ser feita entre dados de um mesmo processamento, para garantir a comparação de dados iguais, quando tudo está funcionando corretamente. A votação também deve ser feita ao mesmo tempo nos 3 módulos, para assegurar que todos os sinalizadores de erro, são resultados de um mesmo processo de votação. Em outras palavras, os 3 módulos devem operar sincronamente. Há várias formas de se alcançar a sincronização neste caso de Redundância com três módulos. Uma revisão é apresentada no capítulo 4. De uma maneira geral as formas mais comuns são: o uso de um único relógio externo; o uso de relógios independentes, sincronizados através de realimentação mútua (com o uso de algoritmos ou circuitos); e a sincronização de tarefas do processamento,

através de um ponto comum de encontro ("rendez vous"), sem referência a relógios. Para o Anel foi escolhida a sincronização através do ponto comum de encontro. O ponto de encontro, é onde os processadores aprontam os dados para serem votados e enviados à saída do sistema. Desta forma todos entram no ciclo de votação sincronizados. Para se obter esta sincronização, foi usado o esquema de transmissão com reconhecimento ("handshake").

3.4.3.a - TRANSMISSÃO COM RECONHECIMENTO ("HANDSHAKE")

Este processo consiste na troca de sinais que sincronizam a troca de dados entre um computador e um periférico, ou entre dois computadores. Aqui ele é usado para sincronizar a troca de dados entre os módulos e sincronizar o processo de votação. Ele também é usado na troca dos sinalizadores de erro.

Neste esquema, cada módulo do anel opera como um transmissor, quando envia o dado e como receptor, quando recebe o dado de outro módulo.

Quando o dado está pronto para a comparação, o módulo transmissor coloca os dados na saída do subsistema de comunicação e ativa a linha de handshake, informando ao outro módulo que os dados na saída do subsistema são válidos. O receptor mostra que recebeu os dados, enviando um sinal de reconhecimento. Quando termina a transmissão, o módulo transforma-se em um receptor. O intercâmbio de dados se processa nos seguintes passos:

1. o módulo transmite o dado;

2. recebe o reconhecimento do receptor;
3. recebe o dado do módulo adjacente;
4. envia o sinal de reconhecimento da recepção.

A sequência dos passos não precisa ser rigorosamente esta. Um módulo pode ser um receptor e depois transmissor e pode começar como receptor passar a transmissor e só depois completar a recepção.

3.4.3.b - TRANSMISSÃO DOS DADOS

Para implementar o processo de Transmissão com Reconhecimento, foi usado o circuito integrado 8255 da INTEL [INTEL, 1981]. Trata-se de uma interface de comunicação paralela programável. Ela possui três portos paralelos A, B e C de 8 bits cada, foi programada para operar no modo 1. Neste modo, os portos A e B operam como entrada ou saída e 4 linhas do porto C são usadas como linhas de controle para o processo de transmissão com reconhecimento.

Na figura 3.13.a está mostrado o esquema de comunicação no anel. O porto A foi programado como entrada e o porto B como saída. As linhas de saída do porto B estão conectadas às do porto A, seguindo a circulação do anel. Quando um dado em cada módulo vai ser submetido à votação, este é enviado para o porto B. Neste instante, o sinal de controle $\overline{OBF}$ ("output buffer full") do porto C vai para um nível lógico baixo, indicando que há um dado válido para saída no porto. Este sinal $\overline{OBF}$ está conectado

ao pino $\overline{STB}$ ("strobe input") do módulo adjacente no sentido da circulação. Um nível baixo neste pino carrega o dado no porto A do módulo adjacente. Neste instante o sinal IBF ("Input Buffer Full") do módulo receptor, vai para um nível lógico alto, sinalizando que o dado foi carregado no porto de entrada. Em outras palavras é a resposta de reconhecimento da recepção. Este sinal fica ativado, (nível alto), pelo sinal $\overline{SBT}$, e vai para o nível lógico baixo, quando o processador lê o sinal do porto, através da borda de subida do sinal $\overline{RD}$ (sinal de leitura), por isso o IBF é conectado ao sinal de entrada $\overline{ACK}$ ("Acknowledge") do módulo transmissor. Um sinal com nível lógico baixo na entrada $\overline{ACK}$, avisa o transmissor que o dado transmitido foi aceito.

Neste modo, os dados ficam travados ("latched"), tanto na saída do porto quanto na entrada, durante o processo de transmissão. Depois que cada módulo recebeu o sinal de reconhecimento ($\overline{ACK}$), e enviou o seu sinal (IBF), eles passam para rotina de votação. O status da transmissão é obtido através de leitura dos sinais de "handshake" no porto C.

3.4.3.c - TRANSMISSÃO DOS SINALIZADORES DE ERRO

A votação é feita por comparação e a implementação da rotina é discutida no item 3.5. Após a comparação cada módulo ativa o sinalizador de erro (EF), conforme o resultado. A configuração dos sinalizadores de

a) Transmissão dos dados para votação. (*)

b) Transmissão dos sinalizadores de erro. (*)

Fig. 3.13: Esquema de transmissão no subsistema de comunicação do anel.
(*) Legendas na próxima página.

LEGENDA DA FIGURA 3.13

- $\overline{DBF}$ - INDICA QUE HÁ UM DADO PRONTO NO PORTO.
- $\overline{ACK}$ - INDICA QUE O DADO FOI ACEITO.
- $\overline{STB}$ - CARREGA O DADO NO PORTO DE ENTRADA.
- $\overline{TBF}$ - INDICA QUE O DADO FOI CARREGADO NA ENTRADA.
- EFP - SINALIZADOR DE ERRO GERADO PELO MÓDULO.
- EFN - SINALIZADOR DE ERRO RECEBIDO DO PRÓXIMO MÓDULO.
- EFA - SINALIZADOR DE ERRO RECEBIDO DO MÓDULO ANTERIOR
- $\overline{SFN}$ - INDICA PARA O PRÓXIMO MÓDULO QUE O SINALIZADOR DE ERRO FOI ATIVADO
- $\overline{SFA}$ - INDICA PARA O MÓDULO ANTERIOR QUE O SINALIZADOR DE ERRO FOI ATIVADO.
- $\overline{ACKN}$ - INDICA QUE O SINALIZADOR DE ERRO DO PRÓXIMO MÓDULO FOI ACEITO.
- $\overline{ACKA}$ - INDICA QUE O SINALIZADOR DE ERRO DO MÓDULO ANTERIOR FOI ACEITO.
- $\overline{RACKN}$ - RECEPÇÃO DO RECONHECIMENTO DO PRÓXIMO MÓDULO
- $\overline{RACKA}$ - RECEPÇÃO DO RECONHECIMENTO DO MÓDULO ANTERIOR
- $\overline{RFA}$ - RECEPÇÃO DO AVISO QUE SETOU O FLAG ANTERIOR
- $\overline{RFN}$ - RECEPÇÃO DO AVISO QUE O FLAG DO PRÓXIMO MÓDULO FOI SETADO.

TABELA 3.1. - TABELA PARA SELEÇÃO DO MÓDULO DE SAÍDA

	EFA	EFB	EFC	Módulos de saída
0	0	0	0	Nenhum
1	0	0	1	MC
2	0	1	0	MB
3	0	1	1	MB
4	1	0	0	MA
5	1	0	1	MC
6	1	1	0	MA
7	1	1	1	MA
Tabela obtida considerando circulação horária				
EF = 0 → dados não concordam				

erro dos 3 módulos é que permite selecionar qual módulo deve alimentar a saída do sistema.

A seleção do módulo de saída pode ser feita por circuitos [OBAC RODA, 1983.a] ou por programação. Neste projeto a seleção é feita por programação, dentro de cada módulo. Para isto, é necessário que o módulo conheça os sinalizadores de erro dos outros. De posse dos sinalizadores, cada módulo acessa o barramento de E/S, se é o modo escolhido, ou deixa o acesso livre para outro módulo. A tabela 3.1 mostra as configurações possíveis. Esta tabela foi construída, considerando que o módulo que alimenta a saída é aquele cujo resultado é correto depois da comparação. Considerando o sentido da circulação.

Para a transmissão dos sinalizadores de erro, foi usada mais uma interface programável 8255. Neste caso a interface foi programada, com o porto A como saída e o porto B como entrada. Neste modo, cada porto A, B ou C pode ser programado como entrada ou saída, e o porto C pode ser dividido em 2 grupos de 4 bits, programados independentemente como entrada ou saída. Os dados nos portos de saída ficam travados ("latched"), mas não nos portos de entrada, sendo constantemente atualizados.

A implementação desta transmissão foi feita também usando um esquema de transmissão com reconhecimento, mas sob controle do processador e não "automático" como no caso da transmissão de dados para votação. Isto foi feito para aproveitar um único circuito, pois cada módulo deve enviar e receber os sinalizadores dos outros dois módulos. O esquema de transmissão está

mostrado na figura 3.13.b. Cada sinalizador de erro (EFP) é constituído de 2 bits. Quando há concordância de dados processados, a palavra enviada é 01H, e quando há discordância a palavra é 10H. Isto evita que falhas nos circuitos, indiquem uma palavra errada, para o resultado da votação. É mais provável, que uma falha faça os bits ficarem travados em apenas um nível lógico 00H ou 11H.

Os portos em cada módulo têm a seguinte configuração:

PORTO A: Saída

PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
X	X	$\overline{\text{ACK}} \text{ A}$	$\overline{\text{ACK}} \text{ N}$	$\overline{\text{SFA}}$	$\overline{\text{SFN}}$	EFP 1	EFP 0
		0	0	0	0	*	*

* → depende do resultado de votação.

Os sinais $\overline{\text{SFX}}$ indicam que os sinalizadores de erro foram ativados, com A para módulo anterior e N para o próximo módulo. Os sinais $\overline{\text{ACKX}}$ indicam, que o módulo recebeu o sinalizador do módulo anterior (A), e do próximo módulo (N) no anel. A legenda da figura 3.13, define cada um dos sinais.

PORTO B: Entrada

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
$\overline{\text{RACKA}}$	$\overline{\text{RACKN}}$	$\overline{\text{RF}} \text{ A}$	$\overline{\text{RF}} \text{ N}$	EFA 1	EFA 0	EFN 1	EFN 0
0	0	0	0	*	*	*	*

Nos sinais $\overline{RF}$, chegam os avisos de que os sinalizadores de erros foram ativados. Nos sinais $\overline{RACK}$ chegam o reconhecimento do recebimento dos sinalizadores enviados, tanto pelo módulo anterior, como pelo módulo posterior ao módulo objeto no anel.

Durante o processo de troca dos sinalizadores de erro, cada módulo fica testando as condições dos sinais de controle, tanto no porto de entrada como no de saída. Somente quando todos os sinais de controle estão no nível lógico baixo, em ambos os portos, é que os módulos vão consultar a tabela de seleção para saída.

Após o término deste processo, é enviada a palavra FCH para os portos de saída, para garantir que todos os bits dos sinais de controle fiquem em nível lógico alto, e os bits dos sinais de erro sejam zerados.

3.4.3.d. - CONFIGURAÇÃO DO CONECTOR DOS MÓDULOS

A configuração do conector dos módulos foi projetada de modo que qualquer módulo possa ocupar qualquer posição no anel. Cada módulo lê a sua posição no Anel, através do conector e desta forma pode se identificar como módulo A, B ou C, para a tomada de decisão de quem alimenta a saída do sistema, após a votação.

O conector também permite que o sistema opere sem que um dos módulos esteja conectado. A ausência de um módulo, é "sentida" pelos outros, como se os dados deste

módulo enviados ao anel e os sinalizadores de erro estivessem travados em um valor fixo. Para os outros 2 módulos, é como se houvesse um módulo falho no anel. O sistema como um todo continua operando normalmente. Isto facilita o reparo do sistema.

3.5 - PROGRAMAS DE GERENCIAMENTO DO SISTEMA.

O sistema de computação tolerante a falhas, com arquitetura em anel, apresentado nos parágrafos anteriores não estaria completo e nem poderia funcionar, sem um conjunto de programas de gerenciamento (software) adequado ao mesmo. Nos próximos parágrafos, as principais rotinas de gerenciamento dos subsistemas de processamento, entrada e saída e de comunicação no anel serão descritas e analisadas.

Inicialmente, antes de se tratar especificamente dos programas do sistema, é necessário definir a forma como estes programas serão armazenados no dispositivo secundário de armazenamento, ou como é estruturado o sistema de disco flexível desta arquitetura.

3.5.1 - SISTEMA DE ARMAZENAGEM EM DISCO FLEXÍVEL.

O sistema implementado, como já foi visto anteriormente, é composto de 3 módulos independentes, onde cada módulo compreende um subsistema de processamento, um

subsistema de entrada/saída e um subsistema de comunicação. Desta forma, a interface do disco flexível pode ser implementada de três formas distintas:

- a- Um disco único para os três módulos, com setores comuns aos 3 módulos.
- b- Um disco único para os três módulos, com setores independentes por módulo.
- c- Um disco para cada módulo.

As vantagens e desvantagens deste sistemas serão discutidas nos próximos parágrafos.

a - Disco único para os três módulos.

Neste caso os três módulos compartilham o acesso a um único disco flexível, que é formatado sem triplicação dos setores. Desta forma, por exemplo, quando o sistema é carregado, cada processador faz a leitura do primeiro setor do disco, e continua lendo os próximos setores, como se o disco fosse de seu uso exclusivo. A diferença é que na leitura de cada setor, o processador entra em uma fila de espera, composta pelos outros dois processadores, que também estão tentando ler o mesmo setor.

O gerenciamento do processo, onde é definido qual processador vai acessar primeiro o setor, é feito à nível de circuitos, pelo subsistema de entrada e saída através da "linha bastão". Este processo é enquadrado como um processo comum de entrada e saída, cujo mecanismo já

foi descrito no parágrafo 3.4.2. No caso de escrita no disco, cada módulo faz a verificação da soma ("Check Sum") dos bytes do setor à transferir, e o resultado é submetido à votação, sendo que somente o módulo vencedor faz a escrita do setor no disco.

Uma característica importante do acesso ao disco único, é o fato da controladora de disco ser considerada como parte do sistema de disco flexível, e não do processador. Desta forma, quando um módulo ganha acesso ao disco, na realidade ele ganha acesso à controladora, e sendo assim o subsistema de entrada e saída passa o sinal de relógio ("Clock") do processador vencedor, dá um reset no porto paralelo da controladora (8255), e um reset no integrado controlador de disco (2793). Esta característica tem a desvantagem de requerer um BIOS para os processadores, mais elaborado que o utilizado no micro LIE (2.34). No BIOS do micro LIE, o processador passa as palavras de comando ao circuito controlador de disco, ao longo de um conjunto de subrotinas dispersas através do BIOS. Quando é necessário fazer a leitura/escrita de um setor físico, estas palavras de comando, bem como as de identificação da trilha/setor já foram passadas à muito. Esta forma de abordagem não é conveniente no caso da arquitetura em anel, uma vez que quando um módulo ganha o acesso ao disco, para escrita de uma trilha, ele deve passar também seu sinal de relógio ("Clock"). Isto implica que o circuito integrado controlador de disco e o porto paralelo devem ser resetados. Desta forma as palavras de comando anteriores ficam perdidas. Portanto um BIOS para poder funcionar com estes módulos deve ter todas

as operações de entrada/saída em disco concentradas em uma única rotina. Na configuração atual, o BIOS do micro LIE dificilmente pode ser aproveitado para o gerenciamento dos módulos, quando se pretende operá-los compartilhando um único sistema de disco flexível.

Com exceção do problema do BIOS, o programa de gerenciamento deste processo é bastante simples e apenas implementa o procedimento já descrito em 3.4.2.

Nesta implementação as seguintes vantagens podem ser apontadas:

- 1- O usuário não tem que se preocupar em trabalhar com triplicata dos disquetes, como ocorre com o caso c.
- 2- O espaço disponível em disco é três vezes maior que no caso b.
- 3- O custo do sistema é menor pois requer apenas uma controladora e um acionador de disco.

Entretanto as seguintes desvantagens podem ser apontadas:

- 1- A confiabilidade do sistema fica restrita à do sistema de disco flexível, que é via de regra menor que a dos circuitos que compõe cada módulo, comprometendo a confiabilidade de todo o sistema. Neste caso o sistema de

disco está sujeito à três pontos de falhas: na controladora, no acionador de disco e no meio magnético do disquete. Desta forma o sistema de disco se constitui em um ponto vulnerável do sistema ("Hardcore"). Basta que haja apenas uma falha de leitura/escrita de um setor para levar à quebra do sistema ("Crash").

2- Impede o aproveitamento do BIOS do micro LIE, pois requer um BIOS especial para tratar a comutação dos módulos na controladora de disco .

b- Um disco único para os três módulos, com setores independentes por módulo.

Esta configuração é semelhante à anterior, apenas com a diferença que os setores pertencentes à cada módulo estão fisicamente separados no disco flexível. Os processadores também, como no caso anterior, entram em uma fila de espera. Entretanto esta fila é apenas para acesso à controladora e ao acionador, pois os setores são diferentes para cada módulo. Desta forma teremos uma trilha (0, 1, ... ,n) lógica para cada módulo, e assim sucessivamente, em um mesmo disquete.

O gerenciamento do sistema, tanto a nível de circuitos, quanto a nível de programas é semelhante ao caso anterior (a). A diferença é que cada módulo faz a verificação da soma ("Check Sum") dos bytes do setor à

transferir, e o resultado é submetido à votação. Os módulos cujo resultado concordam, fazem a escrita do setor no disco. Caso seja detectada uma falha em um módulo, é gerada uma mensagem no console e o módulo falho pode receber de um dos outros o setor correto a transferir. Os mesmos problemas da implementação anterior, quanto à passagem dos sinais de relógio do processador ("Clock") e da implementação do BIOS, também ocorrem aqui. Desta forma, as seguintes vantagens podem ser apontadas:

- 1- O usuário não tem que se preocupar em trabalhar com triplicata dos disquetes, como ocorre com o caso c.
- 2- Aumenta a confiabilidade do sistema de disco flexível, pois uma vez que os setores são triplicados, a ocorrência de uma falha no meio magnético de um setor físico não quebra o sistema, como ocorre no caso (a).
- 3- O custo do sistema é menor pois requer apenas uma controladora e um acionador de disco, como no caso (a).

