UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Felipe Augusto da Silva

CONCEPÇÃO E VALIDAÇÃO DE ARQUITETURA ROBUSTA BASEADA EM SOFT PROCESSORS PARA USO EM COMPUTADORES DE BORDO DE SATÉLITES ARTIFICIAIS

Dissertação submetida ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Engenharia Elétrica.
Orientador: Prof. Dr. Eduardo Augusto Bezerra

Florianópolis
2013

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

Silva, Felipe Augusto da
Concepção e validação de arquitetura robusta baseada em soft processors para uso em computadores de bordo de satélites artificiais / Felipe Augusto da Silva ; orientador, Eduardo Augusto Bezerra - Florianópolis, SC, 2013.
103 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Elétrica.

Inclui referências

1. Engenharia Elétrica. 2. Computador de bordo para satélites artificiais. 3. Tolerância a falhas em FPGAs. I. Bezerra, Eduardo Augusto. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

Felipe Augusto da Silva

CONCEPÇÃO E VALIDAÇÃO DE ARQUITETURA ROBUSTA BASEADA EM SOFT PROCESSORS PARA USO EM COMPUTADORES DE BORDO DE SATÉLITES ARTIFICIAIS

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Engenharia Elétrica", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica.

Florianópolis, 25 de setembro de 2013.

_____
Prof. Dr.Patrick Kuo Peng
Coordenador do Curso

Banca Examinadora:

_____
Prof. Dr. Eduardo Augusto Bezerra
Orientador
Universidade Federal de Santa Catarina

_____
Prof. Dr. Raimes Moraes
Universidade Federal de Santa Catarina

_____
Prof. Dr. Djones Vinicius Lettnin
Universidade Federal de Santa Catarina

_____
Prof. Dr. Fabian Luis Vargas
Pontifícia Universidade Católica do Rio Grande do Sul

Esse trabalho é dedicado aos meus Pais que fizeram tudo que estava ao seu alcance para que eu chegasse até aqui.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha família; ao meu Pai que sempre se empenhou para prover as melhores condições para que eu me dedicasse aos estudos, e sempre cobrou que eu fizesse o mesmo para alcançar meus objetivos; a minha Mãe que sempre me apoiou e amou incondicionalmente; ao meu Irmão que sempre esteve comigo em todos os momentos difíceis e de conquista; aos meus Tios e Tias que sempre me trataram com um filho; e aos meus Primos que sempre foram meus melhores amigos, meus irmãos. Sem o apoio e o amor de vocês eu nunca teria alcançado nada na vida.

Gostaria também de agradecer a todos os amigos que estiveram presentes nos muitos anos de minha caminhada acadêmica, que teve inicio no curso de Engenharia na PUCRS em 2004 e agora chega ao final de mais uma etapa com a conclusão do curso de Mestrado na UFSC em 2013. Agradeço especialmente aos amigos que fiz no GAPH e no GSE, a família do Charrua Rugby Clube, aos amigos que estiveram comigo na Irlanda, aos que estiveram comigo em Florianópolis nos anos do mestrado e aos amigos que fiz nos anos morando em Israel. Todos tiveram uma parcela de contribuição importante na minha formação profissional e pessoal. Tenho certeza que algumas das amizades que cultivei nesses anos estarão presentes em minha vida para sempre.

Agradeço ao Prof. Dr. Eduardo Augusto Bezerra, primeiramente pelo convite para realizar o Mestrado na UFSC, mas principalmente por todo o apoio, paciência e a confiança depositada em mim nos últimos sete anos.

Muito obrigado a todos vocês.

"The important thing is not to stop questioning; curiosity has its own reason for existing."
(Albert Einstein, 1955)

RESUMO

A flexibilidade introduzida pela utilização de FPGAs (*Field Programmable Gate Array*) SRAM comerciais em aplicações embarcadas, faz com que esta tecnologia se torne uma alternativa atraente para aplicações militares e espaciais. No presente trabalho, foi desenvolvido um Computador de Bordo utilizando *soft processor* embarcado em um FPGA do tipo SRAM. O Computador de Bordo é baseado em requisitos funcionais especificados pelo Instituto Nacional de Pesquisas Espaciais (INPE) para o Computador de Bordo a ser utilizado em suas futuras missões. Módulos de *software* e *hardware* foram implementados visando executar as principais funcionalidades de um Computador de Bordo.

No entanto, os avanços oriundos de tecnologias nanométricas trazem uma maior vulnerabilidade dos componentes eletrônicos a efeitos de radiação. Em aplicações críticas é importante que técnicas de tolerância a falhas sejam utilizadas para aumentar o grau de confiabilidade das aplicações.

Com o intuito de mitigar falhas causadas pela radiação a qual computadores de bordo são expostos no espaço, uma técnica de tolerância a falhas não intrusiva foi desenvolvida. A técnica proposta visa aplicar mecanismos de detecção de falhas utilizando um monitor de barramento para comparar os dados de saída de um *soft processor* principal com seu módulo redundante. Caso os dados sejam diferentes, um sinal de erro é gerado, iniciando a estratégia de tolerância a falhas.

A técnica proposta se mostrou eficiente quando comparada a técnicas do estado da arte como a Redundância Tripla (*Triple Modular Redundancy*, TMR) e Tolerância a Falhas em Hardware Implementadas em Software (*Software Implemented Hardware Fault Tolerance*, SIHFT) para identificação de falhas simples em tempo de execução com menor ocupação de área e sem alterar o desempenho da aplicação.

Palavras-chave: Computador de bordo; Tolerância a falhas; LEON3; Monitor de barramentos; *Single Event Upset*; Aplicações espaciais.

ABSTRACT

The flexibility introduced by Commercial Off The Shelf (COTS) SRAM based FPGAs in on-board system designs make them an attractive option for military and aerospace applications. However, the advances towards the nanometer technology come together with a higher vulnerability of integrated circuits to radiation perturbations. In mission critical applications it is important to improve the reliability of applications by using fault-tolerance techniques.

In this work, the concept of an On-Board Computer (OBC) system aiming a soft-processor embedded on a SRAM based FPGA is proposed. The OBC comply with functional requirements of the Brazilian Institute of Space Research (INPE) for the OBC that will be employed in future missions. Modules of software and hardware were implemented in order to execute the main capabilities of the OBC.

In order to mitigate the faults caused by radiation on the space environment, a non-intrusive fault tolerance technique has been developed. The proposed technique targets soft processors (e.g. LEON3), and its detection mechanism uses a Bus Monitor to compare output data of a main soft-processor with its redundant module. In case of a mismatch, an error signal is activated, triggering the proposed fault tolerance strategy. This approach shows to be more efficient than the state-of-the-art Triple Modular Redundancy (TMR) and Software Implemented Hardware Fault Tolerance (SIHFT) approaches in order to detect and to correct faults on the fly with low area overhead and with no major performance penalties.

Keywords: On-board computer; Fault tolerance; Fault injection; LEON3; Bus Monitor; Single Event Upset; Space Applications.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACDH	Attitude Control and Data Handling
AEB	Agencia Espacial Brasileira
AMBA	Advanced Microcontroller Bus Architecture
AOC	Attitude and Orbit Control
APB	Advanced Peripheral Bus
ASIC	Application-Specific Integrated Circuit
BCH	Bose, Chaudhuri and Hocquenghem
CAN	Controller Area Network
CCSDS	Consultative Committee for Space Data Systems
CMEM	Cache Memory
CNAE	Comissão Nacional de Atividades Espaciais
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CSC	Computer Software Components
CTA	Centro Tecnológico da Aeronáutica
ECSS	European Cooperation For Space Standardization
EDAC	Error Detection and Correction
ESA	European Space Agency
ESS	Electrical Support Equipament
FP	Falha Permanente
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPS	Global Positioning System
GRLIB	Gaisler Research Library
HDLC	High Level Data Link Control
IAE	Instituto de Aeronáutica e Espaço
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
INPE	Instituto Nacional de Pesquisas Espaciais
ITA	Instituto Tecnológico de Aeronáutica
MAC	Medium Access Control
MRO	Mars Reconnaissance Orbiter
NASA	National Aeronautics and Space Administration
OBC	On-Board Computer
OBDH	On-Board Data Handling
OBT	On-Board Time
PCDU	Power Conditioning and Distribution Unit
PPS	Pulsos por Segundo
RTEMS	Real Time Executive for Multiprocessor Systems

RTOS	Real Time Operational System
SAR	Synthetic Aperture Radar
SBSR	Simpósio Brasileiro de Sensoriamento Remoto
SEE	Single Event Effects
SEL	Single Event Latchup
SET	Single Event Transients
SEU	Single Event Upset
SIHFT	Software Implemented Hardware Fault Tolerance
SMP	Symmetric Multi-Processor
SPI	Serial Peripheral Interface
SRAM	Single Random Access Memory
SSTL	Surrey Satellite Technology Ltd
TBMEM	Trace Buffer Memory
TMR	Triple Modular Redundancy
TRIBUFF	3-state buffers
UTMC	Unidade de Telemetria e Telecomando
VGA	Video Graphics Array

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento da tecnologia espacial auxilia na busca por um melhor entendimento do nosso planeta, e também de soluções a serem utilizadas em problemas do cotidiano. Cada vez mais, a utilização de satélites vem sendo considerada um elemento essencial na sociedade moderna. A rotina atual da população depende diretamente dos recursos disponibilizados pelos satélites, como por exemplo, os sistemas de telefonia e de transmissão de imagens para televisão. Além disso, existe uma variedade de satélites em desenvolvimento com os mais diversos objetivos (ESA, 2012), como por exemplo:

- Satélites de Astronomia - para observação do espaço, galáxias, planetas distantes e outros objetos do espaço;

- Satélites Científicos - desenvolvidos para realizar experimentos científicos no ambiente espacial. Exemplos desses satélites incluem aqueles para estudo do Sol (radiação, vento solar, inversão de polaridade), experimentos em gravidade zero entre outros;

- Satélites de Comunicação - satélites estacionários para fins de telecomunicação;

- Satélites de Navegação - enviam sinais de tempo e localização para dispositivos móveis na Terra, para determinar a localização dos mesmos;

- Satélites de Observação da Terra - utilizados para monitoração de meio ambiente, meteorologia, criação de mapas entre outros;

- Satélites de uso militar.

