FUNDAÇÃO HERMÍNIO OMETTO

CENTRO UNIVERSITÁRIO HERMÍNIO OMETTO - FHO

Av. Maximiliano Baruto, 500 CEP 13607-339 Araras SP

Tel (19) 3543 -1439 Fax (19) 3543 -1440

Aluno: Alexandro Souza RA: 93871

Aluno: André Carneiro RA: 92854

Aluno: Elias A. da Silva RA: 92756

Aluno: Gabriel Resende RA: 94038

Aluno: Gabriel Seratti RA: 88156

Aluno: Jonas Sbarai RA: 93967

MA1 – Desenvolvimento de unidade de injeção de falhas

ARARAS/SP

10/2020

CENTRO UNIVERSITÁRIO HERMÍNIO OMETTO - FHO
Av. Maximiliano Baruto, 500 CEP 13607-339 Araras SP
Tel (19) 3543 -1439 Fax (19) 3543 -1440

Sumário

OBS: destaque para referência "[24] SANTOS, Ana Carla dos Oliveira. Tolerância a falhas para sistemas embarcados. 2000. Disponível em: http://www.cin.ufpe.br/~tg/2000-1/acos.doc. Acesso em: 04 out. 2020.", pois, excepcionalmente o trabalho apresentado por essa autora foi extremamente essencial na elaboração do trabalho, não teve como fugir do que a autora propôs, o contexto dela e a forma que ela colocou se encaixam perfeitamente no trabalho proposto.

1. Modelo para aplicação de tolerância a falhas em sistemas embarcados

Para esta seção vamos somente lidar com o modelo proposto, visto que o estudo de caso e o tratamento de falhas será visto nas seções a seguir.

1.1. Modelo proposto

O modelo proposto na Figura 1, ilustra as fases do projeto de sistemas embarcados considerando a introdução de mecanismos de tolerância a falhas.

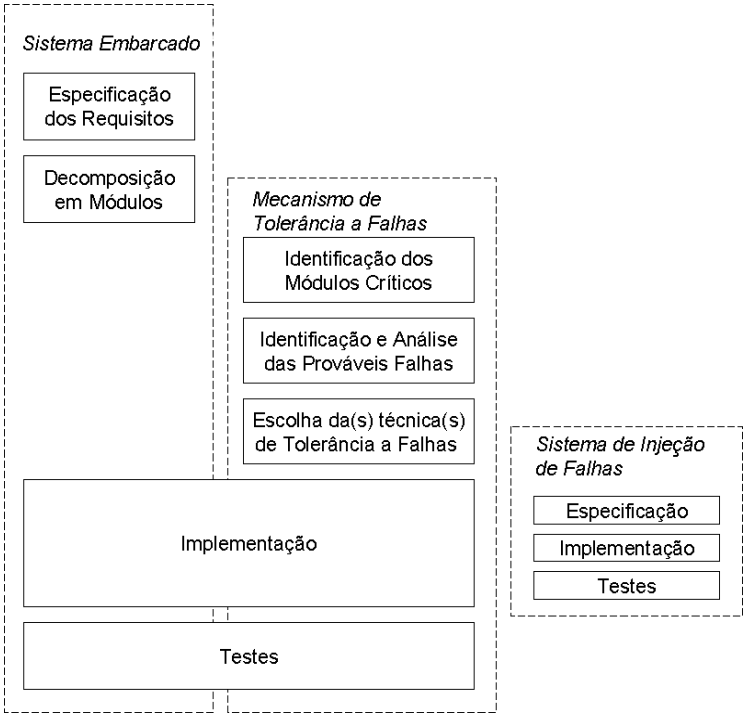

Figura 1 - Modelo para aplicação de Tolerância a Falhas a Sistemas Embarcados.