As seguintes desvantagens podem ser apontadas:

- 1- Assim como no caso (a), a confiabilidade do sistema fica restrita à do sistema de disco flexível.
- 2- Também impede o aproveitamento do BIOS do micro LIE, pois requer um BIOS especial para tratar a comutação dos módulos na controladora de disco, como no caso (a).

3- O espaço de armazenamento em disco é um terço do caso (a).

c - Um disco para cada módulo.

Nesta configuração cada módulo conta com um sistema de disco flexível independente, constituído da placa controladora, acionador e meio magnético (disquete). Desta forma todo o sistema de disco flexível é parte do subsistema de entrada e saída do módulo em questão. Portanto uma falha no sistema de disco é considerado como uma falha no módulo.

O uso de um sistema disco para cada módulo apresenta as seguintes vantagens sobre os casos anteriores:

- 1- O sistema de disco flexível deixa de ser um ponto vulnerável do projeto ("Hardcore"), e qualquer falha neste é tratada como uma falha no módulo.
- 2- O BIOS do micro LIE pode ser aproveitado, exigindo apenas um grau tolerável de mudanças.
- 3- Simplifica o desenvolvimento dos programas de gerenciamento (Software) de todo o sistema tolerante a falhas, pois os programas de gerenciamento do sistema de disco são os mesmos do micro LIE.

Em contrapartida o sistema apresenta as seguintes desvantagens:

1- O usuário tem que se preocupar em trabalhar com os discos em triplicata (apenas as inserções e retiradas dos discos nos acionadores, pois uma vez criados os discos, em todas as operações posteriores com a máquina os três disco são automaticamente atualizados).

2- O custo do sistema é maior que nos outros casos.

Considerando as vantagens e desvantagens de todos os casos, a implementação final da máquina foi feita utilizando um sistema de disco flexível para cada módulo (caso c).

3.5.2 - DESCRIÇÃO DOS PROGRAMAS DE GERENCIAMENTO.

Uma vez definida a implementação com os três sistemas de disco flexível independentes, buscou-se tirar o máximo proveito desta configuração, pois a mesma permite a implementação do sistema de gerenciamento com o mínimo de alterações no BIOS desenvolvido para o micro LIE. As alterações efetuadas são divididas em dois grupos gerais: aquelas relacionadas com as diferenças de arquiteturas

entre o micro LIE e o Sistema em Anel; e as associadas à implantação dos programas de gerenciamento da tolerância a falhas.

As alterações devidas às diferenças de arquitetura entre os computadores, são as mesmas que se fazem necessárias quando se implementa uma nova máquina CP/M comum (diferenças de endereçamento nas portas de entrada/saída, em memórias RAM's, em ROM's e formas de acesso à disco). Este processo, embora trabalhoso, não apresenta detalhes relevantes à implementação dos programas de gerenciamento do Sistema em Anel. Entretanto é conveniente salientar que tais diferenças entre as arquiteturas do micro LIE e do Sistema em Anel são inerentes e fundamentais às características de tolerância a falhas deste último, e portanto este tipo de alteração no BIOS é inevitável.

O segundo tipo de alteração, é o que realmente concerne aos programas de gerenciamento do Sistema em Anel. Mesmo neste caso, a filosofia seguida foi a de fazer o mínimo possível de alterações no BIOS original do micro LIE, sem entretanto comprometer o desempenho do Sistema em Anel. Tendo esta filosofia como meta, os programas de gerenciamento devem tratar com os seguintes casos:

- a- Entrada de dados do teclado.
- b- Saída de dados para o vídeo, ou outro dispositivo de saída.
- c- Entrada de dados de outro dispositivo.

Como pode ser observado, todos os casos acima envolvem operações de entrada/saída no BIOS do sistema. A solução encontrada para manter a meta de alteração mínima no BIOS, foi a de substituir cada um destes pontos de entrada/saída do BIOS, por uma chamada da rotina correspondente nos programas de gerenciamento do Sistema em Anel. Dentro desta filosofia, cada um destes casos será analisado separadamente nos parágrafos seguintes.

a-Entrada de dados pelo teclado.

Este é o caso mais simples de todos, uma vez que o Sistema em Anel faz votação apenas na saída dos dados. Desta forma, os dados de entrada não necessitam de votação. Cada processador tem seu porto serial de entrada/saída, e a saída serial, do terminal de vídeo (no padrão RS-232) alimenta um conversor RS-232/TTL (1489), que é ligado em paralelo na entrada de cada porto serial dos módulos. Portanto o acionamento de qualquer tecla no teclado, é passado aos três processadores. Neste caso é utilizada a capacidade intrínseca de sincronização do porto serial, e a própria rotina de entrada de caracteres do BIOS, de cada processador, pode ser empregada.

b-Saída de dados para o vídeo, ou outro dispositivo de saída.

Como já foi visto anteriormente, o Sistema em

Anel faz a votação sobre os dados de saída, portanto é exatamente neste caso que os programas de gerenciamento do sistema exercem sua ação principal. Procurando manter a filosofia inicial de realizar o mínimo de modificações no BIOS, na rotina de saída de dados do BIOS é feita uma chamada para o principal programa de gerenciamento do Sistema: a Rotina de Saída de Dados do Sistema em Anel (RSD), cujo fluxograma esta esquematizado na fig.3.14.

Nesta figura, está exemplificada a chamada da Rotina de Saída dos Dados do Sistema em Anel, à partir da rotina CNDOUT. A função da rotina CNDOUT do BIOS é dar a saída dos dados do Monitor (e portanto do CP/M também) no vídeo.

A RSD após a inicialização dos portos de comunicação do anel e dos portos de transmissão dos sinalizadores de erro, chama a Rotina de Votação e Sincronização (RVS). No retorno da RVS a rotina RSD recebe a informação do módulo que venceu a votação e somente para este módulo ela habilita o envio de dados para o vídeo.

A Rotina de Votação e Sincronização esta esquematizada na fig. 3.15, e tem por função realizar a sincronização e votação entre os módulos. a RVS chama a Rotina de Transmissão dos Dados para Votação (RTD), responsável pela transmissão dos dados nos portos e, devido ao sistema de transmissão com reconhecimento, pela sincronização dos módulos. O fluxograma da RTD está apresentado na fig.3.16. A rotina RVS também coordena a chamada da Rotina de Transmissão dos Sinalizadores de Erros (RTS), segundo a concordância ou não entre os

dados do módulo atual e o módulo anterior . A RVS decide qual foi o módulo vencedor, e aciona os sinalizadores ("flags") correspondentes, consultando a tabela de decisão fornecida pela Rotina " Monta a Tabela de Decisão " (MTD). O fluxograma da RTS está mostrado na fig. 3.17 , e o da MTD na fig. 3.18 . Este conjunto de rotinas completa o sistema de gerenciamento na parte referente à votação e sincronização dos dados de saída.

c-Entrada de dados de outro dispositivo.

As características da arquitetura do Sistema em Anel, principalmente o sistema de linha bastão, permite que os módulos compartilhem um dispositivo único de entrada de dados. Tal dispositivo poderia ser por exemplo um conversor A/D, para aplicações em sistemas de controle. Neste caso o esquema de funcionamento do sistema com a linha bastão, para gerenciar o acesso ao dispositivo e evitar o conflito dos módulos é semelhante ao descrito no parágrafo 3.5.1-a, quando foi tratado o caso de compartilhamento de um sistema de disco flexível único, com os três módulos. E o procedimento do programa de gerenciamento para este caso já foi descrito no parágrafo 3.4.2.

RSD

FIG. 3.14 - ROTINA DE SAIDA DOS DADOS DO SISTEMA EM ANEL. (RSD).

FIG. 3.15 - ROTINA DE VOTAÇÃO E SINCRONIZAÇÃO. (RVS)

FIG. 3.16 - ROTINA DE TRANSMISSAO DE DADOS

FIG. 3.18 - ROTINA MONTA TABELA DE DECISAO

CAPÍTULO 4

SINCRONIZAÇÃO EM SISTEMAS TOLERANTES A FALHAS COM VOTAÇÃO

4.1 INTRODUÇÃO

A sincronização entre os módulos, de uma arquitetura NMR tolerante a falhas, é o ponto vital para o funcionamento destas estruturas. O problema de sincronização nestes casos, apresenta dois aspectos. Um é o problema de sincronizar os elementos da arquitetura, de maneira que estes elementos executem seus processamentos com alguma simultaneidade. O outro aspecto, é utilizar os métodos desenvolvidos no primeiro, para construir relógios tolerantes a falhas. Os relógios seriam por sua vez, usados para sincronizar as arquiteturas NMR.

O objetivo deste capítulo é apresentar as soluções existentes, para sincronizar os módulos das arquiteturas NMR com votação. O uso de relógios tolerantes a falhas, constitui uma das soluções apresentadas.

Esta revisão, permite avaliar o método de sincronização usado na Arquitetura em Anel com 3 módulos.

4.2 SINCRONIZAÇÃO

Nos sistemas de computação, que usam a redundância modular com votação, como técnica de tolerância a falhas, é essencial que haja sincronização na operação dos módulos. A idéia central de sincronização para estas arquiteturas, é a realização do processo de votação sobre a mesma versão de dados, [WENSLEY, 1983.a] [FRISON, 1982]. Votar sobre a mesma versão de dados, significa que os módulos/processadores precisam realizar suas tarefas, dentro de aproximadamente, o mesmo intervalo de tempo. Logo, a aquisição de dados, processamento, votação, saída de dados, devem ser realizados em paralelo pelos módulos (pelo menos, pelos em bom funcionamento), seguindo o mesmo fluxo de processamento. Por outro lado, mesma versão de dados, não significa que os dados devam ser rigorosamente iguais. Embora isto seja normalmente verdade, aqui pode significar que os dados são "legitimamente diferentes" [OBAC RODA, 1982.b], como aqueles produzidos pela leitura de conversores analógicos/digitais. Neste caso a estratégia de votação deve ser adequada para tratar com estes dados, por exemplo tomando um valor médio [KLAASSEN, 1977].

Há duas formas gerais de se obter sincronização nestes sistemas de computação, que executam processos redundantes [BUTLER, 1985] :

1. Através de relógios , "clocks";
2. Através de ponto comum de encontro . ("rendez-vous").

A sincronização através de relógios, tem sido a forma mais utilizada nos sistemas tolerantes a falhas com votação. A palavra relógio tem sido usada na literatura sobre sincronização, tanto para designar um simples circuito oscilador, constituído por um oscilador a cristal e circuitos lógicos [DAVIES, 1978], como para definir uma função que ordena eventos [LAMPART, 1978] [BABAOĞLU, 1978]. Um evento pode ser uma parte de um programa, pode ser a execução de uma simples instrução de máquina [LAMPART, 1978], ou pode ser também uma rotina de aquisição de dados. Uma sequência de eventos constitui um processo. Como ordenador de eventos, o relógio pode ter um significado abstrato, como o relógio lógico ("logical clock") tratado por Lamport [LAMPART, 1978]. Neste caso, o relógio é um modo de atribuir um número a um evento, permitindo a ordenação destes eventos, sem se preocupar com o tempo físico. Ou pode ter um significado mais "palpável", onde o número atribuído ao evento é função do tempo físico, no qual o evento ocorreu [LAMPART, 1978]. Em outras palavras, com este significado, o relógio constitui uma referência de tempo para os eventos. É com este significado que a palavra relógio será usada, pois, para sincronizar eventos em tempo real, é necessário que o relógio seja função do tempo [BABAOĞLU, 1978]. Deste conceito, não se exclui os circuitos osciladores simples, que fornecem os sinais de temporização para outros circuitos, também neste caso eles fornecem uma referência de tempo. Por outro lado, o relógio pode ser considerado como um processador, capaz de operações lógicas e

constituído, na realidade, de um processador e seu relógio interno [WENSLEY, 1978].

O outro modo de sincronizar as arquiteturas NMR com votação, é através de um ponto comum de encontro, ou um ponto de "rendez-vous" [WENSLEY, 1983.a] [BUTLER, 1985]. Este modo pode ser usado nos sistemas, onde o projeto permite que os módulos executem seus processamentos, com diferentes velocidades. Neste caso, para que a votação seja realizada sobre a mesma versão de dados, é necessário que os módulos cheguem a um ponto comum de encontro, antes da votação. Após este encontro, todos devem executar a votação em paralelo. A maneira de garantir este encontro, é através da comunicação síncrona entre os módulos. Neste esquema, o módulo que chega primeiro ao ponto de encontro, avisa os outros, e fica esperando pela chegada dos demais. Somente quando todos chegaram ao ponto de encontro, é que tem início o processo de votação. O ponto de encontro, neste caso, é o ponto no processo, onde cada módulo apronta o dado para ser submetido a votação. Este modo de sincronizar as arquiteturas NMR com votação, é análogo ao mecanismo de "rendez-vous", usado na comunicação síncrona entre processos em sistemas distribuídos [KIRNER, 1988] [LAGES, 1986].

4.3 SINCRONIZAÇÃO COM RELÓGIOS

A sincronização de sistemas redundantes através de relógios, pode ser feita nas seguintes formas [DAVIES,

1978] :

1. Cada módulo do sistema NMR possui o seu próprio relógio, extremamente exato, sendo este independente dos outros. A sincronização é governada pela precisão destes relógios.
2. O sistema NMR possui um único relógio externo, que fornece uma base de tempo única, para todos os módulos do sistema. Os módulos mantêm a sincronização baseados na mesma referência de tempo. Este relógio externo, também pode ser construído usando a redundância modular para tolerar falhas, já que pode constituir um ponto vulnerável do sistema.
3. Através de Realimentação Mútua ("Mutual Feedback"), onde não há uma referência de tempo comum para os módulos. Cada módulo tem seu próprio relógio, e cada um monitora o relógio do outro, e corrige o seu no caso de haver diferença ("drift"). Este procedimento tanto pode ser implementado com programas [WENSLEY, 1978], como por circuitos [DAVIES, 1978].

4.3.1 RELÓGIOS INDEPENDENTES

O primeiro modo de sincronização, através de relógios independentes extremamente exatos é impraticável para circuitos reais. Relógios extremamente exatos são difíceis de construir, devido às diferenças intrínsecas de cada componente. Por menor que se fizesse a diferença de tempo ("drift") de um relógio para outro, o sistema NMR acabaria perdendo o sincronismo com o passar do tempo [DAVIES, 1978].

4.3.2 RELÓGIO EXTERNO

O uso de um relógio (referência de tempo) externo, comum a todos os módulos, tem sido um dos modos usados para sincronizar as arquiteturas NMR. A pesquisa neste campo, é voltada principalmente para a simulação e desenvolvimento de relógios tolerantes a falhas [DAVIES, 1978] [KESSELS, 1984] [VASANTHAVADA, 1986] [DALY, 1973].

Este modo, é também chamado sincronização forte ("tight synchronization") [HOPKINS, 1978]. Todos os módulos trabalham em passo, baseados na mesma referência de tempo. Ele tem vantagens e desvantagens, sendo adequado para sincronizar os sistemas NMR, que utilizam a votação por circuitos. A votação pode ser feita em tempo real, pois os módulos não têm que gastar tempo com rotina de

sincronização. Este tipo de sincronização permite a validação imediata dos dados internos. Contudo, o uso de um relógio comum, pode constituir um ponto vulnerável do sistema. Para as arquiteturas NMR tolerantes a falhas, os relógios também devem ser tolerantes a falhas. Outro problema que surge com o uso de um único relógio, é que o sistema fica mais sujeito a falhas correlatas, por exemplo, um efeito transiente pode afetar todos os processadores do mesmo modo.

A forma clássica de implementar o relógio tolerante a falhas é através de sistemas com relógio de fase amarrada ("phase-locked clock") [HOPKINS, 1978] [DALY, 1973] [KESSELS, 1984] [VASANTHAVADA, 1986]. Outra forma é construir este relógio, usando o método de realimentação mútua [DAVIES, 1978], que será descrita no próximo item.

Cada módulo do relógio tolerante a falhas, de fase amarrada, é composto por: - um oscilador para ajustar a frequência de operação de cada módulo individual [HOPKINS, 1978] [DALY, 1973] (mais recentemente um VCXO, - "voltage controlled crystal oscillator" - [VASANTHAVADA, 1986]); - filtros tipo RC; - detector de fase; - amplificadores e lógica digital para operações de decisão. Para implementar este sistema, com tolerância a falha em f módulos, são necessários no mínimo $3f + 1$ módulos, conforme demonstrado por Daly et al [DALY, 1973], para manter a fase amarrada em pelo menos $2f + 1$ módulos. Um sistema deste tipo, para tolerar falhas em um módulo, está mostrado na figura 4.1. Neste esquema, o receptor de

Fig. 4.1 : Relógio de fase AMARRADA TOLERANTE À FALHA.

Fig. 4.2 : Efeito da falha em um sistema com 3 relógios.

relógio (CR) é um circuito digital responsável pela sintetização de um sinal de temporização, baseado nos sinais de saída de todos os outros osciladores. São necessários $3f + 1$ módulos, pois todos os sinais não coincidem precisamente, embora cada módulo do sistema esteja com a fase amarrada aos demais. Se for usado, a decisão por maioria do tipo 2 em 3, a saída do sistema irá depender do elemento defeituoso, conforme está mostrado na figura 4.2.

Dependendo do tipo de falha, pode haver necessidade de um número de módulos maiores que $3f + 1$ [KRISHNA, 1985], para implementar um sistema de relógio tolerante a falhas. Aumentando o número de módulos, a complexidade de circuitos necessários para implementar o sistema, também aumenta. Assim sendo, nem sempre o uso de um relógio externo ao sistema, mesmo que este seja também tolerante a falhas, é a solução mais simples para a sincronização. Existem vários estudos neste sentido, [KESSELS, 1984] [VASANTHAVADA, 1986] [KRISHNA, 1985] [VASANTHAVADA, 1984] mas as soluções apresentadas são complexas.