Atualmente alguns satélites com tecnologia brasileira estão orbitando a Terra, existindo também, alguns projetos para futuros lançamentos (Epiphanio, 2013). Neste contexto, é importante salientar que esses equipamentos têm que ser desenvolvidos para funcionar pelo período completo de sua missão sem a necessidade de manutenção, sendo essa usualmente não viável devido aos custos envolvidos. O espaço é um ambiente hostil para componentes eletrônicos, sendo

necessária a utilização de componentes especiais tolerantes a radiação. Além disso, o projeto do sistema deve contemplar partes do circuito com alto grau de confiabilidade através da aplicação de técnicas de tolerância a falhas, permitindo que o sistema opere continuamente mesmo na existência de falhas.

Entre os componentes do satélite que exigem alto grau de confiabilidade pode-se ressaltar o Computador de Bordo (*On-Board Computer*, OBC). Tipicamente, o OBC de um satélite artificial é a unidade central de processamento do satélite. O OBC é responsável por funções essenciais para o funcionamento do satélite, como por exemplo, gerência da comunicação com a Terra e controle de atitude e órbita. Maiores detalhes sobre a estrutura de um satélite artificial e as funções executadas pelo OBC serão apresentadas na Seção 3.2. Considerando o nível de confiabilidade exigido, torna-se imprescindível que o OBC possa resistir aos diversos fatores externos que podem afetar seu funcionamento sem que este seja alterado, pois uma falha no OBC pode significar o fim da missão em questão. Na Seção 3.1, são apresentados os possíveis efeitos causados pela radiação a qual equipamentos eletrônicos estão expostos quando utilizados em ambientes hostis como o espaço.

Devido às características dos dispositivos FPGA e ao alto grau de confiabilidade necessário nesse tipo de equipamento, à arquitetura inicial do *soft processor* adotado como modelo de referência foi modificada visando a aplicação de uma técnica de tolerância a falhas. Com a utilização da técnica de tolerância a falhas, o OBC desenvolvido apresenta graus de confiabilidade próximos dos necessários para esse tipo de aplicação.

1.1 Objetivo Geral

O presente trabalho tem como objetivo principal, a concepção de uma arquitetura de computador de bordo confiável para utilização em satélites, utilizando *soft processors* embarcados em FPGAs (*Field Programmable Gate Array*). A principal contribuição tecnológica é a utilização do conceito de controle de fluxo por observação de barramentos para identificação de falhas simples em tempo de execução de maneira não intrusiva; ou seja, sem alterar o funcionamento do processador e sem modificações na aplicação sendo executada.

1.2 Objetivos Específicos

Durante o desenvolvimento da pesquisa buscou-se o aperfeiçoamento de conhecimentos adquiridos ao longo dos cursos de graduação e mestrado. Os objetivos específicos incluem:

- Compreender conceitos relacionados a sistemas embarcados para aplicações espaciais;
- Investigar os efeitos da radiação espacial em componentes eletrônicos;
- Conceber uma arquitetura de computador de bordo para uso em aplicações espaciais;
- Definir estratégias para melhoria da confiabilidade da arquitetura proposta;
- Realizar a validação da arquitetura proposta.

1.3 Organização do Texto

No Capítulo 2, são discutidos os trabalhos relacionados e o estado da arte. No Capítulo 3, é descrita uma arquitetura de referência de um computador de bordo para aplicações espaciais, e também os efeitos da radiação em componentes eletrônicos. A arquitetura de hardware e software do OBC proposto é mostrada no Capítulo 4. No Capitulo 5, é apresentada a técnica de tolerância a falhas proposta, assim como as modificações implementadas na arquitetura do *soft processor* para que os efeitos da radiação sejam minimizados. No Capitulo 6 são ilustradas as ferramentas utilizadas ao longo do desenvolvimento do trabalho. As metodologias adotadas para o teste do sistema visando à validação da arquitetura modificada e a validação da técnica de tolerância a falhas são discutidos no Capitulo 7. O Capitulo 8 apresenta os resultados obtidos, ilustrando a eficiência da arquitetura desenvolvida Por fim, no Capitulo 9 são apresentadas as conclusões e os trabalhos futuros.

2. TRABALHOS RELACIONADOS

A demanda atual por satélites artificiais faz com que se busquem maneiras de reduzir seu custo de produção. A utilização de componentes eletrônicos de uso geral, sem nenhuma proteção especial contra radiação, em aplicações espaciais aparece como uma possibilidade de alcançar esse objetivo.

Porém, para que seja possível utilizar componentes comerciais de prateleira (*Commercial Off-The-Shelf*, COTS) no ambiente espacial, é necessário à aplicação de técnicas para mitigação das falhas ocasionadas pela radiação. Assim, o estudo sobre os efeitos da radiação em componentes eletrônicos e técnicas para mitigação desses efeitos são de grande interesse para indústria espacial.

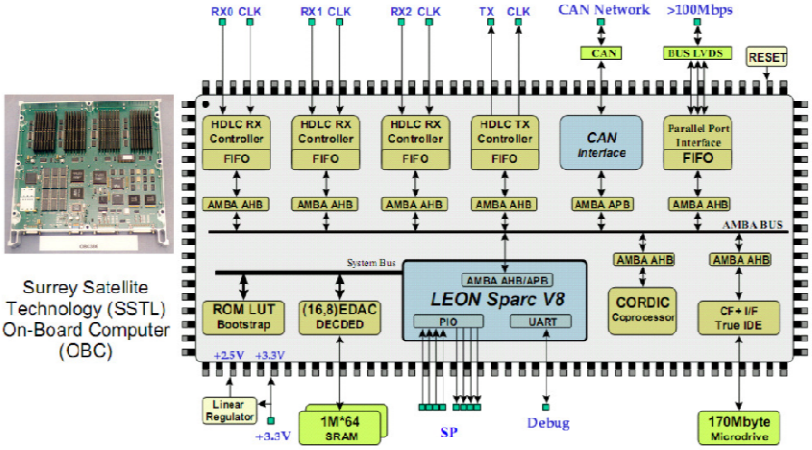
A pesquisa realizada busca demonstrar a importância do tema proposto e sua contextualização no estado da arte. Para tanto, foi realizada uma revisão bibliográfica buscando o aperfeiçoamento dos conhecimentos necessários para o desenvolvimento deste trabalho e a descrição de outras pesquisas nos temas de interesse, como sistemas embarcados, aplicações espaciais, tolerância a falhas em FPGAs e em processadores de uso geral.

Em (Pessota, 1999) foi realizado um estudo de diversas arquiteturas de computadores de bordo para satélites com o intuito de selecionar a arquitetura a ser utilizada no projeto de um satélite para uma missão de 10 anos de duração na órbita terrestre. As arquiteturas estudadas são baseadas em sistemas utilizados pelo INPE em satélites com missões de curta duração. As arquiteturas foram analisadas considerando a confiabilidade do sistema. Esse trabalho é importante, pois demonstra a utilização de diversas técnicas de tolerância a falhas em arquiteturas reais de computadores de bordo para satélites, sendo que alguns deles ainda estão em órbita.

O estudo sobre a utilização de FPGAs como unidade central de processamento de satélites é realizado no Centro Espacial de *Surrey* (Surrey Space Centre, 2013). Onde ocorre o desenvolvimento de satélites de pequeno porte utilizando o padrão de pilha de protocolos de comunicação do *Consultative Committee for Space Data Systems* (CCSDS) (CCSDS, 2003a) (CCSDS, 2003b) (CCSDS, 2003c). Entre os projetos realizados, pode-se citar o projeto *ChipSat* (Vladimirova & Curiel, 2004). No *ChipSat*, um computador de bordo de um satélite artificial é implementado na forma de um *System-on-Chip* (SoC). Núcleos de propriedade intelectual escritos na linguagem de descrição de hardware VHDL são usados para construir sistemas do computador

de bordo. Os principais blocos do SoC consistem em: microprocessador (LEON SPARC), unidade de detecção e correção de erros (EDAC) em memória, sistemas de inicialização (*boot*), controlador HDLC (*High Level Data Link Control*), interface de rede CAN (*Controller Area Network*), interface IDE (*Integrated Drive Electronics*), coprocessador aritmético e interface de barramento de periféricos AMBA. Um FPGA *Xilinx Virtex* é utilizado para prototipação do SoC. Na Figura 1 é apresentado o diagrama de blocos do *ChipSat* e suas conexões aos pinos do FPGA.

Figura 1 - Diagrama de blocos do *ChipSat*.



Fonte: (Vladimirova & Curiel, 2004)

O projeto visa implementar um sistema para um satélite de pequeno porte (cerca de 100 a 200 kg). A especificação do SoC é baseada em requisitos de missões de satélites da empresa *Surrey Satellite Technology Ltd*. (SSTL) que desenvolve e manufatura satélites de pequeno porte a mais de 20 anos. O estudo mostrou que é possível implementar as funcionalidades de um OBC para pequenos satélites em FPGAs. Porém, FPGAs de uso geral estão suscetíveis aos efeitos da radiação em ambientes hostis como o espaço. Sendo necessário que se apliquem técnicas para identificação de possíveis falhas comportamentais causadas por esses efeitos.