O modelo apresenta três sistemas distintos: o sistema embarcado em si, o mecanismo de tolerância a falhas e o sistema de injeção de falhas. Após identificar os módulos do sistema, deve-se fazer a identificação dos módulos críticos, sendo essa uma fase exclusiva do mecanismo de tolerância a falhas, assim como as duas fases seguintes. Escolhidas as técnicas

que serão aplicadas ao sistema com o objetivo de tolerar às falhas, parte-se para a modelagem e implementação do sistema embarcado e do mecanismo de tolerância a falhas que devem ser feitos não só em paralelo, mas em conjunto já que o mecanismo de tolerância a falhas estará inserido no sistema embarcado. O sistemas de injeção de falhas, poderá ser desenvolvido paralelamente ao sistema embarcado, para conferir a característica desejável de interferência mínima no sistema destino.

2. Estudo de caso

Para esta sessão iremos abordar a aplicação-alvo escolhida, considerando seus aspectos de falha. Sendo assim iremos enfatizar seu sistema de tolerância a falhas, onde este será o foco desejado para andamento ao trabalho.

A aplicação adotada será o caso de um par de semáforos de um cruzamento de trânsito, faz-se necessário um sistema de tolerância a falhas, já que o mesmo não pode sinalizar para que ambos os lados cruzem ao mesmo tempo.

2.1. Aplicação do modelo

Seção dedicada a esmiuçar as fases do proposto em questão.

2.1.1. Especificação de requisitos

O sistema deve controlar os semáforos do cruzamento de modo que o sinal verde dure 5 segundos e o sinal amarelo dure 1 segundo. O sistema nunca deve deixar que os dois sinais fiquem verdes e deve possuir alguma forma de mostrar que ocorreu falha em seu processamento. No caso da ocorrência de falhas, o sistema deve mostrar sinal vermelho ou amarelo para os dois cruzamentos, de forma que os motoristas percebam que o sistema de semáforos está inativo.

2.2. Decomposição em módulos e identificação dos módulos críticos

O sistema será representado por um único módulo apenas e este módulo será considerado crítico e sobre ele serão aplicadas as técnicas de tolerância a falhas. Pois é verificado que não há necessidade de mais módulos além disso.

2.2.1. Identificação e análise de prováveis falhas

Há algumas falhas prováveis no sistema proposto. A temporização do sistema pode ser alterada devido à ocorrência de alguma falha. O sistema pode gerar saídas erradas, por conta de erros no projeto ou alguma interferência externa pode modificar o valor de algum bit. Além disso, o sistema pode simplesmente parar seu funcionamento, devido a desgaste físico do material, por exemplo, deixando de produzir as saídas.

A frequência de ocorrência das falhas não pode ser identificada imediatamente. Apenas com a utilização do sistema por algum tempo, poderá ser extraída essa informação.

2.2.2. Escolha das técnicas de tolerância a falhas

O grau de confiabilidade exigido na especificação da aplicação indica somente que o sistema deve ser capaz de avisar a ocorrência de falhas em seu processamento, apresentando um comportamento pré-definido que lhe garanta a segurança. Assim, a aplicação de uma técnica para detecção de falhas é suficiente. Neste estudo de caso, duas técnicas serão implementadas: verificação de consistência e duplicação com comparação. Cada técnica será aplicada em uma abordagem diferente ao problema.

2.2.3. Implementação

Verificação da Consistência

O sistema controla os dois semáforos enviando três bits para cada um deles. Cada bit indica uma luz do semáforo: verde, vermelha e amarela. Um dos invariantes desse sistema é que as duas luzes verdes nunca podem estar acesas ao mesmo tempo. O mecanismo de detecção de falhas apenas verifica se os bits que indicam a luz verde nos dois semáforos estão ativos simultaneamente. Se estiverem, o mecanismo de detecção de erros modifica a saída deixando apenas as luzes amarelas acesas e envia um sinal indicando que houve erro. O sistema continua funcionando até que ocorra outro erro.

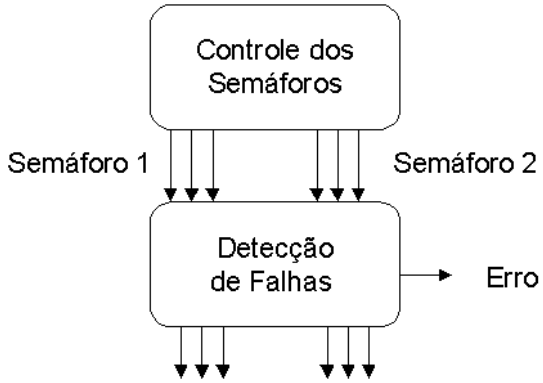

Figura 2 - Estudo de caso: Verificação de Consistência.