Usando a técnica de realimentação mútua, basta $2f + 1$ módulos para implementar relógios tolerantes a falhas, desde que o sistema não sofra falhas maliciosas. Se a ocorrência desta falha for considerada, então há a necessidade de se usar mais um nível de $2f + 1$ módulos, acoplado ao primeiro. Esta situação, aumentaria a complexidade do sistema implementado.

O projeto FTMP - "A Highly Reliable Fault - Tolerant Multiprocessor for Aircraft" - [HOPKINS, 1978], usa um relógio externo para sincronizar o sistema. O FTMP é uma arquitetura NMR que faz a votação e sincronização por circuitos. O relógio externo é tolerante a falhas, e implementado usando o sistema de relógio com fase amarrada. O FTMP é constituído de um conjunto de módulos de processamento com memória local, um conjunto de módulos de memórias e módulos de entrada e saída. Um conjunto de três módulos de memória, três módulos de processamento e três módulos de E/S formam uma unidade de processamento (Fig. 4.3). Os módulos são conectados por canais seriais redundantes. Toda informação processada é conduzida em triplicata, e votadores locais em cada módulo mascaram as falhas, através de votação bit a bit. A interface de cada módulo, com o duto de dados, é controlada por dois Circuitos de Reconfiguração ("bus guardian"), que recebem comandos de reconfiguração através de 3 canais de dados. Estes geram os comandos para ligar ou desligar um módulo na triade, e para seleccionar um duto de dados a ser conectado à saída. O sistema todo opera sincronamente, recebendo os sinais de temporização de um relógio externo tolerante a falhas com fase amarrada. Cada módulo e circuito de Reconfiguração recebe os 4 sinais de relógio, fornecidos pelo relógio externo (Fig. 4.1). O sistema tem em todos os pontos disponíveis, 4 sinais de relógio. Cada receptor de relógio em cada módulo, tem 3 destes sinais para gerar um sinal de relógio para o módulo. Isto é feito para tolerar falhas na transmissão dos sinais de relógio.

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
 BI - INTERFACE COM O DUTO DE DADOS

Fig. 4.3: Arquitetura do FTMP.

Como todos os 3 sinais que chegam ao módulo, estão rigorosamente em fase, basta uma votação por maioria, para tolerar uma falha. Quando uma falha é detectada, o sistema reconfigura-se e toma o outro sinal de relógio dos 4 disponíveis.

4.3.3 REALIMENTAÇÃO MÚTUA

O conceito básico do processo de Realimentação Mútua, é a não existência de uma referência de tempo única, comum. Cada módulo monitora o relógio (referência de tempo) do outro e se corrige.

Neste caso, cada módulo tem o seu próprio relógio, para ordenar suas ações. Este relógio não precisa ser o relógio implementado por circuitos, usado para os sinais de temporização do processador e outros circuitos. É um relógio físico, no sentido que é função do tempo, mas tem o significado de ordenador de eventos no sistema. Para sistemas simples, ele pode ser o próprio relógio de temporização. A realimentação mútua pode ser implementada por circuitos ou programas.

4.3.3.a REALIMENTAÇÃO MÚTUA POR PROGRAMAÇÃO

A implementação desta forma de sincronização através de programação, se faz através de algoritmos,

baseados na informação a respeito dos relógios de cada módulo. Esta informação pode ser na forma de mensagens, que se constituem em leituras dos relógios individuais, e são trocadas periodicamente, para a sincronização do sistema [LAMPORT, 1985] [WENSLEY, 1978] [FRISON, 1982] [BUTLER, 1985]. As mensagens podem ser simples pulsos com largura determinada, ou a borda de subida ou descida de um sinal periódico. Podem também ser mensagens a respeito de algum evento comum a todos os módulos. Podem ser o valor de um registrador. Na referência [BABAOĞLU, 1987] é apresentado um algoritmo para sincronizar sistemas distribuídos, onde é usado o tempo gasto no envio de uma mensagem, como meio de sincronização do sistema. Esta troca de mensagens faz parte normalmente do sistema operacional distribuído, onde há troca de informação periódica, seguida de computação local, não representando custo adicional de programação. Um algoritmo similar a este é apresentado na referência [INFIS, 1988], para sincronizar sistemas tolerantes a falhas. A pesquisa em algoritmos para sincronização de sistemas, é voltada principalmente para sistemas distribuídos [BABAOĞLU, 1987] [LUNDELIUS, 1983] [CRISTIAN, 1986] [LAMPORT, 1985]. Contudo alguns algoritmos podem ser usados em arquitetura NMR tolerantes a falhas [INFIS, 1988]. O sistema "VOTRICS" ("Voting Triple - Modular Computing System") [WIRTHUMER, 1989] e [THEURETZBACHER, 1986], é uma arquitetura tolerante a falha, voltada para uso em computadores disponíveis comercialmente ("off the shelf"), onde a sincronização é implementada através de algoritmo, para

sincronizar os relógios das 3 unidades independentes.

A vantagem de se ter relógios independentes, está na ausência de um ponto comum de falhas. No caso de relógios, porém, a falha em um deles pode afetar os outros, de maneira que o sistema perde a sincronização. Os algoritmos desenvolvidos, visam também evitar que a falha afete os outros relógios [BUTLER, 1985]. Um algoritmo clássico, é o algoritmo da média, usando pelo menos 3 relógios. Cada módulo obtém o valor do relógio do outro e calcula a média, usando este valor para resincronização. Na presença de uma única falha, considera-se que o valor médio se aproxima do valor correto. Este algoritmo, apesar de extremamente simples, não resiste a falhas maliciosas. Se o processador falho, enviar diferentes valores de seu relógio para os módulos em bom funcionamento, o sistema perde o sincronismo. Usando o exemplo apresentado em [WENSLEY, 1978] é possível ilustrar isto. Um sistema possui três módulos A, B e C, com A um pouco adiantado em relação a B, sendo que o relógio de C não funciona corretamente. Os valores que A e B recebem de C são diferentes. O tipo de falha de C é tal que para A, o relógio C está adiantado. Para B, o relógio C está atrasado. Os dois módulos A e B observam corretamente que A está adiantado em relação a B. Com base nesta situação o módulo A, escolhe a sua própria leitura de relógio como valor médio para se resincronizar. Para B, também o seu próprio valor é o valor médio para resincronização. Com isto, A continua adiantado em relação a B e o sistema como um todo, perde a sincronização.

O problema crítico com a sincronização através de realimentação mútua, é evitar que a falha em um dos relógios, afete os outros em bom funcionamento. Para evitar que isto ocorra, geralmente são usados $n \geq 3f + 1$ módulos/relógios (f é o número de módulos/relógios falhos).

A referência [PEASE, 1980] prova que é impossível obter sincronização com $2f + 1$, se é permitido aos relógios falharem de modo arbitrário, como o exemplo acima, apresentando falhas maliciosas no fornecimento da leitura do relógio, aos outros módulos. Soluções para sincronização com $2f + 1$ módulos, envolve o uso de restrições sobre a natureza de falha [DE SOUZA, 1975] [INFIS, 1988], a necessidade de identificação do módulo/relógio falho [FRISON, 1982], ou o uso de mais um nível (de circuitos, ou aplicações do algoritmo) no processo de sincronização [DAVIES, 1978].

O sistema "SIFT" - "Software Implemented Fault Tolerance", usa relógios independentes para obter a sincronização através de um algoritmo [WENSLEY, 1978]. Trata-se de uma arquitetura tolerante a falhas, desenvolvida pela NASA, para uso em aeronaves. O sistema é formado por 5 módulos, cada unidade é composta por um processador e sua memória associada, um controlador de canal de comunicação e o processador de entrada e saída (Fig. 4.4). Os processadores de E/S são conectados aos sensores e atuadores da aeronave. O sistema executa tarefas de controle, de forma que a saída dos atuadores é gerada dentro de intervalos de tempos estabelecidos. Esta

Fig. 4.4 : Arquitetura do Sistema SIFT.

atuação é feita na forma de um conjunto de tarefas, que consistem numa sequência de iterações. Depois da execução de uma iteração, cada processador coloca o resultado em sua memória associada. No começo da próxima iteração, os processadores que precisam deste resultado, obtêm o valor a ser usado, lendo a memória dos outros processadores e fazendo a votação por maioria. Esta votação é feita por programação em cada um dos processadores, e sempre no início de uma iteração.

A sincronização entre os módulos, deve assegurar que os diferentes processadores, executem a mesma iteração, no mesmo período de tempo. Neste projeto, sendo a sincronização feita também por programação, é permitido aos módulos uma diferença de até 50 micro-segundos na execução das iterações. O algoritmo de sincronização é executado em duas partes [WENSLEY, 1978]. Primeiro cada processador calcula um "vetor de consistência" através de leitura dos relógios de todos os outros processadores. Cada elemento deste vetor corresponde a um relógio, sendo calculado da seguinte forma:

- 1- O módulo p obtém o valor do relógio do módulo q, diretamente. Na verdade, q lê o seu próprio relógio e transmite o seu valor a p. Isto é feito através de uma rede de difusão ("broadcast"), pois quando q transmite o seu relógio, ele o transmite a todos os módulos [BUTLER, 1985].

2- Obtém de cada módulo r , o valor que eles leram para o relógio do módulo q .

3- Se a maioria destes valores concordam, o módulo p coloca este valor como elemento do vetor de consistência. Caso contrário, ele coloca um valor a revelia, que indica falha no relógio de q ("default value").

4- O elemento do vetor correspondente ao próprio p é o seu próprio relógio.

Tendo obtido o "vetor de consistência", cada módulo ajusta o seu relógio. Para isto é tomado um valor ("skew") δ , determinado a priori. Este valor é a máxima diferença que pode existir entre dois elementos do vetor. Estes elementos correspondem aos relógios sem falhas, os valores "a revelia" são ignorados. Qualquer componente, que não está dentro de δ , quando calculada esta diferença em relação a pelo menos duas outras componentes, também é ignorada. O novo valor de relógio é ajustado tomando o valor médio das componentes restantes. O reajuste de cada relógio é feito periodicamente, para garantir que o sistema permaneça sincronizado.

Em resumo [BUTLER, 1985], o sistema de sincronização para o "SIFT" é composto pelos relógios independentes pertencentes a cada módulo; uma rede para difusão das leituras dos relógios aos outros módulos e pelo algoritmo que ajusta a sincronização.

4.3.3.b REALIMENTAÇÃO MÚTUA POR CIRCUITOS

Um modo de sincronização através de realimentação mútua, usando circuitos, é apresentado na referência [DAVIES, 1978]. Nesta referência é apresentado um modelo, chamado "sincronização votada" ("synchronization voting"), que usa $2f + 1$ módulos ou relógios, para tolerar f falhas. O modelo do sistema está mostrado na figura 4.5.a. Na figura 4.5.b está o modelo para o sincronizador. Este consiste basicamente de um elemento votador por maioria, seguido de um elemento para gerar um atraso. Cada módulo tem um sinal chamado "início" ("Go") e um chamado "pronto" ("Ready"). O sinal de "início" é uma entrada e indica ao módulo, que ele pode começar o ciclo de votação. O sinal "pronto" é uma saída e indica que o módulo terminou o ciclo de votação. O ciclo de votação é o intervalo de tempo entre dois sinais de "início". Quando a votação é aplicada a nível de transferência de registros em um processador, o ciclo de votação coincide com o ciclo de relógio ("clock cycle"). Se a votação é aplicada apenas sobre os sinais de saída de um módulo, então o ciclo de votação, é o intervalo de tempo entre o aparecimento de dois sinais de saída. Para sincronizar os módulos do sistema, cada sincronizador envia um sinal de "início", para o seu módulo associado, baseado na votação por maioria dos sinais de "pronto" de todos os módulos do sistema. Na figura 4.5, pode-se observar que há um sincronizador para cada módulo. Os módulos e sincronizadores são isolados dos outros, de

Fig. 4.5.a : Modelo do Sistema.

Fig.4.5.b : Modelo do Sincronizador.

Fig. 4. 5: Modelo de sincronização com realimentação mútua : Sincronização Votada.

maneira que a falha de um sincronizador, só afeta o módulo ao qual ele está associado. É considerado que uma falha no sincronizador é equivalente a uma falha no módulo correspondente. O atraso é introduzido, para permitir ao sistema, tolerar módulos mais lentos. Este atraso também permite que o sistema continue a operar, com f das saídas "pronto" inativas.

O tempo de atraso, deve ser maior que a máxima diferença no tempo de envio do sinal "pronto", pelos módulos mais lento e mais rápido. A fixação deste tempo de atraso, permite que a sincronização seja realizada apenas com $2f + 1$ sincronizadores, pois se evita situações como aquela mostrada na figura 4.2, onde o sinal falho interfere na saída, devido a diferença na chegada dos sinais dos relógios em bom funcionamento. Os autores demonstram matematicamente, que é possível com este modelo, manter a sincronização do sistema, mesmo na presença de f falhas, desde que a falha não seja maliciosa. Por exemplo, se um módulo falha, o sinal "pronto", não deve ser interpretado de maneira diferente pelos diferentes sincronizadores. Neste caso para tolerar f falhas, deste tipo, deve-se usar $f + 1$ níveis de sincronizadores, como mostrado na figura 4.6. Com o uso de $f + 1$ níveis de módulos, haverá ao menos uma coluna de módulos livre de falhas e o sistema continuará sincronizado. Isto pode representar uma desvantagem, pois o custo adicional de circuitos usados nos níveis para tolerar falhas deste tipo, pode ser maior do que o uso de $3f + 1$ módulos. Por outro lado, uma cuidadosa análise do

Fig. 4.6: Sistema utilizando níveis de sincronizadores.

sistema e da probabilidade da ocorrência destas falhas, pode indicar ser esta uma solução adequada para a sincronização, principalmente em sistemas TMR. Usando o modelo de realimentação mútua, através de sincronização votada, o artigo [DAVIES, 1978] traz o projeto de um relógio, que pode ser usado como relógio externo, para sincronizar outros sistemas. O circuito está mostrado na figura 4.7, e consiste de um oscilador a cristal de 8MHz, circuitos digitais ECL para o votador e amplificador, resistores, capacitores e um comparador de tensão. Em termos do modelo de sincronização votada, cada cristal e amplificador é um módulo, e cada votador é um sincronizador sem o atraso, e as entradas do votador são as saídas dos 3 amplificadores. Desta forma os três cristais são excitados com pulsos cuja frequência é a sua frequência média. O atraso não é necessário, porque o único dado que o "módulo" produz é o sinal de "pronto". Não é necessário esperar os dados ficarem prontos. Também um cristal mais lento, pode receber o sinal de "início" um pouco antes de enviar o sinal de "pronto". Um cristal pode ser excitado por uma frequência mais alta que a sua frequência natural. Como não há o atraso, e de forma a impedir que uma falha controle a saída do votador, a diferença entre as transições nas saídas dos amplificadores, em bom funcionamento, são feitas as menores possíveis. Na referência [MCCONNEL, 1981], também são apresentados vários esquemas para implementação da sincronização votada, usando diferentes tipos de sinalização, tais como pulsos, transição de estado e nível.

Fig. 4.7: Relógio tolerante à falha com sincronização votada.

de sinal.

Este modelo de sincronização votada, pode ser implementado por programação [DAVIES, 1978] [OBAC RODA, 1983.a]. Através de programação, cada módulo monitora o "status" da saída dos outros módulos, que indica o fim do ciclo de votação. Quando a maioria dos módulos sinaliza que completou o ciclo, cada módulo entra em um ciclo de espera ("delay loop"), esperando pelos módulos mais lentos. Este procedimento é adequado para sincronizar sistemas que executam tarefas repetitivas.

4.4 PONTO COMUM DE ENCONTRO

A sincronização através de ponto comum de encontro ("rendez vous"), não tem sido muito explorada em arquitetura NMR tolerantes a falhas. Este método é mais utilizado em sistemas distribuídos. Ele é baseado na comunicação síncrona entre os módulos (ou entre processos em sistemas distribuídos) [KIRNER, 1988]. Na comunicação síncrona, o módulo remetente somente prossegue com o seu processamento, quando o destinatário recebe a mensagem enviada. O destinatário só continua o processamento após receber a mensagem que estava esperando. Desta forma é estabelecido um diálogo entre os módulos, para que um se informe do estado do outro [LAGES, 1986]. Após a troca de mensagens, os módulos continuarão seu andamento em paralelo. Este método é portanto adequado, quando 2 (ou

mais) módulos devam executar alguma função ao mesmo tempo, como é o caso da votação de dados. Nas arquiteturas NMR, as mensagens trocadas podem ser simples sinalizadores ("flags") [CHANDE, 1989], que indicam o estado do sistema, por exemplo: "dado pronto para votação". Ou podem ser o próprio dado a ser votado, como foi implementado na Arquitetura em Anel.

Um sistema tolerante a falhas, que faz a sincronização através de ponto comum de encontro, é apresentado na referência [CHANDE, 1989], chamado "Modular TMR Multiprocessor Systems". O sistema é implementado com 3 módulos, utilizando o Z-80 como microprocessador. Cada módulo é um microcomputador independente, com sua própria memória local e circuitos de entrada/saída. O sistema está mostrado na figura 4.8, e foi projetado para controle de motores elétricos.

Os módulos comunicam um com o outro, através de uma memória global (GM) compartilhada, usada exclusivamente como meio de comunicação entre os processadores. O acesso a memória global, é feito através de um sistema ("bus arbiter") que controla a alocação desta memória. Cada microprocessador requisita o acesso, através deste controlador. Todos os módulos executam o mesmo programa dedicado ao controle dos motores elétricos, e um multiplexador, com base no "status" dos módulos, seleciona um módulo em bom funcionamento para ser conectado aos atuadores dos motores. O programa de controle reside na memória local, e é dividido em duas partes: uma parte de aquisição de dados e outra para

Fig. 4.8 : Arquitetura tolerante à falhas comunicando através de MEMÓRIA GLOBAL.

computação dos resultados.