O conceito de identificação de falhas simples em tempo de execução foi introduzido em (Straka, Tobola, & Kotasek, 2007). Os autores utilizaram um modelo de verificação funcional para validar a implementação de um protocolo de comunicação em FPGA. Utilizando

uma linguagem de verificação formal, é possível descrever o comportamento de um componente por propriedades lógicas. A descrição formal do protocolo de comunicação é transcrita em um modelo de maquinas de estado finito, tornando possível a tradução do modelo para VHDL e implementação em FPGA. Assim, o módulo de verificação em tempo de execução (*On-Line Checker*) verifica se nenhuma das propriedades definidas na verificação formal é desrespeitada, gerando um sinal de erro caso isso ocorra.

Seguindo a linha de pesquisa, os autores utilizaram os conceitos descritos por Straka em (Straka, Kotasek, & Winter, 2008), e em (Straka, Kastil, & Kotasek, 2010), onde um estudo sobre o *overhead* de área em FPGA causado pela técnica apresentada foi elaborado. Os autores comparam a área em FPGA necessária para utilização de Redundância Modular Tripla (*Triple Modular Redundancy*, TMR), Redundância Dupla com um *Checker* e Redundância Dupla com dois *Checkers*. Os resultados mostram que o *overhead* causado pela técnica torna inviável que a mesma seja utilizada em aplicações complexas. Devido ao elevado número de propriedades necessárias para caracterizar o comportamento desse tipo de aplicação, a área utilizada pelo *On-Line Checker* pode, facilmente, ultrapassar a área disponível em um FPGA. A área utilizada pode se tornar um fator crítico na implementação da técnica em FPGAs COTS.

O conceito de identificação de falhas simples em tempo de execução constituiu-se em uma alternativa interessante para mitigação de efeitos de radiação. Porém, a maneira como a técnica foi aplicada mostrou-se não viável para aplicações complexas, como é o caso do OBC. Sendo assim, foi necessário buscar alternativas que possibilitassem a identificação de falhas simples em aplicações complexas como um processador embarcado.

No trabalho desenvolvido por Zalke (B. zalke & Pandey, 2009) é apresentado o conceito de controle de fluxo em tempo de execução. Um comparador é aplicado para identificar erros no fluxo do programa em execução em um microcontrolador. O modelo em VHDL do microcontrolador 8051 (Thomson, 1991) embarcado em um FPGA é utilizado no trabalho. Para identificar erros no fluxo do microcontrolador, o comparador calcula uma assinatura baseada em sinais de controle e no estado atual da memória de instruções do microprocessador. A assinatura calculada é comparada com um conjunto de assinaturas previamente armazenadas. Caso a assinatura calculada seja inválida, um erro é identificado. O conjunto com todas as assinaturas possíveis é calculado previamente e é baseado em todos os

possíveis caminhos de execução que a aplicação a ser executada no microcontrolador pode seguir. Os autores utilizaram um injetor de falhas, simulando o efeito de troca em um dos valores internos do processador para demonstrar o funcionamento da técnica.

A necessidade de calcular o conjunto de assinaturas antecipadamente, limita a utilização da técnica. Em aplicações com grau elevado de complexidade, torna-se inviável que todos os estados possíveis do processador, e caminhos de execução da aplicação sejam analisados.

Em (Bernardi, Bolzani, Rebaudengo, Reorda, Vargas, & Violante, 2006) e (García, et al., 2012), os autores utilizam um observador de barramentos para fazer controle de fluxo da aplicação em execução. A técnica utiliza uma combinação de *software* e *hardware* para identificar falhas em tempo de execução. A aplicação a ser executada é alterada para que sejam adicionadas instruções que contém informações a respeito do fluxo normal da aplicação. Durante a execução, o barramento do processador é monitorado em busca de uma das instruções, quando identificada, é possível verificar as informações da instrução para detectar execuções incorretas.

A principal contribuição do presente trabalho é a utilização do conceito de controle de fluxo por observação de barramentos para simples em tempo de execução de maneira não intrusiva. Ou seja, não alterando o funcionamento do processador, e sem modificações na aplicação sendo executada. Além disso, buscou-se diminuir o *overhead* de área em FPGA, para que a técnica possa ser utilizada em FPGAs COTS, e tornar a técnica não dependente da aplicação sendo executada no processador.

3. COMPUTADORES DE BORDO PARA APLICAÇÕES ESPACIAIS: PROBLEMAS E ARQUITETURA

3.1. EFEITOS DA RADIAÇÃO NA ELETRÔNICA EMBARCADA

A radiação no espaço é gerada por partículas emitidas de diversas fontes que vão além do sistema solar. Os efeitos causados por essas partículas radioativas causam não só a degradação precoce de componentes eletrônicos, mas também, falhas nos sistemas elétricos e eletrônicos de veículos espaciais e satélites. Com a miniaturização dos circuitos eletrônicos, resultante dos avanços da tecnologia de fabricação de semicondutores, já é possível que os efeitos de radiação sejam percebidos em componentes utilizados ao nível do mar.

Existem três fontes principais de partículas carregadas responsáveis pelas falhas em componentes eletrônicos, são elas: os Raios Cósmicos, os Ventos Solares e o Cinturão de Van Allen (Velazco, Fouillat, & Reis, 2007) (Battezzati, Sterpone, & Violante, 2011).

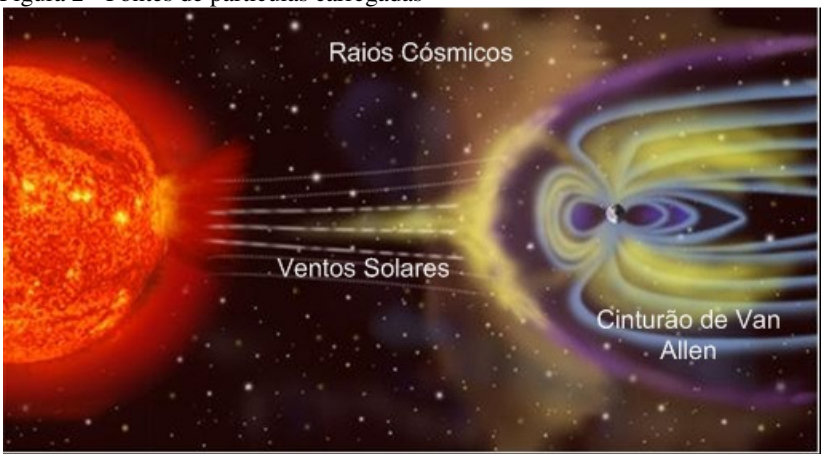
Os Raios Cósmicos são partículas altamente carregadas, em sua maioria íons pesados, que viajam a velocidades próximas a da luz no espaço sideral. Quando essas partículas chegam a Terra e colidem com os núcleos dos átomos da atmosfera terrestre dão origem a outras partículas com menos energia, chamadas Raios Cósmicos secundários.

O processo de fusão no interior do Sol produz uma grande quantidade de prótons e elétrons. Essas partículas viajam no espaço através do Vento Solar. O Vento Solar é irradiado pelo Sol em todas as direções. Variações na coroa solar devido à rotação do Sol e às suas atividades magnéticas tornam o Vento Solar variável e instável. Além das partículas providas pelo Sol, os Ventos Solares carregam partículas de outras estrelas e íons altamente carregados gerados por fenômenos como a Supernova. Supernova é o nome dado aos corpos celestes surgidos após as explosões de estrelas que produzem objetos extremamente brilhantes.

Todas essas partículas carregadas sofrem influência dos campos magnéticos dos planetas formando cinturões radioativos ao redor dos mesmos. No caso da Terra, o cinturão é conhecido como Cinturão de Van Allen. O Cinturão de Van Allen é constituído de elétrons e prótons presos no campo gravitacional da Terra (Velazco, Fouillat, & Reis, 2007).

A Figura 2 ilustra a influência das fontes de partículas carregadas sobre a Terra.

Figura 2 - Fontes de partículas carregadas



Fonte: (Battezzati, Sterpone, & Violante, 2011)

Componentes eletrônicos modernos são muito sensíveis a esse ambiente altamente radioativo. A presença de partículas ionizantes altamente carregadas (prótons, elétron, íons pesados) induz diferentes efeitos quando em contato com semicondutores.

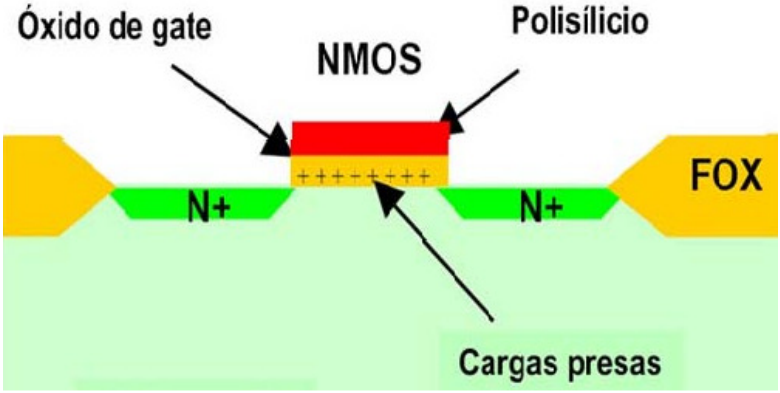
Para que componentes eletrônicos possam ser utilizados com segurança no espaço, é necessário que esses tenham a capacidade de continuar realizando suas tarefas por todo o período da missão independentemente dos efeitos que a radiação cause sobre eles. Existem diversos tipos de falhas em hardwares que são causadas pela radiação. Porém, os dois efeitos principais levados em consideração em aplicações espaciais são a Dose Ionizante Total (*Total Ionizing Dose,* TiD) (Benfica, et al., 2012) e os Efeitos em Eventos Simples (*Single Event Effects*, SEE) (García, et al., 2012).