Duplicação com comparação

O sistema controla os dois semáforos enviando dois bits para cada um deles. Cada bit indica uma luz do semáforo: verde e vermelha. Entre uma luz verde e outra, o sistema mantêm os dois sinais vermelhos acesos por 1 segundo, para manter a segurança e compensar a falta do sinal amarelo. Para a introdução do mecanismo de tolerância a falhas, foram utilizados dois módulos idênticos do sistema e um componente para realizar a comparação das saídas e detectar erros na execução.

Para essa abordagem se faz necessário controle de sincronismo entre os dois módulos, para garantir que ambos encontram-se no mesmo estado da execução do programa.

Para isso, são acrescentados um flag para cada módulo redundante que indica que um novo dado foi enviado para o módulo de detecção de falhas. Recebendo esse sinal, o módulo de detecção de falhas aguarda a sinalização do outro módulo. Se o tempo para o segundo módulo sinalizar que modificou seus sinais de saída exceder um timeout, o sistema considerará que houve falha neste módulo e deve retornar as saídas que o módulo que sinalizou primeiro enviou. Isso é feito porque considera-se que a probabilidade de ocorrer duas falhas simultaneamente nos dois módulos é muito pequena e se um dos módulos falhou, provavelmente o outro deve estar correto. Ainda assim, o sistema deve sinalizar que houve erro, indicando que aquele resultado foi obtido apenas de um dos módulos e, portanto, possui credibilidade menor que os resultados obtidos da execução dos dois módulos, comparação entre eles e constatação de que as duas saídas estão iguais.

Se não houver timeout, o sistema compara as duas saídas. No caso de discordância entre os módulos, o mecanismo de detecção de falhas não tem condições de descobrir qual dos dois módulos falhou para retornar a saída do outro módulo. Então, atendendo os requisitos de segurança, o sistema coloca na saída os sinais que indicam que as duas luzes vermelhas dos dois semáforos devem ficar acesas.

No caso de não ocorrer falha alguma, o módulo de detecção de falhas retorna as saídas de um dos módulos, já que estas são iguais.

Sempre após retornar uma saída, o módulo de detecção de erros, através do sinal ‘Continue’, sinaliza aos dois microcontroladores que eles podem continuar a execução do programa. Assim, a cada mudança na saída, os módulos redundantes são sincronizados.

Os programas carregados nos dois microcontroladores são idênticos, mas poderiam ter sido projetados independentemente, utilizando a técnica de N-versões.

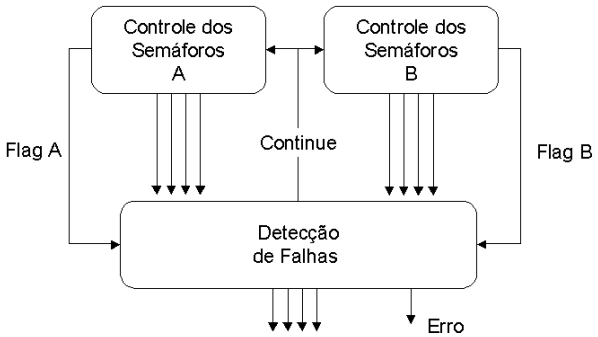

Figura 3 - Estudo de caso: Duplicação com Comparação.

3. Classificação de técnicas de tolerância a falhas

Nesta seção abordaremos com mais detalhes as técnicas escolhidas para tolerância de falhas e será apresentada mais uma extra na qual poderíamos adotar ao sistema também.