Quando todos os três módulos estão funcionando corretamente, eles trabalham no modo tolerante a falhas, executando a votação por programação. Quando um módulo falha, eles trabalham com 2 processadores no modo de detecção de falhas. Para tolerar (ou detectar) falhas, o sistema compara os dados em duas ocasiões: depois da aquisição dos dados e depois do cálculo dos resultados (tratamento dos dados e produção de sinais de atuação). O gerenciamento do sistema é feito através de sinalizadores ("flags"), há 3 tipos destes sinalizadores:

1. Para comparação de dados - são ativados após a aquisição de dados e após a computação de resultados
2. Para indicação de estado do módulo - sinalizam quando um módulo está funcionando corretamente e quando apresenta falhas. Eles são atualizados com base no resultado da comparação de dados.
3. Para sincronização - o processo de aquisição e computação dos resultados é dividido em eventos. Estes sinalizadores são utilizados para sincronizar estes eventos.

É através da memória global, que todos os módulos têm acesso aos dados para votação/computação e

acesso aos sinalizadores para tomada de decisões e sincronização. A memória, o controlador de acesso à memória, e o circuito de multiplexação, não são triplicados, sendo construídos com circuitos de alta confiabilidade.

Para sincronização, os módulos consultam os sinalizadores de sincronização armazenados na Memória Global. Quando um módulo termina um evento, por exemplo a aquisição de um dado, ele indica esta situação através de um sinalizador correspondente na Memória Global. Em seguida, este módulo fica esperando os outros 2 módulos, terminarem a aquisição de dados. Tão logo ele receba o aviso dos outros módulos, ele parte para a comparação dos dados. Ao terminar este evento, novamente sinaliza indicando o término da comparação e espera a resposta dos outros. Nesta fase, para sincronizar com os outros módulos, o módulo em questão, se assegura que os outros estão saudáveis, através do exame dos sinalizadores de estado do módulo. Se há um módulo com falhas, ele é ignorado. Quando não há problema, o módulo espera pelo sinalizador de término de comparação, durante um intervalo de tempo fixo. Se neste intervalo ele não recebe a resposta do outro módulo, ele atualiza o sinalizador de estado do outro módulo, indicando que este apresenta falhas. Depois de sincronizar com os dois outros módulos, eles seguem juntos para o processamento do dado. Em cada etapa, toda vez que termina um evento, o sistema entra no processo de sincronização. Há 4 eventos gerais: aquisição de dados; término da comparação de dados; término do

processamento de dados; término da comparação dos cálculos. Em todas etapas um módulo envia seu sinal e fica esperando pela resposta do outro.

4.4.1 A SINCRONIZAÇÃO NA ARQUITETURA EM ANEL

Na Arquitetura em Anel, com relógios independentes, o tempo de aquisição e processamento dos dados, pode diferir de um módulo para outro. Esta divergência, pode ocorrer devido à diferenças intrínsecas, nos elementos de aquisição de dados (transdutores, conversores, acionadores de disco), e à diferença de fase nos próprios relógios de cada módulo. Como a votação deve ser realizada sobre a mesma versão de dados, os módulos devem iniciar a votação ao mesmo tempo. Assim, há a necessidade de se estabelecer um ponto de encontro, para que todos os módulos andem juntos a partir daí. No caso da Arquitetura em Anel o ponto de encontro nos processos, é o ponto onde cada processador apronta o dado para votação. A partir daí é estabelecido um "diálogo" entre os módulos através de comunicação síncrona. O mecanismo utilizado, é o da transmissão do dado com reconhecimento ("handshake"), cuja implementação foi descrita no capítulo 3. Na figura 4.9.a., está mostrada a sincronização dos módulos por transmissão com reconhecimento. O módulo que apronta o dado, o coloca no porto de saída e avisa o módulo receptor (próximo módulo no anel-circulação horária). A partir daí, ele fica esperando o aviso de reconhecimento do receptor.

Nesta espera, ele também verifica se chegou um dado vindo do módulo anterior a ele no anel. Somente após o recebimento do reconhecimento da sua transmissão, e do envio de seu reconhecimento de recepção, para o módulo anterior, é que o módulo em questão, parte para a rotina de votação. Como no anel, os três estão amarrados um ao outro, e há o ciclo de espera, todos entram juntos na rotina de votação, e fazem a comparação sobre a mesma versão de dados. Este processo ocorre todas as vezes que há um dado para ser enviado à saída do sistema, de maneira que os módulos mantêm a sincronização, durante todo o processamento. Este processo de sincronização, através de ponto comum de encontro ("rendez vous"), garante sincronização total [KIRNER, 1989].

Um módulo pode perder a sincronização devido a uma falha transiente, ou a uma falha permanente. Na figura 4.9.b, está mostrada a sincronização, no caso de um módulo falho. No caso de uma falha transiente, o módulo pode ser recuperado no início do próximo processamento, através do sinal de reinicialização, dado sempre que se inicia um novo processamento. Durante o corrente processamento, onde ocorreu a falha transiente, o sistema opera com os 2 módulos restantes, mascarando o módulo falho. Isto é o que acontece na atual implementação. Se o sistema for direcionado para executar tarefas periódicas, por exemplo no controle de um processo, então o sistema deve procurar recuperar o módulo que sofreu a falha transiente, no início do próximo ciclo de controle. Além disso, é necessário resincronizar o sistema no início

Fig. 4.9.a : Sincronização dos módulos através de transmissão com reconhecimento.

- (1) ENVIA O DADO
- (2) RECEBE O RECONHECIMENTO

Fig. 4.9. b : Sincronização no caso de falha em um módulo.

Fig. 4.9 : Sincronização na arquitetura em anel.

de cada ciclo de controle, para que os três módulos façam a aquisição dos dados, correspondendo ao mesmo estado do sistema. Se o sistema opera numa configuração mestre/escravo, como em [OBAC RODA, 1983.a], o mestre é o encarregado de resincronizar o sistema ao início do ciclo. Embora simples, este esquema tem a desvantagem de ter um sistema tolerante a falhas, dependendo da atuação de um mestre vulnerável a falhas. Para manter o sistema totalmente independente de um ponto vulnerável, pode ser usado o método de sincronização votada, proposta em [DAVIES, 1978] e descrito em 4.3.3.b, para sincronizar os ciclos de controle. Ele permitiria a re-entrada no sistema, de um módulo que sofreu uma falha transiente. Na Arquitetura em Anel, o próprio porto usado para transmissão dos sinalizadores de erro, poderia ser usado para transmitir o estado do módulo, indicando o fim de um ciclo de controle. Cada módulo então monitoraria os outros, e entraria em uma rotina de espera, até que a maioria dos módulos completassem o ciclo. No final desta espera, cada um iria para o início do ciclo de controle novamente. Isto seria feito sem nenhum custo adicional de circuitos. Uma futura aplicação desta arquitetura, será configurá-la para o controle de processos, e implementar a sincronização do ciclo de controle, através do método de sincronização votada.

4.5 ANÁLISE

Comparando o processo de sincronização utilizado na Arquitetura em Anel, com os outros processos, verifica-se que ele é extremamente simples e flexível. A flexibilidade está no procedimento de sincronização, através de transmissão do dado com reconhecimento, permitindo que os módulos executem seus processamentos em velocidades diferentes. Isto traz a vantagem de deixar o sistema menos sujeito a falhas correlatas.

Esta solução, parece ser uma boa alternativa em relação ao uso de relógios, para sincronizar os sistemas NMR. Em toda literatura pesquisada a esmagadora maioria faz uso do relógio (referência de tempo), como modo de sincronização, seja através de relógios externos de fase amarrada, seja através de realimentação mútua utilizando algoritmos. A maioria dos algoritmos são teóricos e a implementação em sistemas reais, nem sempre é imediata. Algumas aplicações apresentadas, por sua vez [INFIS, 1988], se restringem a sistemas bem simples. Os relógios de fase amarrada tolerantes a falhas, continuam sendo pesquisados [KRISHNA, 1990], e são indicados principalmente para sistemas que trabalham com controle em tempo real, por exemplo em aeronaves. Por outro lado, a construção do próprio relógio não é simples, e não basta que o relógio seja tolerante a falha. O meio de transmissão usado para o envio do sinal ao sistema, também precisa ser tolerante a falhas. Este foi o procedimento mostrado para o "FTMP", onde a transmissão se faz através

de linhas redundantes submetidas a votação. O controle em tempo real não exclui totalmente o modo de sincronização da Arquitetura em Anel, apesar dela usar programação para sincronização. O sistema "SIFT" usa algoritmos, para sincronizar o sistema. Este sistema foi projetado para controle em aeronaves em tempo real, e mesmo assim, gasta tempo na rotina de sincronização lendo os relógios dos módulos, e calculando o vetor de consistência. No anel, quando o dado está pronto, ele é imediatamente colocado na saída, e a maior perda de tempo fica por conta do ciclo de espera. Este ciclo de espera pode ser ajustado a um valor ótimo, já que ele é implementado por programação. As publicações [WENSLEY, 1978] [BUTLER, 1978] sobre o "SIFT", não trazem o tempo gasto no processo de sincronização, mas o sistema August-serie 300 [WENSLEY, 1983.b] [WENSLEY, 1982], baseado no projeto SIFT, indica 200 micro- segundos [WENSLEY, 1981], como tempo de espera, para que os módulos fiquem sincronizados. O August-300 usa o microprocessador de 16 bits e relógio de 5MHz e não faz referências ao tempo gasto na rotina de sincronização. No anel, usando dados do primeiro sistema estudado [OBAC RODA, 1983.a], o tempo gasto para transmissão é de 45 micro-segundos e 555 micro-segundos para completar o reconhecimento, e ciclo de espera. Isto é feito com um microprocessador de 8 bits e relógio de 2MHz.

Com relação ao uso de sincronização mútua, através de algoritmos, a arquitetura em anel, também apresenta vantagens. Grande parte dos algoritmos publicados, utiliza $3f + 1$ módulos/relógios para tolerar f

falhas. Para diminuir o número de módulos, são feitas restrições quanto a natureza das falhas, com foi visto no item 4.3.3.a. A Arquitetura em Anel apresentada, faz a sincronização com $2f + 1$ módulos e falhas do tipo maliciosa não afetam o sistema. Desta forma, não há necessidade de se usar vários níveis de sincronizadores, como no modelo de sincronização votada. Um módulo envia o seu dado, para apenas o próximo vizinho no anel. Não há perigo de um módulo falho enviar diferentes versões do mesmo dado para dois módulos diferentes. Alguma restrição a esta arquitetura, poderia haver quanto a uso de um único canal de comunicação, para o diálogo de sincronização entre dois módulos. Isto porém não constitui um entrave à sincronização do sistema, porque uma falha em um canal é considerada falha do módulo. O sistema foi justamente projetado para tolerar falhas em um módulo. Os outros dois continuam sincronizados.

A sincronização da Arquitetura em Anel, também é mais simples, comparando com o sistema apresentado em [CHANDE, 1989], item 4.4, que usa também um ponto comum de encontro, através de sinalizadores. No sistema com sinalizadores, a Memória Global é o meio de comunicação. De início, ele representa um ponto vulnerável para a arquitetura, mesmo construído com circuitos de alta confiabilidade. Na sincronização do Sistema em Anel não há este ponto vulnerável ao sistema como um todo. Também o gerenciamento do acesso à Memória Global, parece ser mais complexo que o simples uso de portas paralelos no anel. Com a Memória Global, cada módulo tem que requisitar o

acesso à memória, esperar que o controlador conceda o acesso, para só então ler os sinalizadores de sincronização dos outros módulos. Somente um módulo por vez lê, ou escreve na memória. No anel, usando o circuito integrado 8255, operando no modo 1, os sinais de reconhecimento são enviados pelo próprio circuito [INTEL, 1981], facilitando bastante a implementação das rotinas de comunicação e sincronização.

CAPÍTULO 5

ANÁLISE DO SISTEMA

5.1 ANÁLISE DA TOLERÂNCIA A FALHAS.

Para analisar o sistema implementado, quanto aos aspectos de tolerância a falhas, serão considerados cinco aspectos: redundância, detecção de falhas, diagnose, reparo e recuperação da operação após a falha.

A arquitetura implementada baseada no uso de três módulos, assegura de início o mascaramento da falha em um módulo. O uso de três módulos, leva vantagem em relação a simples duplicação. Na duplicação, embora seja possível detectar a ocorrência de uma falha, nem sempre é imediato identificar qual dos dois módulos está errado. Geralmente técnicas adicionais de detecção de erro têm que ser implantadas no sistema. Com o uso de 3 módulos, o processo de votação é o único responsável pela detecção de erro. No sistema implementado, a votação sobre um número parcial de dados é responsável pela detecção e diagnose das falhas. Quando uma falha ocorre, ela é detectada através da comparação dos dados em cada módulo. Se os dados não concordam, então uma falha é detectada. A diagnose, que permite identificar o módulo com problemas, é executada através da análise da configuração dos sinalizadores de erro. As configurações estão mostradas na tabela 5.1. O módulo que dá a saída, avisa o operador qual módulo está com problemas. Testes adicionais podem ser implementados para diagnose do tipo de falha ocorrida: se

no processador, no subsistema de comunicação ou na transmissão. Na implementação atual é executado um teste para verificar erros na transmissão (item 5.2.2). Identificado o módulo com problema, o reparo pode ser efetuado, retirando-se o módulo, enquanto os outros continuam a operar normamente (item 5.2.3). O sistema se recupera com a recolocação do módulo, e no início de um novo processamento.

5.2. ANÁLISE DA ARQUITETURA

No Sistema em Anel implementado, a computação tolerante a falhas está baseada em 3 características fundamentais: redundância de circuitos, votação parcial por programação e facilidade de reparo.

A implementação da redundância por circuitos foi descrita no capítulo três. Como foi visto, ela se baseia na comunicação entre os módulos através de um anel. Esta arquitetura traz algumas vantagens: a facilidade de comunicação entre os módulos; a flexibilidade para expandir o sistema para uma estrutura NMR ("N Modular Redundancy"), com mais módulos e a liberdade de operação dos módulos, através de uma sincronização menos rígida.

5.2.1 ANÁLISE DA COMUNICAÇÃO

Das arquiteturas NMR tolerantes a falhas,

pesquisadas dois sistemas têm algumas características que permitem fazer alguma comparação: o sistema August-300 [WENSLEY, 1981, 1982, 1983.b] e o sistema apresentado em [CHANDE, 1989], chamado "Modular TMR Multiprocessor System", que será tratado daqui para frente com MMS.

O sistema AUGUST-300, é um sistema TMR, baseado no microprocessador 8086, e faz votação e sincronização por programação. A votação se baseia no princípio de votação por maioria. A sincronização é feita através de algoritmos no esquema de realimentação mútua, onde o sistema executa rotinas de sincronização. O conjunto para controle de processo é composto por 3 módulos processadores (CCM - "Control Computer Modules") uma interface com o processo (PIM - "Process Interface Module") e módulos de interface com os sensores e atuadores (TM - "Termination Modules").

A figura 5.1 mostra o sistema. Cada módulo processador tem seu microprocessador 8086, 8 Kbytes de "EPROM" e 32 Kbytes de RAM e relógio de 5MHz. A tolerância a falhas é obtida através da votação por maioria. Se há uma falha, esta será reconhecida pela votação por maioria, a falha será indicada, mas não terá nenhum efeito nos cálculos posteriores. A detecção da falha é feita simplesmente, para indicar que há a necessidade de reparo [WENSLEY, 1982]. Quando cada módulo processador termina o seu cálculo, ele envia o resultado para o módulo de interface PIM, onde a saída correta é selecionada através de votação por circuitos, para eliminar as falhas nos resultados. O circuito votador é também tolerante a falhas. Assim a combinação dos 3 processadores e o

Fig. 5.1: ARQUITETURA DO SISTEMA AUGUST - 300.

circuito votador na saída, pode sobreviver a um erro em, qualquer cálculo, em um módulo ou uma falha no circuito votador. Mas não sobrevive a uma dupla falha, tal que um processador falhe e também um componente do circuito votador. O sistema é baseado na arquitetura "SIFT" [WENSLEY,1978], e comercializado pela "August Corporation".

A comunicação entre os módulos processadores neste sistema August é feita através da leitura da memória. Quando há a votação, cada processador lê a memória dos outros dois processadores, para a obtenção de dados. Como a operação é apenas de leitura ("Read Only"), um módulo falho não corrompe os dados de um módulo funcionando corretamente. Apesar de se comunicarem, os módulos estão isolados um do outro, no que diz respeito a falhas.

O outro sistema "MMS" [CHANDE, 1989] já foi descrito no capítulo 4 (item 4.4). Este sistema utiliza o microprocessador Z-80, tem 16 K de RAM e 8 K de EPROM. Realiza votação e sincronização por programação. Para estes procedimentos, um processador se comunica com o outro através de uma memória global. Um sistema chamado "Bus arbiter", aloca a memória global para cada microprocessador. Tanto a memória global como o alocador da memória não são tolerantes a falhas. A comunicação, sincronização e votação, é governada por sinalizadores ("flags").