3.1.1. DOSE IONIZANTE TOTAL

O acúmulo homogêneo de partículas ionizantes em partes de um semicondutor ao longo do tempo é o motivo do efeito chamado Dose Ionizante Total (*Total Ionizing Dose*, TiD). Entre os efeitos causados pela TiD, pode-se citar o acúmulo de elétrons e lacunas no isolante de um semicondutor causando degradação no desempenho elétrico do componente. Por exemplo, o efeito causado pelo TiD nos transistores do tipo MOSFET. Na Figura 3, é possível notar o acúmulo de cargas positivas no óxido de *gate* devido a sua exposição à radiação. O

acúmulo de cargas facilita a condução de elétrons entre o *drain* e o *source* do transistor.

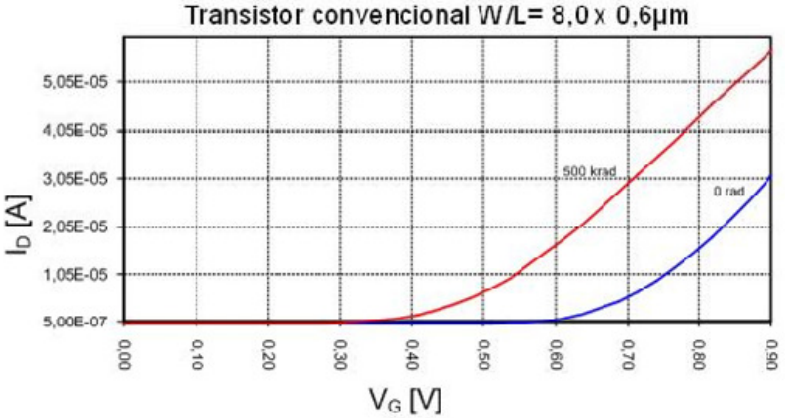
Figura 3 - Efeito de radiação acumulada em transistores MOSFET



Fonte: Traduzido de (Battezzati, Sterpone, & Violante, 2011)

Esse fenômeno faz com que a tensão de *gate* (Vg(V)) necessária para que o transistor saia do estado de corte e passe para o estado de condução, seja menor, como ilustra o gráfico na Figura 4. Na Figura 4, pode-se notar que no transistor sob influência de radiação (linha vermelha) existe uma corrente (Id) para uma tensão (Vg) menor, sendo esse comportamento diferente do esperado para o transistor, representado pela linha azul. A quantidade de radiação a qual esse transistor foi exposto é da ordem de 500 krad, considerada baixa para aplicações espaciais.

Figura 4 - Comparação entre os transistores



Fonte: Traduzido de (Battezzati, Sterpone, & Violante, 2011)

3.1.2. EVENTOS DE EFEITO SIMPLES

Os Eventos De Efeito Simples (*Single Event Effect*, SEE) são perturbações no comportamento de componentes, circuitos ou sistemas eletrônicos causados por uma partícula ionizante. Quando a carga gerada por uma partícula ionizante entra em contato com um nodo sensível de um componente, circuito ou sistema eletrônico, a carga pode ser grande o suficiente para afetar o desempenho elétrico do mesmo.

A maneira como memórias SRAM são fabricadas as tornam suscetíveis a falhas causadas por SEEs. As células de memórias SRAM são formadas por transistores distribuídos de maneira muito densa no encapsulamento. A diminuição do espaço entre transistores e dos transistores em si, a cada avanço tecnológico, faz com que as chances de uma célula ser atingida por uma partícula carregada aumentem. Quando uma partícula carregada atinge uma célula de memória SRAM, essa gera uma corrente momentânea comprometendo a integridade do valor armazenado na célula de memória (Battezzati, Sterpone, & Violante, 2011). No caso de FPGAs SRAM as células de memória guardam configurações sobre a lógica que está sendo executada no FPGA. Uma mudança no valor armazenado em uma célula de memória de um bloco lógico de um FPGA SRAM pode causar uma mudança na lógica armazenada, por exemplo, alterando a função lógica NAND para AND previamente programada na memória (Battezzati, Sterpone, & Violante, 2011) (Entrena, Garcia-Valderas, Fernandez-Cardenal, Lindoso, Portela, & Lopez-Ongil, 2010). Quando a célula atingida guarda valores

referentes à matriz de roteamento, é possível que conexões entre blocos sejam perdidas, ou que blocos incorretos sejam conectados. A Figura 5 ilustra os possíveis erros causados quando uma partícula atinge uma célula de memória de bloco lógico e uma célula de memória da matriz de roteamento de um FPGA SRAM.

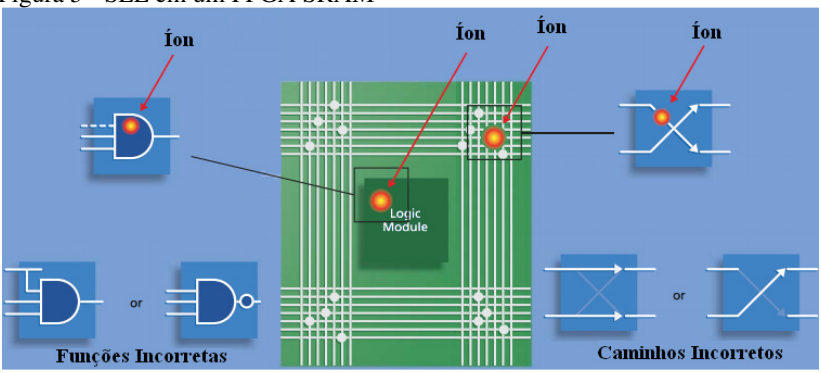
Existem três formas principais de SEEs que afetam FPGAs do tipo SRAM, são elas: Falha Permanente (FP), *Single Event Transients* (SET) e *Single Event Upset* (SEU) (George, Koga, Swift, Allen, Carmichael, & Tseng, 2007) (García, et al., 2012).

3.1.2.1. Falha Permanente

A Falha Permanente (FP) é um tipo de curto circuito que acontece em circuitos integrados; ocorre quando um caminho parasita de baixa impedância é criado entre a entrada e a saída de um componente, anulando o componente. No caso dos FPs, o caminho parasita é criado devido à ação da partícula ionizante sobre o transistor. A ocorrência de FP é uma preocupação primária em aplicações espaciais por ser considerada uma falha destrutiva que pode gerar uma falha permanente na aplicação.

Usualmente, componentes que sofrem efeitos de FPs não são utilizados em aplicações espaciais, pois no caso de ocorrência de um FP o componente estaria permanentemente danificado. Famílias de FPGAs modernos são imunes a FPs, segundo informações dos próprios fabricantes que fazem testes para garantir a imunidade.

Figura 5 - SEE em um FPGA SRAM



3.1.2.2. Eventos Transientes Simples

Eventos Transientes Simples (*Single Event Transients*, SET) são comuns em muitos circuitos semicondutores; ocorrem quando uma partícula ionizante gera uma corrente transiente que pode alterar, momentaneamente, o valor de um processo, causando um erro caso o valor seja registrado. Apesar da ocorrência de SETs ser comum em FPGAs do tipo SRAM, usualmente não é possível distingui-los de SEUs.

3.1.2.3. Evento Simples de *Upset*

Uma partícula carregada ao entrar em contato com uma superfície qualquer perde energia para essa superfície, ionizando-a. No caso de semicondutores, a ionização acarreta em acúmulo de pares de elétron e lacunas que podem mudar o estado lógico do semicondutor.

Em circuitos eletrônicos, podem ocorrer eventos em que o valor de um *bit* é alterado sem razão aparente, não causando danos permanentes ao *hardware*, ou seja, o *bit* pode ser sobrescrito. Esse tipo de evento não pode ser reproduzido, pois depende totalmente de uma ação externa. Esse tipo de erro é conhecido como *soft error*. *Soft Errors* são causados por perturbações elétricas, como por exemplo: fontes de alimentação ruidosas, exposição do circuito a descargas eletrostáticas (ESD - Electro-Static Discharge), à radiação e à interferência eletromagnética (EMI – Electromagnetic Interference), entre outras.

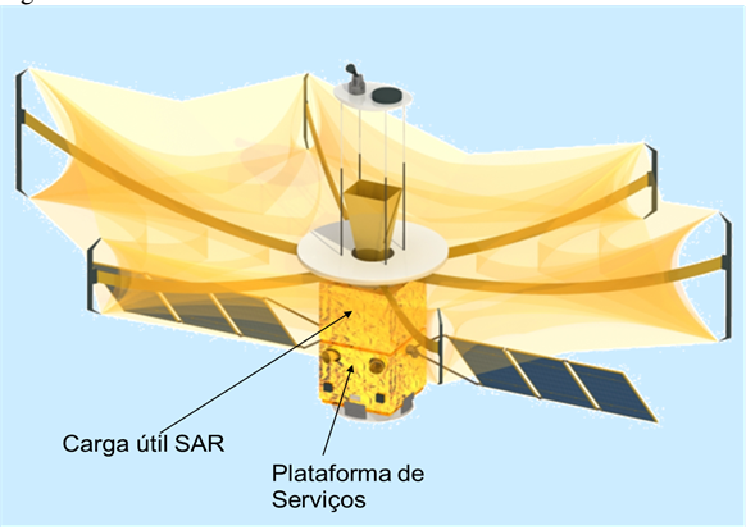
As falhas causadas por Evento Simples de *Upset* (*Single Event Upset*, SEU) são da mesma natureza dos *soft errors*, ou seja, são falhas transientes causadas por eventos externos. No caso, contato com uma partícula radioativa carregada. Os SEUs são considerados falhas não destrutivas, pois o componente que sofreu a troca de valor não está permanentemente danificado e funcionará normalmente caso o valor seja sobrescrito.