3.1. Técnicas analisadas

3.1.1. Verificação de consistência (Consistency Checks)

Utilizando o conhecimento de características inerentes ao sistema (invariantes), essa técnica consiste em realizar verificações em determinados pontos da computação, testando a consistência, ou seja, se os invariantes continuam sendo respeitados. Como exemplo de invariante podemos citar o caso de uma variável sempre conter um valor que pertença a um determinado intervalo. Pode-se testar se o valor dessa variável está no intervalo esperado[1].

Tipo de Redundância Utilizada

As verificações podem ser feitas tanto por hardware quanto por software.

Meio de Implementação

A técnica pode ser implementada por hardware, porém a implementação por software é bem mais utilizada. Um exemplo de verificação de consistência que pode ser feito por hardware é a detecção de código de instrução inválido.

Objetivo

Essa técnica é utilizada para a detecção de erros no sistema.

Aspectos Gerais

A técnica é aplicada a sistemas que possuam características imutáveis durante sua execução. Nem sempre é natural encontrar um invariante em um sistema computacional. A identificação de um falso invariante e a utilização dessa técnica podem ocasionar a detecção de falhas inexistentes. Identificado o invariante, a técnica pode ser facilmente implementada, não exigindo um grau elevado de redundância.

Aspectos da Aplicação em Sistemas Embarcados

Em sistemas embarcados, essa técnica é utilizada pelo uso das interrupções de violação de acesso, falhas no oscilador, variação na saída da fonte de energia, dentre outras. Além disso, se as saídas do sistema apresentam alguma regra de formação, pode-se implementar um componente que verifique se cada saída obedece a regra e assim detectar saídas inválidas para o sistema.

3.1.2. Duplicação com comparação (Duplication with comparison)

Também conhecida como 2MR, essa técnica consiste em uma variação da NMR utilizando apenas dois módulos replicados. Pelo fato de receber apenas duas entradas, o mecanismo de voto majoritário nesse caso se resume a um comparador.

Abaixo serão descritos aspectos específicos dessa variação do NMR.

Objetivo

Essa técnica apenas permite a detecção de erros no sistema. No caso da utilização de redundância de tempo para a implementação desta técnica, apenas falhas temporárias serão detectadas, já que o hardware utilizado para a computação será o mesmo, sendo variado

apenas o instante de tempo em que a computação será realizada. Para utilizar essa abordagem na detecção de falhas permanentes, pode-se utilizar a técnica Alternating Logic, que consiste na utilização de codificação da entrada e decodificação da saída em uma das vezes em que a computação é realizada, como ilustrado na Figura 3. Desse modo, além de detectar as falhas temporárias, será possível detectar algumas falhas permanentes no componente utilizado. Essa técnica, porém, requer tempo e componentes adicionais para a codificação e decodificação.

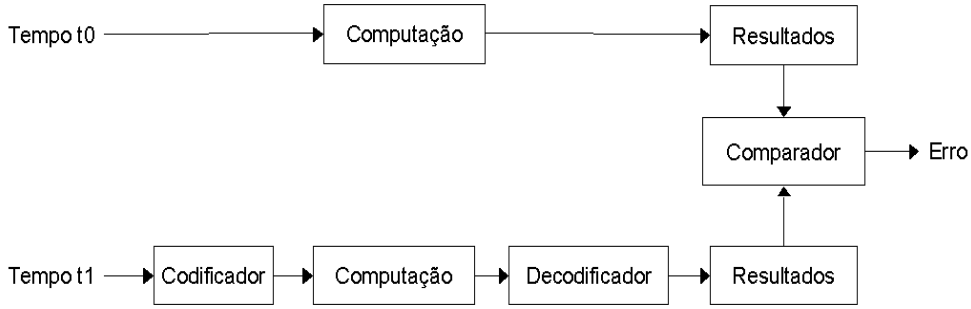

Figura 4 - Alternating Logic.

Aspectos Gerais

A técnica deve ser utilizada em aplicações que não exijam muita disponibilidade, sendo a computação abortada na ocorrência de um erro, ou deve existir algum método auxiliar para detectar qual dos módulos falhou e tratar o problema.