A comunicação de dados para votação na Arquitetura em Anel implementada, foi descrita em 3.4.3. e se faz através da comunicação com reconhecimento

("handshake"). Ela é efetuada através de 8 linhas de dados e não há linhas extras para detecção de erros de transmissão. Os erros são detectados por programação. Um erro na transmissão é considerado uma falha no módulo transmissor. No esquema em que a votação é parcial a comunicação é extremamente simples; um módulo envia o seu dado para o vizinho da esquerda e recebe os dados do vizinho da direita. Do ponto de vista de funcionalidade este sistema leva vantagem sobre os procedimentos usados no August-300 e no MMS. No Anel não há a complexidade de circuitos exigida quando se faz chaveamento de memória. No sistema August-300, cada processador precisa acessar a memória dos outros 2 módulos. E no sistema MMS, cada processador tem de acessar a Memória Global, tanto para escrever o seu dado como para ler os dados dos outros módulos. Em ambos os sistemas, há gasto de tempo para acesso a memória. No Anel, os processadores gastam tempo também esperando o reconhecimento do dado enviado e testando a chegada de um dado, mas este processo já é utilizado também para a sincronização. Com apenas um processo o Anel envia o dado e se sincroniza. No sistema August-300 e no MMS, a sincronização envolve outro procedimento diferente e mais tempo de espera. No sistema MMS, há também a vulnerabilidade de se usar um único sistema de alocação de memória ("bus arbiter") e de Memória Global, embora sejam escolhidos componentes de alta confiabilidade. Este aspecto já foi discutido no capítulo 2, embora seja possível prever a confiabilidade de um lote de componentes, não é possível garantir 100% de confiabilidade para um componente individual.

O esquema de comunicação através do Anel, permite, que um módulo sem falha, corrija o resultado de um módulo que apresentou um dado errado para votação. Este tipo de correção não foi feito nesta implementação, mas é uma perspectiva interessante. Por exemplo se um módulo sofre uma falha transiente, dependendo do tipo de processamento executado, esta falha pode se tornar permanente ao longo do processo. Com a comunicação através dos portos, eles seriam usados como canal de comunicação, para recuperar o módulo com problemas. O seu vizinho, sem falhas, o alimentaria com os dados corretos. Já no esquema de comunicação, como o do sistema August-300, isto não é imediato, porque cada processador apenas lê a memória do outro. Para implementar este procedimento, é necessário alterar a implementação dos circuitos. No Anel, apenas é preciso alterar o programa de gerenciamento sem custo adicional de circuitos. No Anel, isto seria feito sem perigo do módulo falho corromper o módulo funcionando corretamente. A comunicação é efetuada apenas através dos portos. Vários tipos de informação poderão ser enviados através dele, dependendo da aplicação dada à arquitetura tolerante a falhas, e de como se deseja manipulá-la.

A comunicação através do Anel, também deixa o sistema menos susceptível a falhas maliciosas. As falhas maliciosas, do tipo que informações conflitantes são enviadas por um mesmo processador, para processadores diferentes, por rotas independentes, [FRISON, 1982], não afetam a transmissão de dados no Anel. Simplesmente porque a circulação de dados é unidirecional. Cada módulo envia o seu dado para apenas um outro módulo, não há o risco de

enviar informações conflitantes para processadores distintos. Já no caso de circulação bidirecional, uma falha maliciosa poderia fazer com que os processadores não chegassem a um consenso.

No sistema implementado com 3 módulos, o uso do canal não redundante, não se constitui em um ponto vulnerável da arquitetura. Com o processo de votação parcial, não há bloqueio de circulação de dados, quando há uma falha no subsistema de comunicação de dados no Anel. Porque sempre há 2 módulos adjacentes sem falhas, tendo um deles condição de efetuar a comparação de dados e dar a saída do sistema. O benefício que o uso de canais redundantes poderia trazer para esta arquitetura é o aumento da vida útil do sistema, ou seja, da sua confiabilidade e disponibilidade. Por outro lado, ao analisar esta arquitetura, ela pode ser melhorada do ponto de vista de comunicação, se a parte de transmissão for separada da parte de recepção. Neste caso, um módulo que sofra uma falha em apenas um subsistema pode continuar operando normalmente, e o sistema como um todo pode suportar mais de uma falha no subsistema de comunicação.

A arquitetura continuaria funcionando corretamente, enquanto houvesse 2 subsistemas de comunicação adjacentes, junto com os seus processadores, operando corretamente. Isto também aumentaria a vida útil do sistema. O subsistema de comunicação também poderia ser implementado independente do sistema de processamento. Quando um módulo falhasse, o seu subsistema de processamento seria capaz de retransmitir o dado vindo do módulo anterior a ele. Na arquitetura com 3 módulos

Fig. 5. 2 : SISTEMA EM ANEL COM SUBSISTEMA DE COMUNICAÇÃO INDEPENDENTE

implementada, este procedimento não traz nenhuma vantagem. Se um módulo falha, um dos outros 2 módulos irá comparar os dois dados correspondentes aos sistemas funcionando corretamente, sem a necessidade que o dado seja transmitido pelo módulo falho. A figura 5.2 ilustra este fato. Nele o módulo A tem 2 dados corretos, o de C e o seu, para comparar. Não há a necessidade de A enviar o seu dado para C, através de um subsistema de comunicação independente em B, cujo processador apresenta falhas. O uso de um subsistema independente de comunicação é interessante quando se usa mais de 3 módulos, ou quando a votação é majoritária.

5.2.2. ANÁLISE DA VOTAÇÃO PARCIAL

O processo de votação parcial por programação utilizado, tem todas as vantagens já descritas no item 2.7.2, quando se tratou da votação por programação. De um modo geral estas vantagens são: flexibilidade, menos restrições para a sincronização, não sofre desgaste. Além disso o processo, para 3 módulos, tem a mesma confiabilidade da votação por maioria (cap. 3), mas é mais simples, pois não exige que os módulos recebam os dados de todos os outros módulos. Isto facilitou o mecanismo de comunicação. A votação parcial por programação é composta de duas fases. Na primeira, cada módulo envia o dado para o seu vizinho, compara o dado recebido com o seu, e sinaliza com o resultado da comparação. Na segunda fase, cada módulo envia o sinalizador de erro para os outros

dois módulos. Todos os módulos têm que receber os sinalizadores de erro, para através da configuração dos mesmos identificar o módulo com a falha e escolher o módulo, que dará saída ao sistema. É nesta fase, que um módulo sofrendo uma falha maliciosa poderia enviar valores conflitantes de sinalizadores de erro para os outros dois módulos. Isto poderia fazer com que o sistema não chegasse a um consenso, de quem deveria dar saída ao sistema. Com apenas 1 bit para uso como sinalizador de erro, isto poderia ocorrer em várias situações, por exemplo, rompimento de uma linha de transmissão do sinalizador para um dos outros módulos. Isto fica mais claro considerando o exemplo a seguir.

Foi executada uma votação e todos os módulos estão funcionando corretamente. Todos os módulos ao ativarem os seus sinalizadores de erro indicam concordância de resultados. $EFP = 1$ (sinalizador de erro próprio = 1). Porém na transmissão dos sinalizadores de erro do módulo A para os outros módulos ocorre uma falha maliciosa. Para B, o módulo A envia o valor 0 e para C o módulo A envia o valor 1. Os outros módulos fazem suas transmissões corretamente. De acordo com a tabela 5.1, para o processador A, tudo está funcionando corretamente a configuração de sinalizadores é aquela mostrada pela linha 7 da tabela. Para C, as transmissões chegaram corretamente e a configuração é a mesma de A, ou seja, a linha 7. Para B, contudo, a configuração é aquela da linha 4. No momento de dar saída, de acordo com a configuração da tabela, coluna de saída, o módulo A chega a conclusão que deve dar saída aos dados. O módulo B conforme a configuração da

TABELA 5.1. - TABELA PARA SELEÇÃO DO MÓDULO
DE SAÍDA

	$\overline{EFA}$	$\overline{EFB}$	$\overline{EFC}$	Módulos de saída	Módulo q falho
0	0	0	0	Nenhum	Todos
1	0	0	1	MC	MA
2	0	1	0	MB	MC
3	0	1	1	MB	MC
4	1	0	0	MA	MB
5	1	0	1	MC	MA
6	1	1	0	MA	MB
7	1	1	1	MA	nenhum
Tabela obtida considerando circulação horária					
EF = 0 → dados não concordam					

linha 4, chega a conclusão que ele é quem deve dar saída. O módulo C, chega a conclusão que A deve dar saída e não faz nada. Há portanto conflito nas decisões dos módulos A e B. Se não houvesse a falha maliciosa quem deveria dar saída seria o módulo A. Para tolerar também falhas deste tipo, foram usados 2 bits para o sinalizador de erro. De forma que se tem o seguinte:

TABELA 5.2. - CONVENÇÃO PARA OS SINALIZADORES DE ERRO

B0 , B1 → BITS - EF → SINALIZADOR DE ERRO

B1 B0	EF	ESTADO
01 →	1	concordância na comparação
10 →	0	discordância na comparação (falhas comuns)
00 →	0	falha na transmissão
11 →	0	falha na transmissão

Considerando esta convenção, para os sinalizadores de erro, na tabela 5.3 (partes A e B) são mostrados todos os possíveis estados de configuração dos sinalizadores de erro no sistema, no caso de uma falha maliciosa em uma linha (1bit) de transmissão. Por exemplo no estado 18, há uma falha maliciosa na transmissão dos sinalizadores de erro do módulo B para os outros dois módulos. Para A, B transmite o valor correto 0, para C, B transmite o valor incorreto 1. A configuração do sistema na ausência de falha maliciosa, deveria ser a mesma nos três módulos ou seja, 001, e o módulo C deveria dar saída. No estado atual, ela não é a mesma e nenhum módulo dá saída ao sistema. Ocorre uma omissão do sistema em dar saída.

TABELA 5.3.A - ESTADOS DO SISTEMA NO CASO DE FALHAS NA TRANSMISSAO - MALICIOSAS

	SAIDA DO SISTEMA					SAIDA DO SISTEMA					SAIDA DO SISTEMA					SAIDA DO SISTEMA				
	estado 0 EF _A	0 EF _B	0 EF _C	N	N	estado 0 EF _A	0 EF _B	1 EF _C	MC	MC	estado 0 EF _A	1 EF _B	0 EF _C	MB	MB	estado 0 EF _A	1 EF _B	1 EF _C	MB	MB
A*	0	0	0	N		0	0	1	MC		0	1	0	MB		0	1	1	MB	
B	1	0	0	MA	N	1	0	1	MC		1	1	0	MA	N	1	1	1	MA	N
C	0	0	0	N		0	0	1	MC	MC	0	1	0	MB		0	1	1	MB	
		1					2					3					4			
A*	0	0	0	N		0	0	1	MC		0	1	0	MB		0	1	1	MB	
B	0	0	0	N	N	0	0	1	MC		0	1	0	MB	MB	0	1	1	MB	MB
C	1	0	0	MA		1	0	1	MC	MC	1	1	0	MA		1	1	1	MA	
		9					10					11					12			
A	0	0	0	N		0	0	1	MC		0	0	0	N		0	0	1	MC	
B*	0	0	0	N	N	0	0	1	MC		0	1	0	MB	MB	0	1	1	MB	MB
C	0	1	0	MB		0	1	1	MB	N	0	1	0	MB		0	1	1	MB	
		17					18					19					20			
A	0	1	0	MB		0	1	1	MB		0	1	0	MB		0	1	1	MB	
B*	0	0	0	N	N	0	0	1	MC		0	1	0	MB	MB	0	1	1	MB	MB
C	0	0	0	N		0	0	1	MC	MC	0	0	0	N		0	0	1	MC	MC
		25					26					27					28			
A	0	0	1	MC		0	0	1	MC		0	1	1	MB		0	1	1	MB	
B	0	0	0	N	N	0	0	0	N		0	1	0	MB	MB	0	1	0	MB	MB
C*	0	0	0	N		0	0	1	MC	MC	0	1	0	MB		0	1	1	MB	
		33					34					35					36			
A	0	0	0	N		0	0	0	N		0	1	0	MB		0	1	0	MB	
B	0	0	1	MC	N	0	0	1	MC		0	1	1	MB	MB	0	1	1	MB	MB
C*	0	0	0	N		0	0	1	MC	MC	0	1	0	MB		0	1	1	MB	
		41					42					43					44			
* FALHA NA TRANSMISSAO																				

TABELA 5.3.B - ESTADOS DO SISTEMA NO CASO DE FALHAS NA TRANSMISSAO - MALICIOSAS

	SAIDA DO SISTEMA					SAIDA DO SISTEMA					SAIDA DO SISTEMA					SAIDA DO SISTEMA				
	estado	EF _A	EF _B	EF _C	NA	estado	EF _A	EF _B	EF _C	NC	estado	EF _A	EF _B	EF _C	NA	estado	EF _A	EF _B	EF _C	NA
A*	1	0	0	MA	MA	1	0	1	NC		1	1	0	MA	MA	1	1	1	MA	MA
B	1	0	0	MA		1	0	1	NC		1	1	0	MA		1	1	1	MA	
C	0	0	0	N		0	0	1	NC	NC	0	1	0	MB		0	1	1	MB	
	5					6					7					8				
A*	1	0	0	MA	MA	1	0	1	NC		1	1	0	MA	MA	1	1	1	MA	MA
B	0	0	0	N		0	0	1	NC		0	1	0	MB	MB	0	1	1	MB	MB
C	1	0	0	MA		1	0	1	NC	NC	1	1	0	MA		1	1	1	MA	
	13					14					15					16				
A	1	0	0	MA	MA	1	0	1	NC		1	0	0	MA	MA	1	0	1	NC	
B*	1	0	0	MA		1	0	1	NC	N	1	1	0	MA		1	1	1	MA	N
C	1	1	0	MA		1	1	1	MA		1	1	0	MA		1	1	1	MA	
	21					22					23					24				
A	1	1	0	MA	MA	1	1	1	MA	MA	1	1	0	MA	MA	1	1	1	MA	MA
B*	1	0	0	MA		1	0	1	NC		1	1	0	MA		1	1	1	MA	
C	1	0	0	MA		1	0	1	NC	NC	1	0	0	MA		1	0	1	NC	
	29					30					31					32				
A	1	0	1	NC	N	1	0	1	NC		1	1	1	MA	MA	1	1	1	MA	MA
B	1	0	0	MA		1	0	0	MA		1	1	0	MA		1	1	0	MA	
C*	1	0	0	MA		1	0	1	NC	NC	1	1	0	MA		1	1	1	MA	
	37					38					39					40				
A	1	0	0	MA	MA	1	0	0	MA	MA	1	1	0	MA	MA	1	1	0	MA	MA
B	1	0	1	NC		1	0	1	NC		1	1	1	MA		1	1	1	MA	
C*	1	0	0	MA		1	0	1	NC	NC	1	1	0	MA		1	1	1	MA	
	45					46					47					48				
* FALHA NA TRANSMISSAO																				

Consultando a tabela 5.3, ocorre omissão nos estados, 3, 4, 18, 23, 36. E ocorre o conflito nos estados 15, 16, 27, 29 e 45. Os estados da primeira coluna, são normalmente de omissão, pois trata-se, da situação onde não há concordância de dados em nenhum módulo. No sistema implementado, uma falha por omissão do sistema não é catastrófica, porque o sistema simplesmente deixa de enviar o dado para o vídeo. Porém uma falha por conflito leva dois módulos a acessar a mesma saída. Antes de analisar as soluções para este problema, será assumido que o sistema pode tolerar uma falha maliciosa em apenas uma linha de transmissão. No sistema isto foi implementado através da convenção mostrada na tabela 5.2. Com esta convenção se tem as situações de falhas mostrada na tabela 5.4.

Com a convenção adotada os estados a seguir, não existirão devido a uma falha maliciosa: 1, 2, 3, 4, 9, 10, 11, 12, 17, 18, 21, 22, 25, 26, 29, 30, 33, 35, 37, 39, 41, 43, 45 e 47. A transformação dos sinais é feita por programação. Com isto as falhas por omissão são eliminadas, com exceção do estado 24. O estado 39, por exemplo, deixa de existir. Neste caso a falha é na transmissão do sinalizador de C para A, e o estado considerado é aquele em que a configuração na ausência de falhas é (110), ou seja, com certeza o sinalizador de C é 10. Ocorrendo uma falha maliciosa em apenas uma linha este sinalizador só poderá ser transformado em 00 ou 11. logo a transformação no módulo A nunca irá gerar o dígito 1. Este estado não existirá, portanto. Isto também acontece com os outros estados citados. Porém, não se

TABELA 5.4. - FALHAS NA TRANSMISSÃO

valor sina- liza- do B1 B0	trans- forma- ção	valor enviado B1 B0	valor recebido no outro módulo B1 B0	transformação	falha na trans- missão	O b- s.
01	1	0 1	0 1	1	nenhuma	
01	1	0 1	0 0	0	em B0	
01	1	0 1	1 1	0	em B1	
01	1	0 1	1 0	0	em B0 e B1	
10	0	1 0	1 0	0	nenhuma	
10	0	1 0	0 0	0	em B1	
10	0	1 0	1 1	0	em B0	
10	0	1 0	0 1	1	em B0 e B1	

resolveu ainda o problema de conflito. Duas soluções podem ser usadas: auto teste para verificação da transmissão, uso da linha bastão.

Auto Teste da Transmissão: considerando o estado 46, onde se tem uma situação de conflito, o auto teste feito pelos módulos A e C resultam no seguinte: o módulo A consulta um registrador de transformação para verificar se os bits que foram transformados eram do tipo 11 ou 00. Ao verificar que um destes casos ocorreu, o módulo A considera que houve erro na transmissão e não dá saída aos dados. O módulo C, faz o mesmo procedimento, e verifica que não houve erro de transmissão e corretamente dá saída aos dados. Esta solução elimina todas as situações de conflito. Porém introduz novas situações por omissão. Por exemplo, no estado 23, a introdução desta solução faz com que o módulo A, que estaria dando saída ao sistema, deixe de fazê-lo, pois ao fazer o teste nos sinalizadores de erro, ele verifica que há erro na transmissão do sinalizador do módulo B. E o sistema deixa de dar saída, se omitindo.