3.2. PLATAFORMA DE SERVIÇOS DE UM SATÉLITE ARTIFICIAL

O presente trabalho contempla a pesquisa e desenvolvimento de um OBC para plataforma de serviços de satélites. A plataforma de serviços é onde se encontram os módulos de navegação e comunicação do satélite, tendo o OBC como a unidade central de processamento.

Satélites são formados, basicamente, por dois sistemas distintos a plataforma de serviços e a carga útil (*payload*). A plataforma de serviços é responsável por executar funções tais como: alimentação, controle de temperatura, controle de atitude, controle de órbita e gerenciamento de comunicação com a Terra. Por outro lado, a carga útil é a razão de ser do satélite, carregando sistemas dependentes de cada missão, tais como: sistemas de telecomunicação, experimentos científicos, sistemas de monitoração ambiental, telescópios, entre outros. A Figura 6 ilustra a estrutura do satélite MAPSAR, onde estão destacadas a Carga Útil e a Plataforma de Serviços. O MAPSAR é um satélite que utiliza um Radar de Abertura Sintética (*Synthetic Aperture Radar*, SAR) para obtenção de imagens encobertas por determinados tipos de obstáculos como, por exemplo, folhas de árvores. O SAR, carga útil do satélite, opera em micro-ondas fornecendo informações geométricas e elétricas dos objetos em observação, sendo um dos únicos sensores remotos com penetrabilidade na copa vegetal (Schröder, et al., 2005). Com lançamento previsto para 2014, o MAPSAR será utilizado para sensoriamento remoto da Amazônia.

Figura 6 - Estrutura do satélite MAPSAR



Fonte: (Schröder, et al., 2005)

O OBC é a unidade central da plataforma de serviços, sendo responsável por todo o processamento de dados, comunicação e tomada

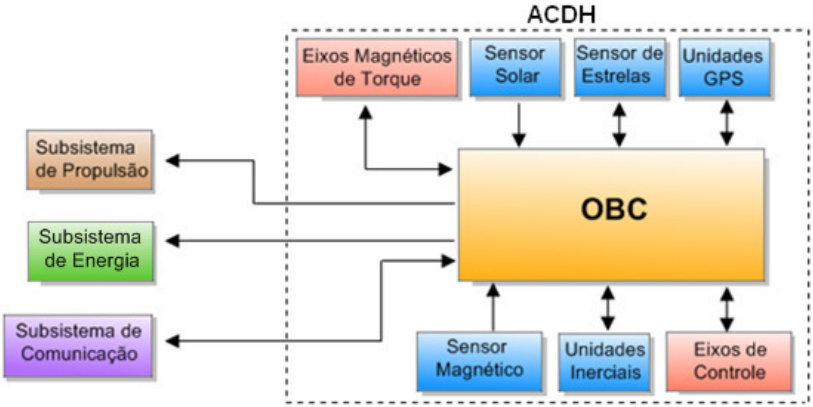
de decisões em um satélite artificial. Em outras palavras o OBC é a central de gerenciamento do satélite, pois todos os módulos referentes ao controle do satélite são ligados diretamente ao OBC. A ocorrência de uma falha no OBC pode causar danos permanentes ao satélite, podendo fazer com que esse perca sua órbita correta, perca comunicação com a Terra, entre outros.

Na Figura 7, é tem-se um diagrama de blocos simplificado de uma plataforma de serviços semelhante à idealizada pelo INPE para suas futuras missões. Representado pelo quadrado pontilhado está o Controlador de Atitude e Manipulação de Dados (*Attitude Control and Data Handling*, ACDH), que é responsável pelo controle de atitude, órbita, e gerência de dados do satélite.

3.2.1. CONTROLADOR DE ATITUDE E MANIPULAÇÃO DE DADOS

O ACDH é composto por um OBC, atuadores e instrumentos responsáveis pelo controle sensorial do satélite. Os subsistemas são gerenciados e configurados através do ACDH, ou seja, existe um canal de comunicação entre o OBC e todos os subsistemas presentes na plataforma de serviços do satélite (INPE, 2001a) (INPE, 2001b) (INPE, 2008). A plataforma é formada por quatro subsistemas principais: de propulsão, de comunicação, de potência e o ACDH.

Figura 7 - Plataforma de Serviços de um satélite



Fonte: (Silva, Ferreira, & Villa, 2009), adaptado de (FUNDEP, 2011)

Como citado anteriormente, o ACDH é responsável pelo controle de atitude e manipulação de dados do satélite. O termo atitude, no caso de um satélite, diz respeito a sua orientação no espaço. O controle de atitude é responsável por estabilizar o satélite e orientá-lo para direções desejadas durante a missão, levando em consideração a existência de perturbações externas atuando no sistema. Essas perturbações de atitude são causadas por torques aerodinâmicos, por torques magnéticos devidos à pressão de radiação solar, torques devidos ao gradiente de gravidade, ou ainda, erros nas medições feitas pelos próprios sensores. A atitude de um satélite deve ser controlada, sendo suficientemente estável para que seu funcionamento não seja comprometido, por exemplo: para permitir que antenas de comunicação estejam voltadas para a Terra; que os experimentos a bordo estejam apontados para a direção correta; para que o controle de temperatura possa utilizar de forma inteligente os efeitos de aquecimento e resfriamento gerados pela radiação solar e pelas sombras; para guiar o satélite executando manobras para posicioná-lo na direção correta; entre outros.

A estabilização do satélite é realizada através de seus atuadores. Por exemplo, pode ser realizada através do efeito de rotação do satélite, também chamado de *Spinning*. Para isso, pequenos propulsores são acionados para controlar a velocidade de rotação do satélite. Outra maneira de alcançar estabilidade é através de três eixos ativos, existem dois tipos de atuadores utilizados nesse caso. Pequenos propulsores, um em cada eixo, que são acionados incessantemente não permitindo que o erro de atitude ultrapasse um valor máximo pré-determinado. E também, utilizando rodas de reação (*Reaction Wheels*). As rodas de reação são mecanismos que contém volantes que podem girar tanto no sentido horário como no anti-horário com velocidades variáveis, sua rotação é ativada e mantida por um motor de rotação controlada. Um sistema de controle com rodas de reação contém três rodas ortogonais entre si, sendo que seus eixos de rotação não têm liberdade de movimento, pois estão fixos ao satélite. Ao movimentar o satélite, a roda reagirá tendendo a aumentar ou diminuir sua velocidade de rotação, da mesma maneira, modificando a velocidade de rotação da roda haverá um movimento no satélite.

O ACDH utiliza seus sensores para definir se há necessidade de uma mudança de atitude, e então, caso necessário, ativa um dos atuadores. Existem dois grupos principais de sensores em um satélite, os de referência celestial e os de referência inercial.

Sensores de referência celestial são os que observam corpos celestiais como estrelas, planetas e o Sol. Um sensor de estrela, por exemplo, tem a capacidade de reconhecer automaticamente uma estrela ou conjunto de estrelas baseado em sua base de dados. Porém, em alguns casos, quando a referência celestial não está disponível, o satélite deve utilizar a referência inercial.

Sensores inerciais têm a capacidade de determinar a direção e velocidade do movimento a qual o satélite foi submetido. Para isso, eles medem os seis graus de liberdade que um objeto tem para se mover no espaço. Há três graus lineares de liberdade (X, Y e Z) que especificam sua posição, medidos por acelerômetros, e três graus de liberdade rotacionais, *Theta* (*Pitch*), *Psi* (*Yaw*), e *Phi* (*Roll*), medidos por giroscópios, que especificam sua atitude. Sendo conhecidas essas seis variáveis, é possível saber a localização do satélite e, após algumas medições, sua velocidade.

Existem outros sensores e atuadores que podem ser utilizados nesse tipo de aplicação, entre eles pode-se citar a Unidade de GPS e o Sensor Magnético. O sensor magnético, também chamado de magnetômetro, é utilizado para medir o sentido, direção e intensidade de campos magnéticos. Assim, sendo conhecido o vetor campo magnético, é possível determinar a posição do satélite. Unidades GPS embarcadas em satélites funcionam de maneira análoga as utilizadas na Terra. Ou seja, é efetuada uma triangulação baseada na posição de três ou mais satélites; assim, é possível calcular a posição atual da Unidade GPS.

Os dados adquiridos dos sensores são utilizados como entrada em algoritmos de controle que geram as instruções corretas para os atuadores. Esses algoritmos são geralmente complexos, pois envolvem diferentes variáveis e devem alcançar altos graus de precisão em seus resultados.

Derivado do ACDH há outra nomenclatura para a plataforma de serviços chamada de *On-Board Data Handling* (OBDH); essa nomenclatura aplica-se quando o controle de atitude e órbita é realizado por outro módulo de forma independente (Aranci, Maltecca, & Ranieri., 1997). A arquitetura do OBDH restringe-se a um computador de bordo que se comunica com os demais subsistemas e *payload,* realizando o gerenciamento e processamento da comunicação entre os subsistemas.