Aspectos da Aplicação em Sistemas Embarcados

A implementação dessa técnica é simples e pode ser aplicada facilmente a sistemas embarcados. Um mecanismo de comparação de saídas é requerido e os módulos redundantes devem estar sincronizados para que o componente detector de falhas não indique falhas inexistentes devido a atraso na resposta de um dos módulos.

3.1.3. NMR

A técnica NMR (n-modular redundancy) consiste na utilização de n módulos, com mesma funcionalidade, realizando a mesma computação e utilizando-se um mecanismo de voto majoritário para a escolha da resposta correta. Essa técnica considera que a probabilidade de mais de um módulo apresentar falhas durante a computação é muito pequena e desse modo a confiabilidade do sistema seria aumentada. Essa consideração exige que os módulos sejam independentes no que diz respeito às falhas.

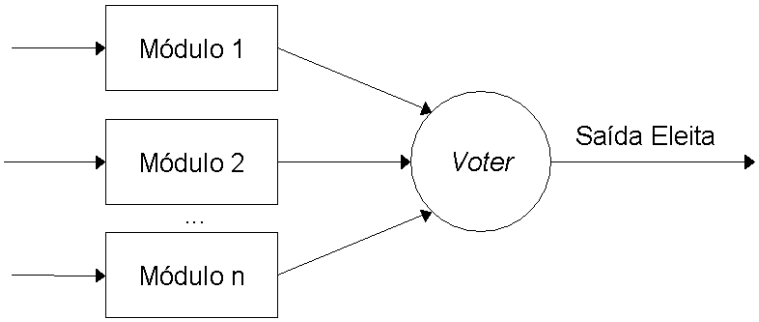

Figura 5 - Técnica NMR (N-Modular edundancy).

Tipo de Redundância Utilizada

Essa técnica pode utilizar dois tipos de redundância:

- Hardware - no caso de replicação de componentes físicos, como unidades lógica-aritméticas, chips de memória...
- Tempo - no caso de utilizar um mesmo componente, porém realizar o processamento n vezes em instantes distintos.

Há ainda a replicação de módulos em software, porém essa técnica é conhecida como Programação em N-Versões e será discutida separadamente.

Meio de Implementação

A implementação pode ser feita por hardware, se os módulos duplicados e o comparador forem componentes de hardware; ou híbrida, no caso de os módulos duplicados serem implementados por software e a comparação ser feita por hardware.

Objetivo

A técnica permite a detecção e o mascaramento de falhas do sistema, dependendo do número de módulos utilizados.

Aspectos Gerais

Essa técnica enfrenta o problema do ponto único de falha (single-point-of-failure), representado nesse caso pelo mecanismo de votação. A confiabilidade do sistema como um todo estará limitada à confiabilidade do componente votante, como mostrado pela fórmula abaixo[1].

R(t) = [1 - p1(t) . p2(t) . ... . pn(t)] . [1 - pv(t)],

onde pn(t) é a probabilidade do módulo n falhar no instante t , pv(t) é a probabilidade do votante falhar no instante t e R(t) é a função de confiabilidade do sistema.

Portanto, só é conveniente a utilização da técnica se a confiabilidade do votante for maior que a confiabilidade apresentada pelos módulos separadamente.

A técnica pode ser bem aplicada em um grande número de aplicações sendo provavelmente a mais utilizada das técnicas de tolerância a falhas.

Aspectos da Aplicação em Sistemas Embarcados

Para a aplicação em sistemas embarcados, essa técnica pode ser bem utilizada. Os recursos adicionais requeridos são o componente para voto por maioria, que, em algumas variações dessa técnica, pode apresentar–se com diferentes funcionalidades desde um simples comparador a um componente mais sofisticado de ordenação de saídas, como veremos mais adiante. Além disso, um método para sincronização entre os módulos redundantes deve ser utilizado.

4. Referências

[1] Pradhan, D. K.; FAULT-TOLERANT COMPUTER SYSTEM DESIGN. Prentice Hall 1996

[2] Jansch-Pôrto, I.; Weber, T. S. RECUPERAÇÃO EM SISTEMAS DISTRIBUÍDOS. XVI Jornada de Atualização em Informática, XVII Congresso da Sociedade Brasileira de Computação. 1997

[3] Cechin, S. L.; MODELAGEM DE FALHAS. Instituto de Informática - UFRGS. 1997.