Uso da Linha Bastão: Com o uso da linha bastão, nos casos de conflito, os dois módulos tentam acessar a linha bastão e acessa aquele que chegar primeiro. Esta solução tem a vantagem de não introduzir novos casos de omissão. Por outro lado a implementação de linha bastão através de portos paralelos, está sujeita aos mesmos tipos de falhas, que estão sujeitos os sinalizadores de erro. Na linha bastão há ainda os

circuitos de detecção de coincidência que também estão sujeitos a falhas. Também, alguns problemas podem surgir se o módulo que acessar a linha, não for aquele normalmente esperado. Por exemplo, no estado 15, se o módulo B acessar o bastão antes de A, ele ao consultar a tabela 5.1, indicará o módulo C como falho, quando, na realidade, a falha é na transmissão entre A e B. Isto poderia fazer com que o módulo C fosse enviado para reparo. Isto pode ser resolvido fazendo com que o módulo B teste erros na transmissão.

COMENTÁRIOS:

A votação parcial, facilita muito a implementação do sistema, tanto do ponto de vista de circuitos como de programação. Porém, o mecanismo de transmissão de sinalizadores de erro está sujeito a falhas maliciosas que podem levar o sistema a uma situação de conflito ou omissão. No sistema implementado, isto foi parcialmente resolvido através do auto teste da transmissão. No sistema acessando o monitor de vídeo, as falhas por omissão não são catastróficas. Já se o sistema vier a ser usado para controle de processo, falhas deste tipo podem ser críticas. Uma falha por omissão, pode levar a situações onde atuadores, alarmes, válvulas, deixem de funcionar. No sistema atual, a linha bastão está implementada, mas não está sendo usada para resolver as situações de conflito. Ela poderá ser utilizada, se a arquitetura implementada for usada no controle de processo. Também poderá ser utilizada se o sistema for

configurado para acessar um único dispositivo de saída, sem se omitir. O tratamento de falhas maliciosas, ou mal comportadas não é de fácil solução. Geralmente há necessidade de se utilizar mais de um nível de redundância para detecção destas falhas [FRISON, 1982] [DAVIES, 1978], ou algoritmos especiais para tratar com os dados sujeitos a estas falhas [INFIS, 1988]. O uso de 2 bits para os sinalizadores de erro, na configuração 01 e 10 (código "Gray"), já minimiza bem os efeitos destas falhas, e permite a detecção de falhas na transmissão.

5.2.3. ANALISE DA DISPONIBILIDADE

A análise da disponibilidade está baseada na capacidade de reparo do sistema. De acordo com a equação (2.3), quanto menor o tempo que o sistema fica inoperante, maior é a disponibilidade do mesmo.

A Arquitetura em Anel implementada permite que um módulo seja retirado do anel para reparo, enquanto os outros dois continuam operando normalmente. Para os módulos que ficaram, a ausência de um módulo, é idêntica à existência de um módulo em estado de falha permanente. Desta forma o reparo pode ser executado, sem que o sistema fique inoperante. O módulo retirado, pode também ser imediatamente substituído por um módulo novo. Assim o tempo de reparo, pode ficar reduzido a simples inserção de uma placa em um conector. Neste caso a disponibilidade tende a um valor bastante próximo de 1.

Ao recolocar o módulo no conector, há a necessidade de resincronizá-lo com os demais sistemas. O processo de resincronização de um módulo, enquanto os outros continuam operando (resincronização a "quente"), é um dos maiores desafios enfrentados na implementação de arquiteturas tolerantes a falhas [WENSLEY,1983]. No sistema implementado a resincronização é feita através do sinal de "RESET". O módulo reparado só começa a fazer parte do sistema TMR no início de um novo processamento. Este procedimento não traz grandes problemas ao sistema, porque os processamentos em CP/M têm um tempo determinado de execução. O único aspecto crítico, é que o sistema pode ficar inoperante se ocorrer uma falha durante o tempo em que o módulo é reparado. Na atual aplicação, em que o sistema em Anel, processa CP/M, uma resincronização "a quente" é complexa, porque o módulo inserido tem que atualizar o conteúdo de sua memória RAM com os demais módulos, e o estado do seu microprocessador, tal como Contador de Programa, Indicador de Pilha, registradores de "flags". Ao fazer este procedimento, os outros módulos têm que travar o processo em andamento, caso contrário, ao terminar a resincronização, o módulo estará atrasado em relação aos outros. Além disso, um dos dois processadores tem que colaborar nesta resincronização enviando as informações para atualização. Como efetuar a resincronização na Arquitetura em Anel, "a quente" é um tema de pesquisa, que pode ser explorado.

5.2.4 ANÁLISE DO DESEMPENHO.

Para avaliar o desempenho da Arquitetura em Anel, foi considerado quanto o sistema de gerenciamento e votação, atrasa a saída de dados, em relação a um sistema simplex. Para isto, o tempo gasto nos sistemas de gerenciamento foi calculado em 2 casos:

1- operação normal do sistema.

2- falha total em um módulo.

O primeiro caso, operação normal, ocorre quando os módulos estão operando sincronizados perfeitamente, de forma que não há perda de tempo com ciclos de espera.

No segundo caso, considera-se que um módulo está parado e não envia dados, nem sinais de reconhecimento de transmissão de dados, ou sequer sinalizadores de erro. Neste caso o contador é esgotado. Cada contador faz 65635 contagens antes de sair do laço de contagem.

Para um relógio de 4MHZ os valores obtidos foram:

Caso 1 - 360 μ secs.

Caso 2 - 3,4 segs.

Na operação normal, este valor está compatível com os valores tolerados para sistemas de controle. O sistema AUGUST - série 300 [WENSLEY,1981,1983.b], desenvolvido para controle de processos, tolera só no processo de sincronização, um tempo de espera de 200 μ secs. E este sistema opera com relógio de 5MHZ, e faz votação por programação, onde deve gastar mais tempo (não

citado na literatura) nestas rotinas, em relação a um sistema simplex. Ainda assim, este sistema é usado em operações críticas, tais como controle de reatores químicos sujeitos á altas pressões [WENSLEY, 1982]. Na arquitetura em anel o tempo gasto de 360 μ segs. já é o tempo total para sincronização e votação (com relógio de 4MHZ).

Na ocorrência de falhas, o tempo gasto de 3,4 segs, foi calculado considerando o pior caso, em que o contador faz 65535 contagens. Este tempo pode ser ajustado de acordo com a aplicação dada ao sistema. Por exemplo, se o sistema espera apenas 10 vezes antes de continuar o processamento, o tempo gasto cai para 792 μ segs

Este desempenho, torna esta arquitetura promissora para uso no controle em tempo real, principalmente se forem usados microprocessadores mais rápidos e poderosos do que o Z80.

CAPÍTULO 6

CONCLUSÃO E PERSPECTIVAS

Através do estudo da Arquitetura em Anel e da sua implementação, chega-se a conclusão que este sistema abre uma perspectiva diferente na pesquisa em arquiteturas tolerantes a falhas. É uma arquitetura flexível, que facilita o procedimento de sincronização, e a comunicação entre os módulos, para o mascaramento, detecção e diagnose de falhas. Esta arquitetura é adequada para computação de uso geral, e também no controle de processos.

Para uso geral, a implementação da arquitetura com 3 sistemas de disco flexível, aumenta a segurança de funcionamento do sistema em Anel. O sistema STRATUS, é um sistema tolerante a falhas, (com duplicação), voltado para transações comerciais, que usa duplicata dos sistemas de disco, com idênticas cópias de programas. Além da duplicação de dados, isto é feito porque os sistemas de discos, não são facilmente sincronizados [HENDRIE, 1983]. Assim, a pesquisa usando a arquitetura em Anel, e tentando variar as configurações dos sistemas de disco, pode apresentar soluções, para lidar com o acesso a estes dispositivos de E/S, nos sistemas tolerantes a falhas.

Outro problema aberto para pesquisa, é a resincronização de um módulo com o sistema, após o reparo. Segundo Wensley, [WENSLEY, 1983] este é um desafio para os sistemas tolerantes a falhas, principalmente no caso

apresentado neste trabalho, em que o sistema é de uso geral.

Também o desenvolvimento de Arquiteturas Híbridas, a partir deste núcleo, é outro problema que pode ser estudado. No Anel, o chaveamento de módulos reservas pode resolver aspectos críticos [OBAC RODA, 1988], tais como o aumento da complexidade dos circuitos de chaveamento, quando aumenta o número de reservas.

No controle de processos, há o campo representado pelas aplicações críticas, onde a Arquitetura em Anel pode ser usada. Há ainda os problemas concernentes com a própria arquitetura em si, quando aplicada ao controle de processos. Estes problemas podem ser: tratamento de dados legitimamente diferentes, falhas maliciosas na aquisição de dados analógicos, técnicas para resincronização "a quente" do sistema na presença de falhas. Para controle de processos, "on-line", é necessário que o sistema tenha uma certa capacidade de memória para armazenagem de dados. Geralmente estes sistemas, operam em ambientes não controlados, sujeitos a uma vasta gama de agressões. Os sistemas de discos flexíveis são sensíveis a ambientes assim, e também dependendo do processo e do controle, são dispositivos muito lentos para armazenagem de dados. A arquitetura implementada tem duas características que permitem resolver problemas como estes. O uso de triplicata para os sistemas de discos, é uma forma de tolerar a agressividade do meio ambiente. Não é tarefa fácil implementar esta características, nas arquiteturas TMR convencionais [SIEWIOREK, 1978]. É necessário ter cuidado com a sincronização no acesso aos discos. Nas

situações, em que for necessário uma quantidade maior de memória, a Arquitetura em Anel deste trabalho, tem implementados 64 Kbytes em RAM. Pode-se aumentar esta capacidade, usando bancos de memórias como a proposta apresentada em [TRAINA, 1986].

A Arquitetura em Anel, é também independente de um particular conjunto de instruções, de modo que pode ser explorada com microprocessadores mais avançados.

BIBLIOGRAFIA

[ANDERSON, 1981]

ANDERSON, T. e Lee, P.A. : "Fault Tolerance Principles and Practices"; Londres; Prentice-Hall International; 1981

[AVIZIENIS, 1975]

AVIZIENIS, A.: "Architecture of Fault Tolerant Computing Systems"; 5th. Int. Symp. on Fault Tolerant Computing; Paris; 1975; pp. 3 - 36.

[AVIZIENIS, 1976]

AVIZIENIS, A. : "Fault Tolerant Systems"; IEEE Trans. on Comp.; vol. C-25; 1976; pp. 1304.

[BABAOĞLU, 1987]

BABAOĞLU, Ö e Drummond, R. : "(Almost) No Cost Clock Synchronization"; 17th. Int. Symp. on Fault-Tolerant Computing ; Pittsburgh; 1987; pp. 42 - 47

[BARTLET, 1977]

BARTLET, J.F. : "A Non Stop Operating System" ; publicação da Tandem Computers, Inc; ou Cap 11 de [SIEWIOREK, 1983].

[BRODSKY, 1980]

BRODSKY, M. : "Hardening RAMs Against Soft Error"; Electronics; vol. 5; McGraw Hill; 1980

[BUTLER, 1985]

BUTLER, R.W. ; Palumbo, D.L. ; Johnson, S.C. : "Application of a Clock Synchronization Validation Methodology to the SIFT Computer System" ; 15th. Int.Symp.on Fault-Tolerant Computing; Ann Arbor ; junho ; 1985; pp. 194 - 199

[CHANDE, 1989]

CHANDE, P.K. ; Ramani, A.K. e Sharma, P.C. : "Modular TMR Multiprocessor System"; IEEE Trans. on Industrial Electronics; vol. 36, nº 1; fevereiro, 1989; pp. 34 - 41

[CHEN, 1978]

CHEN, L. e Avizienis, A. : "N-Version Programming: A Fault Tolerance Approach to Reliability of Software Operation"; 8 th Symp. on Fault-Tolerant Computing; junho de 1978; pp. 3 - 9

[CRISTIAN, 1986]

CRISTIAN, F. ; Aghili, H. e Strong, R. : "Clock Synchronization in the Presence of Omission and Performance Faults, and Processor Joins"; 16th. Int. Symp. on Fault Tolerant Computing; Viena; junho 1986; pp. 218 - 223

[CRISTIAN, 1989]

CRISTIAN, F. : "Question to Ask when Designing Attempting to Understand a Fault-Tolerant Distributed System"; III Simpósio de Computadores Tolerantes à Falhas; PUC-RJ; setembro, 1989, pp. 1 - 18

[DALY, 1973]

DALY, W.M. ; Hopkins, A.L. e McKenna, J.F. : "A Fault-Tolerant Digital Clocking System"; 3th. Int. Symp. on Fault-Tolerant Computing; Palo Alto, junho 1973; pp. 17 - 21

[DAVIES, 1978]

DAVIES, D. e Wakerly, J.F. : "Synchronization and Matching in Redundant Systems"; IEEE Trans. on Computers; vol. C-27 ; Nº 6 ; junho de 1978; pp. 531 -539

[DE SOUZA, 1975]

DE SOUZA, M. & Paz, E.P. : "Fault-Tolerant Digital Clocking Systems"; Electronics Letters, vol. 11, nº 18; setembro, 1975; pp. 433 - 434

[DRUMMOND, 1987]

DRUMMOND, R. : "Sistemas Distribuídos"; II MINI CURSO, Introdução a Tolerância à Falhas, cap. 5; agosto, 1987, pp. 147 - 171

[FORBES, 1985]

FORBES, M. : "MTBF Values Aid in Predicting Component Failures"; EDN, Nº 13; junho de 1985; pp. 177 - 180

[FISCHER, 1989]

FISCHER, D.P.M. e Obac Roda, V. : "Desenvolvimento de um Sistema de Computação Tolerante à Falhas, de uso Geral e Baixo Custo"; Anais do III Simpósio de Computadores Tolerantes à Falhas; setembro de 1989; pp.110 - 119

[FRAGA, 1987]

FRAGA, J. e Lemos, R. : "Conceituação e Terminologia em Sistemas Informáticos Tolerantes a falhas e Intrusões"; Anais do 2º Simpósio em Sistemas de Computadores Tolerantes à Falhas; agosto 1987; pp. 177 - 188

[FRISON, 1982]

FRISON, S.G. e Wensley, J.H. : "Interactive Consistency and its Impact on the Design of TMR Systems"; 12th. Int. Symp. on Fault-Tolerant Computing; Santa Monica; 1982; pp. 228 - 233

[HEDRIE, 1983]

HEDRIE, G. : "Fault-Tolerant Computers: A Hardware Solution to Part Failures Totally Insulates Programs"; Electronics; vol. 56 ; Nº 2 ; janeiro, 1983; pp. 103-105

[HOGAN, 1983]

HOGAN, T. : "CP/M Guia do Usuário"; McGraw Hill; 1983; 317 pp.

[HOPKINS, 1978]

HOPKINS, A.L. ; Smith, T.B. e Lala, J.H. : "FTMP - A Hight Reliable Fault-Tolerant Multiprocessor for Aircraft; Proc. IEEE, vol. 66, nº 10, outubro 1978; pp. 1221 - 1239

[INFIS, 1988]

INFIS, A.H. , : "Fault-Tolerant Synchronization, Data Matching and Self-Starting in Redundant Systems" ; Tese de Doutorado; Departamento de Eletrônica e Ciência da Computação da Universidade de Southampton; Inglaterra; agosto, 1988; pp. 307

[INTEL, 1981]

INTEL: "Component Data Catalog"; Intel Corporation; 1981

[JOHNSON, 1984]

JOHNSON, B.W.: "Fault Tolerant Microprocessor based Systems"; IEEE MICRO; 1984;
pp. 6 - 21

[JONES, 1979]

JONES, C.P.: "Automatic Fault Protection in the Voyager Spacecraft"; AIAA Paper Nº
79 -1919; American Institute of Aeronautics and Astronautics; 1979.