3.2.2. SUBSISTEMA DE COMUNICAÇÃO

Esta seção descreve a interação do subsistema de comunicação com o OBC, bem como o gerenciamento dos dados provenientes dos instrumentos sensoriais.

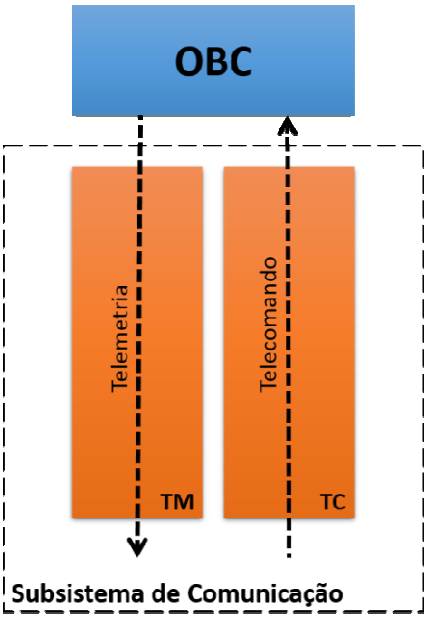
Na plataforma de serviços do satélite, o subsistema de comunicação é o módulo responsável pela comunicação com a Estação Terrestre. Através deste, é possível que comandos sejam enviados para o OBC e que informações sobre o estado dos diversos módulos presentes no satélite sejam monitorados. Exemplos de comandos enviados incluem os de mudança de atitude do veículo espacial, ativação de módulos e sensores, entre outros. Em contrapartida, o satélite necessita enviar respostas a um determinado comando, informações de posição do satélite, situações de exceção ou erro e informações de perda de mensagens. Por essa razão, esse módulo do ACDH é considerado um componente crítico, pois uma falha neste módulo, como citado anteriormente, pode resultar no fim da missão. Além do subsistema de comunicação, em um satélite existem outras formas de troca de mensagens, uma delas é a transmissão de informações provenientes do *payload*, tais como imagens coletadas e dados de sensoriamento remoto.

As mensagens que partem da Estação Terrestre são chamadas de Telecomando (TC). Um TC pode seguir dois fluxos distintos dentro do subsistema de comunicação: ser repassado para que o OBC realize o seu processamento, ou realizar o acionamento de algum instrumento diretamente, sem passar pelo OBC (CCSDS, 2003c) (ECSS, 2003). Essa distinção existe, pois em algumas situações é desnecessário o OBC realizar o processamento do pacote; além disso, em alguns casos o equipamento pode ter sofrido alguma avaria e estar impossibilitado de processar a informação.

Realizando o fluxo contrário ao do TC, estão as mensagens de Telemetria (TM). As TMs possuem caráter informativo, ou seja, relatam com mensagens periódicas aspectos comportamentais relevantes do satélite, ou respondem uma solicitação que foi requisitada por TC (Almeida G. M., 2007) (CCSDS, 1987). Em um caso particular, quando a Telemetria não está enviando nenhuma informação específica, esta é responsável por enviar pacotes vazios, esse mecanismo é utilizado para garantir que o satélite está respondendo e o canal de comunicação está funcionando corretamente, evitando assim, possíveis falhas de comunicação.

Os fluxos de Telecomando e Telemetria estão representados na Figura 8.

Figura 8 - Fluxos de Telecomando e Telemetria



Os fluxos de telecomando e telemetria foram implementados em um projeto no contexto dessa pesquisa, de acordo com as recomendações CCSDS (*Consultative Committee for Space Data Systems*) (Azevedo, 2010). O CCSDS é uma organização oficialmente estabelecida pelas principais agências espaciais mundiais ligadas aos seguintes países: EUA, Canadá, Alemanha, Japão, Reino Unido, França, Rússia, Itália e Brasil. As agências espaciais desses países formam um comitê, que se reúne periodicamente para discutir problemas comuns, ocorridos em sistemas de comunicação, e formular soluções técnicas seguras para esses problemas. A principal atividade dos membros do CCSDS consiste na escrita e manutenção da documentação relativa às recomendações. Uma vez que a participação no CCSDS é completamente voluntária, os resultados das ações tomadas pelo comitê são designados como recomendações e não são consideradas obrigatórias em qualquer agência espacial. O Subsistema de Comunicação utilizado atualmente pelo INPE adota as recomendações do CCSDS e o padrão definido pela Agência Espacial Europeia (*European Space Agency*, ESA).

Para garantir a segurança e a integridade das informações trocadas entre o satélite e os controladores da missão, são implementados nos fluxos de TC/TM algoritmos de codificação e decodificação capazes de realizar detecção e correção de erros. O algoritmo BCH (Bose, Chaudhuri and Hocquenghem) (S□erban, Anton, Tuta□nescu, Ionescu, & Maza□re, 2010) foi utilizado no fluxo de TC. Na TM foram utilizados dois algoritmos, o RS (Reed-Solomon) (Almeida, Bezerra, Cargnini, Fagundes, & Mesquita, 2007) e o Convolucional (Gaisler Research, 2002), que podem ser aplicados de forma isolada ou combinados, o que garante maior confiabilidade na troca de informações.

3.2.3. DEMAIS SUBSISTEMAS

O subsistema de energia é responsável pela geração, armazenamento, condicionamento e distribuição de energia elétrica para todos os subsistemas do satélite. Usualmente, contém uma série de baterias recarregáveis através de luz solar. Existem ainda outras formas de obter energia tais como pilhas nucleares ou células combustíveis de hidrogênio. O satélite possui painéis solares que permitem máxima captação da energia gerada pela luz do Sol. O subsistema de energia deve otimizar o consumo de energia para que a energia em suas baterias seja suficiente para o período em que o satélite estiver sem contato direto com o Sol. É de suma importância que o controle de atitude do satélite posicione-o de maneira a maximizar o aproveitamento da luz do Sol pelos painéis solares.

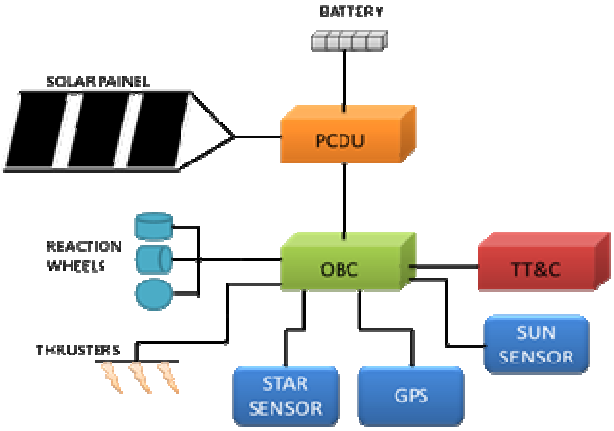
O subsistema de propulsão é responsável pelo controle dos propulsores e do tanque de combustível do satélite. Ele é composto basicamente por filtros, válvulas, tanques, motores e propulsores. É ele que fornece o impulso necessário às manobras de correção de órbita e atitude, determinadas pelos dados providos pelo ACDH. Cabe notar que a massa desse subsistema pode chegar a 50% da massa do satélite no lançamento.

A Figura 9 ilustra um exemplo de disposição dos subsistemas em um satélite. Todos os subsistemas citados nesse capítulo estão ilustrados na figura, assim como as ligações internas entre os mesmos, os sensores, atuadores e demais componentes.

O bloco denominado PCDU (*Power Conditioning and Distribution Unit*) representa a central do subsistema de energia, que também conta com os painéis solares e baterias. Como elemento central na figura está o OBC. O OBC está ligado a todos os sensores e

atuadores do satélite assim como aos demais subsistemas. Entre os sensores estão à unidade de GPS, sensor solar (*Sun Sensor*) e o sensor de estrelas (*Start Sensor*). Entre os atuadores estão às rodas de reação (*Reaction Wheels*). Os propulsores (*Thrusters*) também são atuadores, porém fazem parte do subsistema de propulsão. O subsistema de comunicação está representado pelo bloco TT&C (*Telemetry, Tracking & Command*).

Figura 9 - Subsistemas da Plataforma de Serviços



Fonte: (Ferlini, Silva, Bezerra, & Lettnin, 2012)

3.3. EFEITOS DA RADIAÇÃO NO OBC

A arquitetura de OBC, proposta nesse trabalho, utiliza um FPGA SRAM como módulo central de processamento. Como qualquer componente eletrônico utilizado no espaço, o OBC está exposto aos efeitos de radiação citados na Seção 3.1. O principal efeito de radiação aos quais os FPGAs SRAM, estão suscetíveis é o SEU. Como citado anteriormente, a ocorrência de um SEU pode afetar a lógica previamente programada no dispositivo. No caso do presente trabalho, a lógica embarcada no FPGA é o *soft processor* LEON3 e seus periféricos. Levando em consideração a suscetibilidade dos FPGAs SRAM a efeitos do tipo SEU, é importante que os efeitos de uma mudança de valor em um registrador sejam estudados para a aplicação específica do presente trabalho, ou seja, o LEON3.