[4] GRUPO DE TOLERÂNCIA A FALHAS DA UFRGS. URL: http://www.inf.ufrgs.br/gpesquisa/tf/tolerancia.html

[5] Piccoli, L.; CHECKPOINTS SÍNCRONOS URL: http://www.inf.ufrgs.br/gpesquisa/tf/portugues/ensino/ckpt.html

[6] Vahid, F.; Givargis, T. EMBEDDED SYSTEM DESIGN: A UNIFIED HARDWARE/SOFTWARE APPROACH Department os computer Science and Engineering – University of Califórnia, 1999.

[7] Silveira, E. D.; MICROCONTROLADORES E AUTOMAÇÃO: PANORAMA ATUAL E PERSPECTIVAS FUTURAS. URL: http://www.malbanet.com.br/professorelmo/microcon.htm

[8] EMBEDDED SYSTEMS PROGRAMMING. URL: http://www.embedded.com/

[9] Clark. J. A.; Pradhan. D. K.; FAULT-INJECTION: A METHOD FOR VALIDATING COMPUTER-SYSTEM DEPENDABILITY . IEEE Computer, Vol. 28, No. 6, June 1995 – Abstract.

[10] Jenn, E; Arlat, J; Rimen, M.; Ohlsson, J.; Karlsson, J. FAULT INJECTION INTO VHDL MODELS: THE MEFISTO TOOL. Twenty-Fourth International Symposium on Fault-Tolerant Computing. IEEE, 1994. p. 66-75.

[11] Reliable Software Technologies, FAULT INJECTION SOFTWARE TOOL. URL: http://www.rstcorp.com/FIST-demo/intro.html

[12] J. Karlsson et al. USING HEAVY-ION RADIATION TO VALIDATE FAULT-HANDLING MECHANISMS. IEEE Micro, v.14, n.1, p.8-23, Feb. 1994.

[13] Xess Corporation. XS40, XSP, XS95 BOARD USER MANUAL. 1997-1999 URL: http://www.xess.com

[14] Brown, S. D.; Francis, R.J; Rose, J; Vranesic, Z. G. FIELD-PROGRAMMABLE GATE ARRAYS. Kluwer Academic Publishers, 1992.

[15] KEILSOFTWARE – EMBEDDED DEVELOPMENT TOOLS. URL: http://www.keil.com

[16] XILINX. URL: http://www.xilinx.com/

[17] Avizienis, A. THE FOUR-UNIVERSE INFORMATION SYSTEM MODEL FOR STUDY OF FAULT TOLERANCE, Proceedings of the 24th Annual International Symposium on Fault-Tolerant Computing Santa Monica, Califórnia 1982.

[18] Inacio, C. SOFTWARE FAULT TOLERANCE. URL: http://www.cs.cmu.edu/~koopman/des_s99/sw_fault_tolerance/index.html

[19] INTEL HEX FORMAT. URL: http://www.8052.com/tutintel.htm

[20] Stallings, W. COMPUTER ORGANIZATION AND ARCHITECTURE: DESIGNING FOR PERFORMANCE. 4th Edition - Prentice Hall 1996

[21] Avizeinis, A. THE N-VERSION APPROACH TO FAULT-TOLERANT SOFTWARE, IEEE Transactions of Software Engineering, Vol. SE-11, No. 12 (December 1985), pp. 1491-1501.

[22] Slater, R. FAULT INJECTION. URL: http://www.cs.cmu.edu/~koopman/des_s99/fault_injection/index.html

[23] Sotoma, I. AFIDS - ARQUITETURA PARA INJEÇÃO DE FALHAS EM SISTEMAS DISTRIBUÍDOS. CPGCC, UFRGS, 1997

[24] SANTOS, Ana Carla dos Oliveira. Tolerância a falhas para sistemas embarcados. 2000. Disponível em: http://www.cin.ufpe.br/~tg/2000-1/acos.doc. Acesso em: 04 out. 2020.