[KATZMAN, 1977]

KATZMAN, J.A. : "A Fault-Tolerant Computing System"; publicação da Tandem
Computers, Inc.; ou Cap. 11 de [SIEWIOREK, 1982]

[KIRNER, 1988]

KIRNER, C. e Mendes, S.B.T. : "Sistemas Operacionais Distribuídos"; Editora Campus
Ltda; Rio de Janeiro, 1988; 184 pp

[KIRNER, 1989]

KIRNER, C. : "Sistemas Operacionais para ambientes paralelos"; Apostila do IX
Congresso da SBC e VIII Jornada de Atualização em Informática; Uberlândia-MG;
julho, 1989; 57 pp

[KESSELS, 1984]

KESSELS, J.L.W. : "Two Designs of a Fault-Tolerant Cloking System"; IEEE Trans. on
Comp.; vol. C-33; nº 10; outubro de 1984; pp. 912 - 919

[KLAASSEN, 1977]

KLAASSEN, K.B. e Van Peppen, J.C.L. : "Reliability Improvement by Redundancy Voting
in Analogue Electronics Systems"; Microelectronics and Reliability; vol. 16; 1977;
pp. 593 - 600

[KRISHNA, 1985]

KRISHNA, C.M. ; Shin, K.G. e Butler, R.W. : "Ensuring Fault-Tolerance of Phase-Locked
Clocks"; IEEE Trans. Comp.; vol. C-34; nº 8; agosto 1985; pp. 752 - 756

[KRISHNA, 1990]

KRISHNA, C.M. : "Fault-Tolerant Synchronization using Phase-Locked Clocks";
Microelectron. Reliab.; vol. 30, nº 2; 1990; pp. 275 - 287

[LADUZINSKY, 1985]

LADUZINSKY, A.J. : "Fault-Tolerant Control Means Never Having to say You're
Sorry"; Control Engineering; fevereiro, 1985; pp. 71-74

[LAGES, 1986]

LAGES, N.A.C. e Nogueira, J.M.S. : "Introdução aos Sistemas Distribuídos" ; Editora
UNICAMP; 1986; 229 pp

[LAMPORT, 1978]

LAMPORT, L. : "Time, Clocks, and the Ordering of Events in a Distributed System" ;
Communications of the ACM; julho 1978, vol. 21 ; nº 7; pp. 558 - 565

[LAMPORT, 1985]

LAMPORT, L. e Melliar-Smith, P.M. : "Synchronising Clocks in the Presence of
Faults"; Journal of ACM, vol. 32, nº 1; janeiro, 1985; pp. 52 - 78

[LAPRIE, 1982]

LAPRIE, J.C. e Costes, A. : "Dependability: A Unifying Concept for Reliable
Computing"; 12th Symp on Fault-Tolerant Computing; junho, 1982; pp. 18-21

[LAPRIE, 1985]

LAPRIE, J.C. : "Dependable Computing and Fault Tolerance: Concepts e Terminology";
15th Symp. on Fault-Tolerant Computing; junho, 1985; pp. 2-11

[LEITE, 1987]

LEITE, J.C.B. e Filho Loques, D.G. : "Software" ; Mini Curso - Introdução à Tolerância
à Falhas; Cap. 4; DCC UNICAMP; CPqD TELEBRÁS; agosto 1987; pp. 99 - 146

[LOMBARDI, 1982]

LOMBARDI, F. e Obac Roda, V. : "Software Implemented Fault Tolerance - A
Methodology" ; Microelectronics and Reliability; vol. 22 ; Nº 3 ; 1982; pp. 873
- 885

[LUNDELIUS, 1983]

LUNDELIUS, J. e Lunch, N. : "A New Fault-Tolerant Algorithm for Clock Synchronization"; Proc. of the 3rd ACM Symposium on Principles of Distributed Computing, 1983

[MACHADO TRAINA, 1987]

MACHADO TRAINA, A. : "Desenvolvimento de Software Básico para Gerenciamento de um Sistema Multiusuário Distribuído"; Tese de Mestrado, Instituto de Ciências Matemáticas de São Carlos-USP; 1987; 217 pp

[MARSHALL, 1983]

MARSHALL, T.G. e Attikiouzel, J.A. : "Floppy Disk Data Transfer Techniques"; IEEE MICRO; dezembro de 1983, pp. 17 - 23

[MATHUR, 1975]

MATHUR, F.P. e De Souza, P. : "Reliability Modeling and Analysis of General Modular Redundant Systems"; IEEE Trans. Reliability ; R-24 (5) ; 1975 ; pp. 296 - 299

[MATHUR, 1976]

MATHUR, F.P. e Avizienis, A. : "Reliability Analysis and Architecture of a Hybrid Redundant Digital System: Generalized Triple Modular Redundancy with Self Repair"; Spring Joint Comp. Conf.; vol. 36; maio 1976; pp. 375 - 383

[MAXION, 1987]

MAXION, R.A. ; Siewiorek, D.P. e Elkind, S.A. : "Techniques and Architectures for Fault Tolerant Computing"; Ann. Rev. Comp. Sci.; nº 2, 1987; pp. 469 - 520

[MCCONNEL, 1981]

MCCONNEL, S.R. e Siewiorek, D.P. : "Synchronization and Voting"; IEEE Trans. on Computations; vol. C-30 , nº 2 ; fevereiro , 1981 ; pp. 161 - 164

[MINI CURSO 1987]

MINI CURSO: "Introdução a Tolerância a Falhas"; DCC Unicamp; CPqD TELEBRAS; agosto 1987; pp. 171

[MINI CURSO, 1989]

MINI CURSO: "Fundamentos de Computação Tolerante à Falhas: Técnicas de Tolerância à Falhas em Software"; III Simpósio de Computadores Tolerantes à Falhas; Cap. V; setembro de 1989; pp. 81 - 95

[NICHOLS, 1982]

NICHOLS, E.A. ; Nichols, J.C. e Musson, K.R. : "Data Communications for Microcomputers"; McGraw-Hill Book Company, 1982; 263 pp

[OBAC RODA, 1982.a]

OBAC RODA, V. e Lombardi, F. : "Software Implementation of a Disagreement Detector for a Duplex Hybrid System"; of fifth European Conf. on Electrotechnics; EUROCON'82; Copenhagen; junho de 1982; pp. 673 - 678

[OBAC RODA, 1982.b]

OBAC RODA, V. e Davies, O.J. : "Aspects of Fault-Tolerant Ring Structures"; IEEE Colloquium on Microprocessor Application Requiring High Integrity and Fault-Tolerance, IEE Digest 1982/67, Londres.

[OBAC RODA, 1983.a]

OBAC RODA, V. : "Techniques for High Integrity Intelligent Instrumentation" ; Tese de Doutorado; University College London; 1983.

[OBAC RODA, 1983.b]

OBAC RODA, V. e Davies, O.J., HARVEY, H.E., e LOMBARDI, F. : "Analysis and Implementation of System Procedures for Software Implemented Fault Tolerant Microcomputer Schemes" ; IEEE Second Annual Phoenix Conference Proceedings ; 1983 ; pp. 123 - 127

[OBAC RODA, 1985]

OBAC RODA, V. : "Aspectos Gerais de Sistemas Tolerantes à Falhas em Anel"; Anais do 1º Simpósio de Sistemas de Computação Tolerantes a Falhas; 1985; pp. 25 - 45

[OBAC RODA, 1988.a]

OBAC RODA, V.D.: "Técnicas para Computação Altamente Confiável"; Anais do XXI Congresso Nacional de Informática ; 1988 ; pp. 629 - 636.

[OBAC RODA, 1988.b]

OBAC RODA, V. : "The Hybrid Approach for Improving the Reliability of Ring Systems" ; Microelectron. Reliab. ; vol. 28 ; nº 6 ; 1988; pp. 979 - 989

[OBAC RODA, 1989]

OBAC RODA, V. : "Tolerância à Falhas em Sistemas Híbridos em Anel"; Anais do III Simpósio de Computadores Tolerantes à Falhas; Rio de Janeiro; setembro de 1989; pp. 183 - 196

[OZAKI, 1988]

OZAKI, B.M. ; Fernandez, E.B. e Gudes, E. : "Software Fault Tolerance in Architectures with Hierarchical Protection Levels"; IEEE MICRO; vol. 8; Nº 4; agosto 1988; pp. 30 - 43

[PAULA JR., 1989]

PAULA JR, A.R. ; de Souza, J.M. ; Leite, J. ; Martini, M.R.B. ; Losques Filho, D.G. : "Fundamentos da Computação Tolerante à Falhas" ; Mini Curso; III Simpósio de Computadores Tolerantes à Falhas; Cap. 1; setembro, 1989; pp. 2-24

[PEASE, 1980]

PEASE, M. ; Shostak, R. e Lamport, L. : "Reaching Agreement in the Presence of Faults"; Journal of the Association for Computing Machinery; vol. 27, nº 2; abril, 1980; pp. 228 - 234

[RAMABADRAN, 1988]

RAMABADRAN, T.V.; Gaitonde, S.S. : "A Tutorial on CRC Computations"; IEEE MICRO; vol, 8; Nº 4; agosto 1988; pp. 62d - 75

[RENNELS, 1980]

RENNELS, D.A. : Distributed Fault-Tolerant Computer Systems"; Computer, vol. 13; nº 3, 1980; pp. 55-65.

[SIEWIOREK, 1978]

SIEWIOREK, D.P.; Keni, V.; Mashburn, H.; Mc Connel, S. e Tsao, M. : "C.vmp - A Voted Multiprocessor"; Proc. of the IEEE; vol. 66; Nº 10; outubro 1978; pp. 1190 - 1198

[SIEWIOREK, 1982]

SIEWIOREK, D.P. e Swarz, R.S.: "The Theory and Practice of Reliable System Design"; Digital Press; USA; 1982; pp. 772.

[TAROUCO, 1985]

TAROUCO, L.M.R. : "Redes de Comunicação de Dados"; Livro Técnicos e Científicos Editora S.A.; 1985; 224 pp

[THEURETZBACHER, 1986]

THEURETZBACHER, N. : "VOTRICS - Voting Triple Modular Computing System"; 16th. Int. Symp. on Fault-Tolerant Computing; Viena; 1986; pp. 144 - 150

[TRAINA, 1986]

TRAINA, JR., C. : "Máquina e Modelo de Dados Dedicados para Aplicações de Engenharia"; Tese de Doutorado, Instituto de Física e Química de São Carlos-USP; Departamento de Física e Ciências dos Materiais; 1986; 379 pp

[VON NEUMAN, 1956]

VON NEUMAN, J. : "Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components"; Automata Studies, Princeton University Press; 1956; pp. 43 - 48

[VASANTHAVADA, 1984]

VASANTHAVADA, N. e Marions, P.N. : "Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Failures"; IEEE Trans. on Comp.; vol. C-37; fevereiro, 1984; pp. 73 - 76

[VASANTHAVADA, 1986]

VASANTHAVADA, N. Marinos, P.N. e Mersten, G.S. : "Design and Performance Evaluation of Mutually Synchronized Fault-Tolerant Clock System"; 16th. Int. Symp. on Fault-Tolerant Computing; Viena; junho 1986; pp. 206 - 211

[WAKERLY, 1976]

WAKERLY, J.F. : "Microcomputers Reliability Improvement Using Triple-Modular Redundancy" ; Proc. of The IEEE ; vol. 64 ; junho de 1976; pp. 889 - 895

[WENSLEY, 1978]

WENSLEY, J.H. ; Lamport, L.; Goldberg, J. ; Green, M.W.; Levitt, K.N.; Melliar-Smith, P.M.; Shostak, R.E.; Weristock, C.B. : "SIFT: the Design and Analisis of a Fault-Tolerant Computer for Aircraft Control"; Proc. IEEE; vol. 66; Nº 10; outubro de 1978; pp. 1240 - 1255

[WENSLEY, 1981]

WENSLEY, J.H. : "Fault-Tolerant Computers ensure reliable industrial controls"; Electronic Design; junho, 1981; pp. 129 - 135

[WENSLEY, 1982]

WENSLEY, J.H. e Harclerode, C.S. : "Programmable Control of a Chemical Reactor Using a Fault Tolerant Computer"; IEEE Trans. on Ind. Electronics; vol. IE-29, nº 4; novembro, 1982; pp. 258 - 264

[WENSLEY, 1983.a]

WENSLEY, J.H. : "An Operating System for a TMR Fault-Tolerant System"; 13th.Int. Symp. on Fault-Tolerant Computing; Milão; 1983; pp. 452 - 455

[WENSLEY, 1983.b]

WENSLEY, J.H. : "Fault Tolerant Computers: Industrial-Control System does things in threes for Safety"; Electronics; vol. 56, nº 2; janeiro, 1983; pp. 98 - 102

[WIRTHURMER, 1989]

WIRTHURMER, G. : "VOTRICS - Fault Tolerance Realized in Software"; Proced. of the IFAC Workshop on Safety of Computer Control Systems, SAFE COMP'89; dezembro, 1989; Viena, pp. 135 - 140

APÊNDICE: LISTAGEM DAS ROTINAS DE GERENCIAMENTO

RSD - Rotina de Saída dos dados do Sistema em Anel.....	1.1
RVS - Rotina de Votação e Sincronizado	1.1
RTD - Rotina de Transmissão de Dados	1.2
RTS - Rotina de Transmissão dos Sinalizadores de Erro .	1.4
MTD - Rotina Monta Tabela de Decisão	1.6

```

;RSD: ROTINA DE SAIDA DE DADOS DO SISTEMA EM ANEL
;
;

```

1000'	F5	ORG	1000H	
1001'	C5	PUSH	PSW	;SALVA O CHARACTER QUE VAI SER ENVIADO
1002'	D5	PUSH	B	;E TODOS OS REGISTRADORES
1003'	E5	PUSH	D	
1004'	32 7701'	PUSH	H	
1007'	3E 00	STA	DAAT	;PARA O VIDEO
1009'	32 7706'	MVI	A,00H	;INICIALIZACAO DAS VARIABEIS
100C'	32 7700'	STA	VENCEDOR	
100F'	3E 02	STA	SAIDA	
1011'	32 7703'	MVI	A,02H	;INICIALIZACAO DOS SINALIZADORES
1014'	32 7704'	STA	EFF	;DE ERRO
1017'	32 7705'	STA	EFA	
101A'	3E 98	STA	EFN	
101C'	D3 1F	MVI	A,ST551	;INICIALIZACAO DOS PORTOS DO ANEL
101E'	3E 8A	OUT	EC551	
1020'	D3 0B	MVI	A,ST552	;INICIALIZACAO DOS PORTOS DOS SINALIZA
1022'	3E FC	OUT	EC552	;DE ERRO
1024'	D3 0A	MVI	A,DVID	;DESABILITA A SAIDA DO VIDEO
1026'	3E FC	OUT	EVID	
1028'	D3 0B	MVI	A,SCER	;SINAIS DE CONTROLE PARA ENVIO DOS
102A'	3E FF	OUT	PA552	;SINALIZADORES DE ERRO EM ALTO
102C'	D3 1E	MVI	A,VHSH	;INICIALIZA OS SINAIS DE HANDSHAKE
102E'	CD 104C'	OUT	PC551	;EM ALTO
1031'	AF	CALL	RVS	;CHAMA A ROTINA DE VOTACAO E SINCRON.
1032'	3A 7700'	XRA	A	;LIMPA FLAG
1035'	FE 01	LDA	SAIDA	;TESTA SE E O MODULO QUE VAI SAIR
1037'	C2 1045'	CFI	01	
103A'	3A 7712'	JNZ	PRDOR	
103D'	E4 FB	LDA	HVID	
103F'	D3 0A	ANI	0FBH	;E O MODULO QUE VAI DAR A SAIDA
1041'	3E FC	OUT	EVID	;HABILITA O ENVIO DE DADOS
1043'	D3 0B	MVI	A,SCER	;SAI DEIXANDO OS SINAIS DE CONTRO-
1045'	E1	OUT	PA552	;LE DOS SINALIZADORES DE ERRO EM ALTO.
1046'	D1	POP	H	;RESTAURA OS REGISTRADORES
1047'	C1	POP	D	
1048'	F1	POP	B	
1049'	D3 10	POP	PSW	;CHAMA O DADO PARA SAIR
104B'	C9	OUT	010H	;FIM DA CROUT
		RET		

```

PRDOR:

```

```

;
;RVS: ROTINA DE VOTACAO E SINCRONIZACAO
;

```

```

104C'   C10 1084'
104F'   3E 01
1051'   21 7707'
1054'   BE
1055'   CA 106F'
1058'   AF
1059'   3A 7701'
105C'   21 7702'
105F'   01 7703'
1062'   BE
1063'   CA 106C'
1066'   3E 02
1068'   02
1069'   C3 106F'
106C'   3E 01
106E'   02
106F'   CD 1116'
1072'   CD 121B'
1075'   AF
1076'   3A 7706'
1079'   FE 00
107B'   CA 1080'
107E'   3E 01
1080'   32 7700'
1083'   C9

RVS:      CALL   RTD           ;chama rotina de transmissao de dados
          MVI    A,01H        ;testa TE=1
          LXI    H,T_E
          CMP    M
          JZ     CRTS         ;se esgotou o tempo continua
          XRA    A            ;da saida em DAD ANTERIOR e traz os
          LDA    DAAT          ;sinalizadores de erro.
          LXI    H,DANT        ;endereço do dado anterior
          LXI    B,EFP         ;endereço do flag de erro proprio
          CMP    M
          JZ     IGUAL         ;dados sao iguais
          MVI    A,02H        ;dados sao diferentes
          STAX   B             ;guarda o flag de erro
          JMP    CRTS         ;salta para chamar RTS
          MVI    A,01H        ;guarda o flag de erro
          STAX   B
          CRTS:  CALL   RTS
          CALL   MTB
          XRA    A            ;limpa os flags
          LDA    VENCEDOR
          CPI    00H          ;testa se e o vencedor
          JZ     PERDEDOR
          MVI    A,01H        ;SAIDA 1 para vencedor
          STA    SAIDA
          RET

```

;
 ;
 ;ROTINA DE TRANSMISSAO DE DADOS COM RECONHECIMENTO
 ;
 ;
 ;

```

1084'   3E 00
1086'   32 7707'
1089'   3A 7701'
108C'   D3 1D
108E'   3E FD
1090'   D3 1E
1092'   DB 1E
1094'   E6 20
1096'   FE 00
1098'   CA 10E5'
109B'   01 FFFF
109E'   DB 1E
10A0'   E6 10
10A2'   FE 00

RTD:      MVI    A,00H        ;estabelece o registro de tempo
          STA    T_E          ;esgotado
          LDA    DAAT         ;envia o dado para o proximo modulo
          OUT    PB551        ;do anel
          MVI    A,0FDH       ;VALOR DA MASCARA PARA O ACKB
          OUT    PC551
          IN     PC551         ;vai testar se ACK e zero
          ANI    020H         ;mascara para ACK
          CPI    00H          ;testa ACK
          JZ     RECDAD       ;o proximo modulo recebeu o dado
          LXI    B,TESF       ;inicializa o contador se o modulo
          IN     PC551        ;nao recebeu o dado e vai ler o porto
          ANI    10H          ;C para ver se chegou dado do anterior
          CPI    00H          ;testa STB apos fazer a mascara 10
          LBL_1:

```

```

10A4'  CA 10C2'      JZ      LEDAD1      ;chegou dado do modulo anterior
10A7'  3E 00        MVI      A,00H
10A9'  B8           CMP      B
10AA'  C2 10B1'      JNZ      DCRC1
10AD'  B9           CMP      C
10AE'  CA 10B5'      JZ      SEQ1          ;para de decrementar o contador
10B1'  0B           DCX      B
10B2'  C3 109E'      JMP      LBL1          ;fica esperando a chegada do dado
10B5'  3E 02        MVI      A,02H          ;sai da espera indicando erro na
10B7'  32 7703'      STA      EFP          ;comparacao
10BA'  3E 01        MVI      A,01          ;avisa que o tempo foi esgotado
10BC'  32 7707'      STA      T_E
10BF'  C3 10CB'      JMP      SEQ2          ;testa se o proximo modulo recebeu dado
10C2'  DB 1C        LEDAD1: IN      PC551    ;le dado do modulo anterior
10C4'  32 7702'      STA      DANT
10C7'  3E F9        MVI      A,0F9H          ;ENVIA SINAL DE HANDSHAKE
10C9'  D3 1E        OUT      PC551
10CB'  01 FFFF      SEQ2:  LXI      B,TESP    ;inicializa o contador
10CE'  DB 1E        LBL2:  IN      PC551    ;ve se proximo modulo recebeu o dado
10D0'  E6 20        ANI      020H          ;mascara para o ACK
10D2'  FE 00        CPI      00H
10D4'  CA 1109'      JZ      PRTN          ;ponto de retorno
10D7'  3E 00        MVI      A,00H
10D9'  B8           CMP      B
10DA'  C2 10E1'      JNZ      DCRC2
10DD'  B9           CMP      C
10DE'  CA 1109'      JZ      PRTN          ;ponto de retorno
10E1'  0B           DCX      B
10E2'  C3 10CE'      JMP      LBL2          ;continua no loop de espera
10E5'  01 FFFF      REC IAD: LXI      B,TESP
10E8'  DB 1E        LBL3:  IN      PC551    ;testa se chegou dado do modulo anterio
10EA'  E6 10        ANI      10H          ;mascara para STB
10EC'  FE 00        CPI      00H
10EE'  CA 110A'      JZ      LEDAD2
10F1'  3E 00        MVI      A,00H
10F3'  B8           CMP      B
10F4'  C2 10FB'      JNZ      DCRC3
10F7'  B9           CMP      C
10F8'  CA 10FF'      JZ      SEQ3
10FB'  0B           DCX      B
10FC'  C3 10E8'      JMP      LBL3
10FF'  3E 02        SEQ3:  MVI      A,02H
1101'  3A 7703'      LDA      EFP
1104'  3E 01        MVI      A,01H
1106'  3A 7707'      LDA      T_E
1109'  C9           FRTN:  RET

```

```

110A' DB 1C          LEDAD2:      IN      PA551      ;LE DADO DO MODULO ANTERIOR
110C' 32 7702'      STA      DANT
110F' 3E F9          MVI      A,0F9H      ;ENVIA SINAL DE HANDSHAKE
1111' D3 1E          OUT      PC551
1113' C3 1109'      JMP      FRTN

;
;
;RTS  ROTINA DE TRANSMISSAO DOS SINALIZADORES DE ERRO
;
;
1116' 3A 7703'      RTS:          LDA      EFP          ;carrega o acumulador com o sinalizador
;                                     ;de erro proprio
1119' C6 F0          ADI      0F0H          ;coloca SFA=0 e SFN=0,junto com o
;                                     ;sinalizador de erro
111B' D3 08          OUT      PA552      ;envia o sinalizador e avisa
111D' 01 FFFF        LXI      B,TESP      ;loop de espera
1120' DB 09          DCR1:      IN      PB552      ;ve se o modulo anterior enviou sinali
;                                     ;zador de erro e ja guarda
1122' 32 7704'      STA      EFA
1125' E6 20          ANI      20H          ;mascara para RFA
1127' FE 00          CPI      00H          ;RFA=0?
1129' CA 11DD'      JZ       DESV1        ;vai testar p/ proximo modulo
112C' 0B            DCX      B            ;decrementa o contador
112D' DB 09          IN      PB552      ;proximo modulo enviou sinalizador?
112F' 32 7705'      STA      EFN          ;guarda o sinalizador
1132' E6 10          ANI      10H          ;mascara para RFN
1134' FE 00          CPI      00H          ;RFN=0?
1136' CA 1146'      JZ       DESV2        ;
1139' 3E 00          MVI      A,00H        ;testa se esgotou o tempo de espera
113B' B8            CMF      B            ;
113C' C2 1120'      JNZ      DCR1          ;
113F' B9            CMF      C            ;
1140' CA 117A'      JZ       DESV3        ;sai indicando erro
1143' C3 1120'      JMP      DCR1        ;continua na espera
1146' 3A 7705'      DESV2:      LDA      EFN          ;le o sinalizador
1149' E6 03          ANI      03H          ;deixa so o sinalizador
114B' 32 7705'      STA      EFN          ;porque no porto tem mais sinais
114E' 3A 7703'      LDA      EFP          ;formando a palavra de reconhecimento
1151' C6 E0          ADI      0E0H          ;para o proximo modulo
1153' D3 08          OUT      PA552      ;envia ACKN=0
1155' 01 FFFF        LXI      B,TESP      ;inicializa o contador para esperar
;                                     ;que o modulo anterior leia o sinaliza
;                                     ;dor de erro
1158' DB 09          DCR2:      IN      PB552      ;recebe RFA
115A' 32 7704'      STA      EFA          ;

```


Endereço	Conteúdo	Operação	Operando	Comentário
115D'	E6 20	ANI	20H	; mascara para RFA
115F'	FE 00	CPI	00H	; RFA=0?
1161'	CA 1182'	JZ	DESV4	;
1164'	3E 00	MVI	A,00	;
1166'	B8	CMF	B	;
1167'	C2 114E'	JNZ	DCRF2	;
116A'	B9	CMF	C	;
116B'	CA 1172'	JZ	DESV5	;vai sair indicando erro
116E'	0B	DCX	B	;
116F'	C3 1158'	JMP	DCR2	;
1172'	3E 02	MVI	A,02H	;faz EFA=10B
1174'	32 7704'	STA	EFA	;
1177'	C3 11DC'	JMP	DESV6	;retorna
117A'	3E 02	MVI	A,02H	;faz EFN=10B
117C'	32 7705'	STA	EFN	;
117F'	C3 1172'	JMP	DESV5	;retorna indicando erro p/ EFA e EFN
1182'	3A 7704'	LDA	EFA	;le sinalizador de erro modulo anterior
1185'	E6 03	ANI	03H	;deixa so sinalizador
1187'	32 7704'	STA	EFA	;
118A'	3A 7703'	LDA	EFF	;forma palavra de reconhecimento para
118D'	C6 C0	ADI	0C0H	;modulo anterior
118F'	D3 08	OUT	PA552	;envia ACKA=0
1191'	DB 09	IN	FB552	;vai ver se modulo anterior enviou
				;reconhecimento
1193'	E6 80	ANI	80H	;mascara para RCKA
1195'	FE 00	CPI	00H	;RCKA=0?
1197'	CA 11BD'	JZ	DESV7	;
119A'	DB 09	IN	FB552	;vai ver se proximo modulo enviou
				;reconhecimento
119C'	E6 40	ANI	40H	;mascara para RCKN
119E'	FE 00	CPI	00H	;RCKN=0?
11A0'	C2 1191'	JNZ	DESV8	;
11A3'	01 FFFF	LXI	B,TESP	;
11A6'	DB 09	IN	FB552	;vai ver se proximo modulo enviou
11A8'	E6 80	ANI	80H	;mascara para RCKN
11AA'	FE 00	CPI	00H	;RCKN=0?
11AC'	CA 11DC'	JZ	DESV6	;
11AF'	3E 00	MVI	A,00H	;
11B1'	B8	CMF	B	;
11B2'	C2 11B9'	JNZ	DCRF3	;
11B5'	B9	CMF	C	;
11B6'	CA 1172'	JZ	DESV5	;sai indicando erro
11B9'	0B	DCX	B	;
11BA'	C3 11A6'	JMP	DCR3	;
11BD'	01 FFFF	LXI	B,TESP	;
11C0'	DB 09	IN	FB552	;vai ver se RCKN=0?

```

11C2' E6 40      ANI 40H      ;mascara para RCKN
11C4' FE 00      CPI 00H      ;
11C6' CA 11DC'   JZ  DESV6    ;RETORNA COM TUDO OK
11C9' 3E 00      MVI A,00H    ;
11CB' B8        CMP B        ;
11CC' C2 11D3'   JNZ DCRP4    ;
11CF' B9        CMP C        ;
11D0' CA 11D7'   JZ  DESVA    ;retorna indicando erro
11D3' 0B        DCX B        ;
11D4' C3 11C0'   JMP DCR4     ;
11D7' 3E 02      DESVA: MVI A,02H ;
11D9' 32 7705'   STA EFN      ;
11DC' C9        RET          ;sai da subrotina
11DD' 3A 7704'   DESV1: LDA EFA ;le o sinalizador do modulo anterior
11E0' E6 03      ANI 03H      ;deixa so o sinalizador
11E2' 32 7704'   STA EFA      ;
11E5' 3A 7703'   LDA EFP      ;preparando a palavra de reconhecimento
11E8' C6 D0      ADI 0D0H     ;ACKA=0
11EA' D3 08      OUT FA552    ;
11EC' 01 FFFF   LXI H,TESP    ;
11EF' DB 09      DCR5: IN FB552 ;testa se o proximo modulo enviou o
11F1' 32 7705'   STA EFN      ;sinalizador de erro
11F4' E6 10      ANI 10H      ;mascara para RFN
11F6' FE 00      CPI 00H      ;
11F8' CA 1209'   JZ  DESVB    ;
11FB' 3E 00      MVI A,00H    ;
11FD' B8        CMP B        ;
11FE' C2 1205'   JNZ DCRP5    ;
1201' B9        CMP C        ;
1202' CA 11D7'   JZ  DESVA    ;sai indicando erro
1205' 0B        DCX B        ;
1206' C3 11EF'   JMP DCR5     ;
1209' 3A 7705'   DESVB: LDA EFN ;le o sinalizador do proximo modulo
120C' E6 03      ANI 03H      ;deixa so o sinalizador
120E' 32 7705'   STA EFN      ;
1211' 3A 7703'   LDA EFP      ;preparando a palavra de reconhecimento
1214' C6 C0      ADI 0C0H     ;ACKN=0
1216' D3 08      OUT FA552    ;
1218' C3 1191'   JMP DESVB    ;
;
;
;MTB ROTINA MONTA TABELA DE DECISAO
;
;
;
121R' DB 0A      MTB: IN FC552 ;le a posicao no anel

```

121D'	E6 C0		ANI	0C0H	
121F'	32 7708'		STA	F0SMOD	Imascara para bits de posicao
1222'	B7		DRA	A	!guarda esta posicao
1223'	FE 80		CPI	MD_B	!testa posicao B
1225'	CA 1324'		JZ	LBLB	
1228'	DA 130F'		JC	LRLA	!neste caso e modulo A
122B'	3A 7705'		LDA	EFN	!neste caso e modulo C
122E'	32 7709'		STA	EA1	
1231'	3A 7704'		LDA	EFA	
1234'	32 770A'		STA	EB1	
1237'	3A 7703'		LDA	EFF	
123A'	32 770B'		STA	EC1	
123D'	3E 01	ENC:	MVI	A,01H	
123F'	21 7709'		LXI	H,EA1	
1242'	BE		CMF	H	
1243'	CA 1264'		JZ	DV1	
1246'	3E 02		MVI	A,02H	
1248'	21 7709'		LXI	H,EA1	
124B'	BE		CMF	H	
124C'	CA 125C'		JZ	DV2	
124F'	3E 01		MVI	A,01H	
1251'	32 770C'		STA	FTA	!FTA=1 falha na transmissao
1254'	3E 00	DV3:	MVI	A,00	
1256'	32 7709'		STA	EA1	
1259'	C3 1264'		JMP	DV1	
125C'	3E 01	DV2:	MVI	A,01H	
125E'	32 770F'		STA	FCA	!FCA=1 falha na comparacao
1261'	C3 1254'		JMP	DV3	
1264'	3E 01	DV1:	MVI	A,01H	
1266'	21 770A'		LXI	H,EB1	
1269'	BE		CMF	H	
126A'	CA 128B'		JZ	DX1	
126D'	3E 02		MVI	A,02H	
126F'	21 770A'		LXI	H,EB1	
1272'	BE		CMF	H	
1273'	CA 1283'		JZ	DX2	
1276'	3E 01		MVI	A,01H	
1278'	32 770D'		STA	FTB	
127B'	3E 00	DX3:	MVI	A,00H	
127D'	32 770A'		STA	EB1	
1280'	C3 128B'		JMP	DX1	
1283'	3E 01	DX2:	MVI	A,01H	
1285'	32 7710'		STA	FCB	
1288'	C3 127B'		JMP	DX3	
128B'	3E 01	DX1:	MVI	A,01H	
128D'	21 770B'		LXI	H,EC1	

1290'	BE		CMP	H	
1291'	CA 12B2'		JZ	DZ1	
1294'	3E 02		MVI	A,02H	
1296'	21 770B'		LXI	H,EC1	
1299'	BE		CMP	H	
129A'	CA 12AA'		JZ	DZ2	
129D'	3E 01		MVI	A,01	
129F'	32 770E'		STA	FIC	
12A2'	3E 00	DZ3:	MVI	A,00H	
12A4'	32 770B'		STA	EC1	
12A7'	C3 12B2'		JMP	DZ1	
12AA'	3E 01	DZ2:	MVI	A,01H	
12AC'	32 7711'		STA	FCC	
12AF'	C3 12A2'		JMP	DZ3	
12B2'	3A 770B'	DZ1:	LDA	EC1	
12B5'	4F		MOV	C,A	guarda EC1 em C
12B6'	3A 770A'		LDA	EB1	
12B9'	07		RLC		roda B a esquerda
12BA'	47		MOV	B,A	
12BB'	3A 7709'		LDA	EA1	
12BE'	07		RLC		
12BF'	07		RLC		
12C0'	B0		ORA	B	
12C1'	B1		ORA	C	
12C2'	32 7713'		STA	LINTAB	monta linha da tabela
12C5'	21 7712'		LXI	H,HVID	
12C8'	36 FF		MVI	H,0FFH	palavra para habilitar saida e indi-
12CA'	FE 04		CFI	04H	car modulo falho
12CC'	CA 12F7'		JZ	LBLSA	
12CF'	FE 06		CFI	06H	
12D1'	CA 12F7'		JZ	LBLSA	
12D4'	36 FD		MVI	H,0FDH	palavra para habilitar saida e indi-
12D6'	FE 07		CFI	07	car modulo falho
12D8'	CA 12F7'		JZ	LBLSA	
12DR'	36 FF		MVI	H,0FFH	
12DD'	FE 02		CFI	02H	IDEM
12DF'	CA 1305'		JZ	LBLSB	
12E2'	FE 03		CFI	03H	
12E4'	CA 1305'		JZ	LBLSB	
12E7'	FE 01		CFI	01H	
12E9'	CA 130A'		JZ	LBLSB	
12EC'	FE 05		CFI	05H	
12EE'	CA 130A'		JZ	LBLSB	
12F1'	3E 00	PERD:	MVI	A,00	
12F3'	32 7706'	GANHADOR:	STA	VENCEDOR	
12F6'	C9		RET		

```
12F7' 3E 00          LBLSA:      MVI    A,MD_A      ISAI A
12F9' 21 7708'       POSIC:      LXI    H,POSMOD
12FC' BE             CMP      M          ;testa POSMOD=SAI
12FD' C2 12F1'       JNZ      PERD        ;PERDEDOR
1300' 3E 01          MVI      A,01
1302' C3 12F3'       JMP      GANHADOR    ;VENCEDOR
1305' 3E 80          LBLSB:      MVI      A,MD_B      ISAI B
1307' C3 12F9'       JMP      POSIC
130A' 3E C0          LBLSC:      MVI      A,MD_C      ISAI C
130C' C3 12F9'       JMP      POSIC
130F' 3A 7703'       LBLA:      LDA      EFP          ;e modulo A
1312' 32 7709'       STA      EA1
1315' 3A 7705'       LDA      EFN
1318' 32 770A'       STA      EB1
131B' 3A 7704'       LDA      EFA
131E' 32 770B'       STA      EC1
1321' C3 123D'       JMP      ENC
1324' 3A 7704'       LBLB:      LDA      EFA          ;e modulo B
1327' 32 7709'       STA      EA1
132A' 3A 7703'       LDA      EFP
132D' 32 770A'       STA      EB1
1330' 3A 7705'       LDA      EFN
1333' 32 770B'       STA      EC1
1336' C3 123D'       JMP      ENC
                        END
```

Macros:

Symbols:

CRTS	106F'	DAAT	7701'	DANT	7702'	DCR1	1120'
DCR2	1158'	DCR3	11A6'	DCR4	11C0'	DCR5	11EF'
DCRC1	10B1'	DCRC2	10E1'	DCRC3	10FB'	DCRF2	116E'
DCRF3	11B9'	DCRF4	11D3'	DCRF5	1205'	DESV1	11DD'
DESV2	1146'	DESV3	117A'	DESV4	1182'	DESV5	1172'
DESV6	11DC'	DESV7	11BD'	DESV8	1191'	DESV9	11D7'
DESVB	1209'	DV1	1264'	DV2	125C'	DV3	1254'
DVID	00FC	DX1	128B'	DX2	1283'	DX3	127B'
DZ1	12B2'	DZ2	12AA'	DZ3	12A2'	EA1	7709'
EB1	770A'	EC1	770B'	EC551	001F'	EC552	000B'
EFA	7704'	EFN	7705'	EFF	7703'	ENC	123D'
EVID	000A'	FCA	770F'	FCB	7710'	FCC	7711'
FTA	770C'	FTB	770D'	FTC	770E'	GANHAD	12F3'
HVID	7712'	IGUAL	106C'	LBL1	109E'	LBL2	10CE'
LRL3	10E8'	LRLA	130F'	LRLB	1324'	LRLSA	12F7'
LRLSB	1305'	LRLSC	130A'	LEDAD1	10C2'	LEDAD2	110A'
LINTAB	7713'	MD_A	0000	MD_B	0080	MD_C	00C0
MTB	121B'	PA551	001C	PA552	000B	PR551	001D
PR552	0009	PC551	001E	PC552	000A	PERD	12F1'
PERDED	1080'	POSIC	12F9'	POSMOD	7708'	PRDOR	1045'
PRTN	1109'	RECDAD	10E5'	RTD	1084'	RTS	1116'
RVS	104C'	SAIDA	7700'	SCER	00FC	SEQ1	10B5'
SEQ2	10CB'	SEQ3	10FF'	ST551	009B	ST552	008A
TESP	FFFF	T_E	7707'	VENCED	7706'	VHSH	00FF

No Fatal error(s)

B>