Tipicamente, um microprocessador é formado por diversas unidades funcionais que, normalmente, não são utilizadas ao mesmo

tempo durante a execução de uma aplicação. Caso um SEU atinja um registrador que não está em uso, nenhum defeito seria gerado no nível do sistema e o SEU seria corrigido assim que o registrador fosse sobrescrito. Nos casos em que um registrador em uso sofre influência de um SEU, os efeitos gerados no nível de sistema são totalmente dependentes da função empregada pelo registrador, e os efeitos gerados podem ser divididos nos seguintes grupos (Avizienis, Laprie, & Randell, 2000):

- Silencioso: o efeito causado pelo SEU desaparece devido a uma reescrita no registrador. Apesar da mudança no valor de um registrador causada pelo SEU, o valor errôneo não foi utilizado pelo processador.
- Latente: os valores internos dos *flip-flops* e memórias do processador sob efeito de SEUs são diferentes dos valores em funcionamento normal. Porém, os valores errôneos não são propagados para nenhuma saída do circuito.
- Erro: os efeitos de SEU são propagados para pelo menos uma saída do circuito, gerando um erro no sistema como um todo.

Dentre os efeitos causados pela incidência de SEUs em um processador o mais prejudicial é o Erro. A mudança no estado de um *flip-flop,* causada pelo SEU, pode se manifestar de diferentes maneiras na saída do circuito. Desde um acesso a uma posição errada de memória até um resultado diferente do esperado para uma operação. Para evitar que o erro causado pelo SEU propague-se para fora do processador, podendo causar um defeito no sistema como um todo, é necessário que se aplique técnicas de tolerância a falhas para permitir que as falhas no processador sejam transparentes para o resto do sistema, não causando maiores consequências.

No Capitulo 6, é apresentada uma arquitetura para computadores de bordo para aplicações espaciais. Essa arquitetura é uma evolução da arquitetura utilizada em (Silva, Ferreira, & Villa, 2009) que permite que falhas no processador sejam identificadas e mascaradas em tempo de execução.

4. ARQUITETURA PROPOSTA PARA O COMPUTADOR DE BORDO

Atualmente, grande parte dos equipamentos embarcados em satélites utilizam versões resistentes a efeitos de radiação de processadores comerciais (Fiethe, Michalik, Dierker, Osterloh, & Zhou, 2007). Esses dispositivos realizam o processamento e distribuem as tarefas para os demais componentes do sistema. Porém, essas arquiteturas, normalmente, utilizam porções consideráveis de área em uma placa e os custos de aquisição são relativamente elevados (Fiethe, Michalik, Dierker, Osterloh, & Zhou, 2007).

Uma das possíveis opções é a utilização de arquiteturas com processadores embarcados em dispositivos de lógica programável (FPGA) (Fiethe, Michalik, Dierker, Osterloh, & Zhou, 2007). Com essa técnica é possível atingir níveis satisfatórios em relação à área de ocupação do circuito. Já que utilizando FPGAs é possível embarcar um ou mais processadores em um único circuito integrado além de diversos periféricos agregados ao processador o que aumenta o desempenho da aplicação. Dispositivos de lógica programável são extremamente populares em soluções de projeto devido a sua grande flexibilidade e capacidade de reconfiguração, que reduz o tempo de desenvolvimento como um todo. Assim sendo, tornam-se, também, atraentes para soluções espaciais, pois agregam diversos benefícios para o projeto em termos de alta densidade lógica, alto desempenho, baixo custo das ferramentas de desenvolvimento e rápida adaptação às alterações do projeto durante o período de desenvolvimento.

4.1. O SOFT PROCESSOR LEON3

Existem dois tipos de processadores que podem ser embarcados em FPGAs os processadores com *soft* e *hard cores*. Processadores com *hard cores* ou *hard processors* são processadores que utilizam uma área dedicada do circuito integrado, como exemplo, pode-se citar a família Xilinx Virtex-4 FX que tem um Power-PC integrado ao circuito. Já os processadores com *soft cores* ou *soft processors* são embarcados no FPGA utilizando as células lógicas de uso geral. Assim, é possível embarcar diferentes processadores no FPGA sem que sejam necessárias modificações em sua fabricação. Como exemplo de *soft processors,* pode-se citar os processadores da família LEON (Gaisler, 2013).

Em meados de 1997, a ESA iniciou o projeto LEON com o intuito de desenvolver um processador de alto desempenho para ser

utilizado em suas missões. O objetivo era alcançar um processador portátil, com código aberto, capaz de alcançar bom desempenho, compatível com diversos softwares e com baixo custo. Atualmente, a família LEON conta com quatro versões do processador. As duas primeiras, LEON1 e LEON2, foram desenvolvidos diretamente pela ESA, enquanto que as versões LEON3 e LEON4 foram desenvolvidas pela empresa Aeroflex Gaisler. O LEON3 é a versão mais consolidada no mercado, sendo utilizado em uma série de sistemas para aviação e aplicações comerciais.

O LEON3 é distribuído pela Aeroflex Gaisler como parte da GRLIB IP *Library*. A GRLIB é um conjunto integrado de núcleos de Propriedade Intelectual (*Intelectual Property*, IP) desenvolvidos para utilização em Sistemas Embarcados em um Chip (*System on a Chip*, SoC). Para facilitar a utilização, todos os componentes utilizam um barramento comum o AMBA-2.0 (*Advanced Microcontroller Bus Architecture*) AHB (AMBA *High-performance Bus*). A GRLIB também conta com uma série de *scripts* para facilitar a configuração, simulação e síntese de seus componentes. Pode-se parametrizar a GRLIB através de configurações em seu VHDL sem alterar os demais pacotes de configuração global. Assim, é possível instanciar vários núcleos em um mesmo projeto, com diferentes configurações. As características da GRLIB e do LEON3 também podem ser definidas através de uma ferramenta gráfica, facilitando a configuração do sistema e a adição de periféricos *on-chip* como memórias, interfaces de rede entre outros. A GRLIB pode ser sintetizada com as principais ferramentas comerciais como Synplify, Synopsys DC, Cadence RC, Xilinx XST e Altera Quartus, utilizando *scripts* ou a interface gráfica das ferramentas.

O barramento AMBA, utilizado pela GRLIB, foi desenvolvido pela ARM Ltd, com intuito de facilitar a integração de diferentes periféricos em um SoC independente de tecnologia ou fabricante. Por sua facilidade de utilização, ampla documentação e por ser um padrão aberto o AMBA é atualmente um dos líderes do mercado, sendo o principal barramento dos processadores ARM. A versão AMBA-2.0 AHB possui dois barramentos distintos. O *Advanced High-performance Bus* (AHB) é utilizado para controladores de memória, controles de comunicação e IPs que necessitam de maior velocidade de comunicação. Enquanto o *Advanced Peripheral Bus* (APB) é utilizado para periféricos como controladores de PS2 e VGA.

Como citado anteriormente, o processador utilizado na GRLIB é o LEON3. O processador LEON3 é uma descrição VHDL sintetizável

de um processador de 32-bits compatível com a arquitetura SPARC V8. Entre as características do processador pode-se destacar:

- Conjunto de registradores, instruções e modos de endereçamento do SPARC V8;
- Pipeline avançado de sete estágios;
- Unidades de MAC, utilizadas em operações de DSP;
- *Cache* de instruções e de dados separadas (Arquitetura Harvard);
- AMBA-2.0 AHB *bus interface*;
- Suporte a *on-chip debug*;
- *Symmetric Multi-Processor support* (SMP);
- Até 125 MHz em FPGA e 400 MHz em ASIC com tecnologia de 0,13µm;

O LEON3 foi certificado pela SPARC *International* como sendo compatível com o SPARC V8, tornando possível que aplicações desenvolvidas para a arquitetura SPARC V8 sejam utilizadas no mesmo.

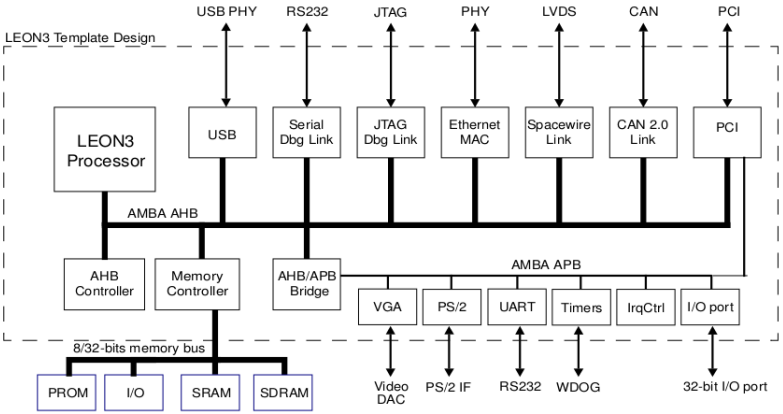
A Figura 10 ilustra um exemplo de um sistema gerado utilizando a GRLIB.

4.2. ARQUITETURA DE COMPUTADOR DE BORDO BASEADA NO SOFT PROCESSOR LEON3

Uma versão inicial de um Computador de Bordo para satélites foi desenvolvida durante o trabalho de conclusão de curso do autor da presente pesquisa: "Concepção do *On-Board Data Handling* com Sensor Inercial para Aplicações Espaciais" (Silva, Ferreira, & Villa, 2009). Esse trabalho de conclusão visou a implementação de um modelo básico para gerência e manipulação de dados a bordo de um satélite e integração do mesmo com um subsistema de comunicações. A integração foi utilizada para exemplificar o fluxo completo das mensagens transmitidas entre a Estação Terrestre e o veículo espacial. Foi utilizado um ESE (*Electrical Support Equipment*) para geração dos

TCs e verificação das TMs. Assim, o ESE é o responsável pelo papel de Estação Terrestre no fluxo de comunicação. Com o intuito de adicionar funcionalidades ao computador de bordo foram utilizados dois sensores inerciais, um acelerômetro, um giroscópio, e um painel de instrumentos.

Figura 10 - Exemplo de um sistema utilizando a GRLIB



Fonte: (Aeroflex Gaisler, 2001)

Na arquitetura desenvolvida na presente dissertação, assim como no trabalho de conclusão, o processador LEON3, embarcado em um FPGA Virtex-II da Xilinx, foi utilizado como elemento principal da Plataforma de Desenvolvimento. O LEON3 é a unidade central do bloco OBC, ilustrado no canto superior esquerdo da Figura 11. Sobre o LEON3 foi utilizado o sistema operacional de tempo real RTEMS que serve de suporte ao desenvolvimento das aplicações do OBC, a utilização do RTEMS será detalhada na Seção 4.4.

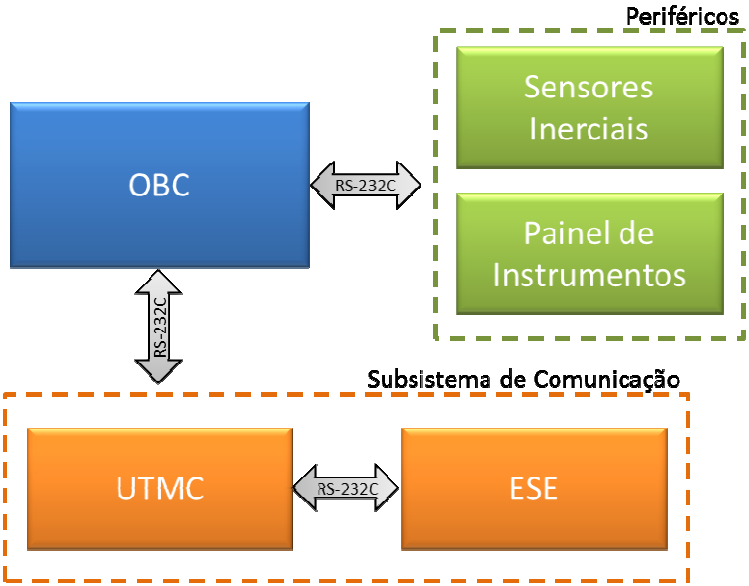
Como citado anteriormente, foram adicionados dois periféricos, com funcionalidades pertinentes aos instrumentos do OBC.

O bloco Sensores Inerciais é um periférico que possui dois sensores: um acelerômetro e um giroscópio. Ambos utilizam o protocolo SPI para enviar dados sensoriais. Um microcontrolador 8051 foi utilizado para receber dados dos sensores e enviar de forma serial para o LEON3. No microcontrolador, foram implementadas as funções do protocolo SPI para leitura dos sensores e do protocolo RS-232C para enviar os dados para o LEON3.

O Painel de Instrumentos utiliza um microcontrolador 8051 para simular a ocorrência de eventos no satélite. Basicamente, foram implementados dois eventos: LEDs de visualização da potência dos

propulsores; e um sinal sonoro indicando falha no sistema. Estes eventos possuem caráter fictício no funcionamento do satélite, porém foram idealizados visando agregar funcionalidades ao OBC e aumentar o número de possibilidades de Telecomandos disponíveis no projeto. A comunicação do processador com o microcontrolador é também através do protocolo serial RS-232C.

Figura 11 - Arquitetura Proposta

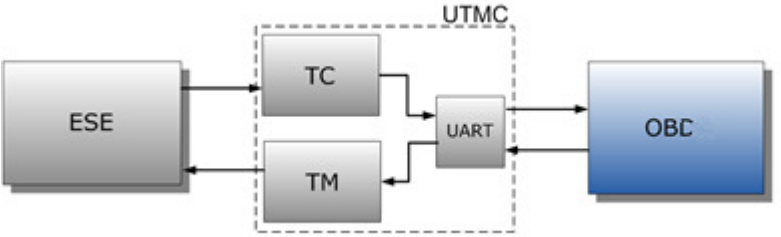


Fonte: (Silva, Ferreira, & Villa, 2009)

A interface do OBC com o subsistema de comunicação (bloco UTMC na Figura 11 e Figura 12) utiliza também o protocolo RS-232C. É importante ressaltar que a UTMC utilizada é o protótipo inicial do subsistema de comunicação que será empregado nas próximas missões do INPE. A UTMC irá se comunicar com o OBC e com o *Electrical Support Equipment* (ESE).

O ESE é um equipamento de teste desenvolvido em paralelo ao projeto do protótipo da UTMC; seu propósito é implementar rotinas de automação de testes para validação do protocolo que está sendo executado na UTMC. O ESE desempenha a função da Estação Terrestre. Portanto, o ESE foi utilizado para interagir com a UTMC enviando os Telecomandos e recebendo as Telemetrias.

Figura 12 - Fluxo de Comunicação



Fonte: (Silva, Ferreira, & Villa, 2009)

A Figura 12 representa o diagrama de fluxo da comunicação do projeto. Assumindo o ESE como ponto de partida da Comunicação, desempenhando a função da Estação Terrestre, os quadros de TC são montados e enviados para a UTMC. Na UTMC, o quadro passa pelas camadas da pilha de protocolos de TC, extraindo pacotes que são enviados para o OBC. O OBC irá desempacotar esses dados, e interpretar os comandos existentes na área de dados. A partir do comando, o OBC irá tomar uma atitude e enviará para a UTMC um pacote de TM. Na UTMC, esse pacote é encapsulado em um quadro de TM e enviado para o ESE, completando assim o fluxo de comunicação.

Após validação do fluxo de comunicação, realizada em (Silva, Ferreira, & Villa, 2009), buscou-se aperfeiçoar a arquitetura desenvolvida adicionando novos módulos de *software*. Os módulos de *software* desenvolvidos no presente trabalho executam outras funções importantes para o funcionamento do OBC e levam em consideração, os requisitos temporais exigidos nesse tipo de aplicação.

4.3. INTERFACE ENTRE SOFTWARE E HARDWARE DO COMPUTADOR DE BORDO

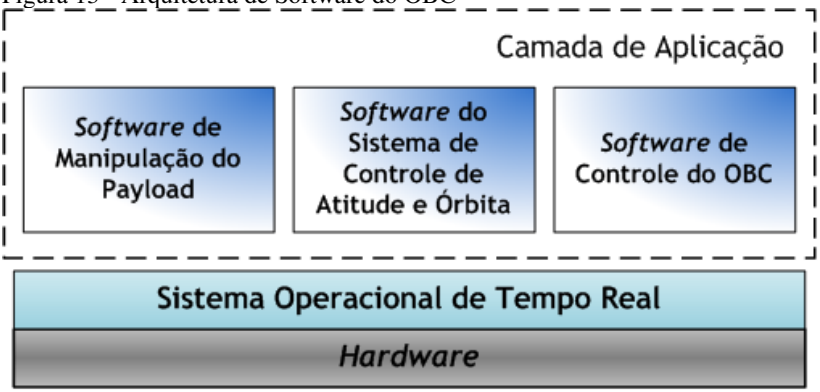
Os módulos de software têm como funções principais realizar a manipulação dos dados provenientes dos sensores, o controle de atitude e órbita e o controle de energia do satélite. Neste capítulo, será realizada uma descrição dos módulos que compõem o software do OBC.

Os componentes de software foram desenvolvidos visando à arquitetura de *hardware*, Seção 4.2, como suporte para a execução das aplicações. O principal requisito das aplicações é temporal. O Sistema Operacional (SO) deve ser capaz de cumprir os requisitos de tempo das aplicações. Por essa razão, a arquitetura do OBC deve ser projetada com

um Sistema Operacional de Tempo Real que seja capaz de gerenciar aplicações multitarefa.

A arquitetura do software do OBC é dividida em duas camadas: a Camada do Sistema Operacional e a Camada de Aplicação. A camada do SO faz referência às suas primitivas, funções nativas, estruturas, além de englobar as funções de acesso ao *kernel*, os *drivers* das interfaces de entrada e saída e outros serviços do Sistema Operacional. Já a Camada de Aplicação é onde se situam os módulos responsáveis pelo processamento e gerenciamento das informações, e integração com os demais subsistemas do satélite. A arquitetura de software do OBC está ilustrada na Figura 13.

Figura 13 - Arquitetura de Software do OBC



4.4. SISTEMA OPERACIONAL DE TEMPO REAL

Em aplicações críticas, que necessitem de um grau elevado de confiabilidade, é necessário que o sistema operacional garanta que todas as tarefas sejam cumpridas em um tempo pré-determinado, evitando que eventos críticos não sejam identificados e causem falhas na aplicação. A utilização de um sistema operacional de tempo real é importante devido à complexidade de um OBC, que necessita responder aos sensores, comandos e solicitações de acordo com prazos bem definidos, garantindo a integridade da aplicação.

Sistemas de tempo real caracterizam-se pela necessidade fundamental de manter um sincronismo constante entre as tarefas, isto é, o sistema deve atuar de acordo com a dinâmica de estados do processo, levando em conta os tempos de execução e as prioridades de cada tarefa (Malcolm & Zhao, 1994).

Assim, sistemas de tempo real não dependem somente do resultado lógico de computação, mas também do momento em que os resultados são produzidos, onde diversas tarefas são executadas e o escalonamento em função das restrições temporais é um grande problema. Tarefas recebem dados de entrada, executam um algoritmo e geram saídas. A tarefa está logicamente correta se gerar uma saída correta em função dos dados de entrada em um prazo temporal satisfatório (K & A., 1994). Devido a esses fatores, pode-se afirmar que, o tempo é o principal objetivo dos sistemas de tempo real, pois as tarefas executadas pelo processador devem ser concluídas antes de seu limite temporal (*deadline*).

O sistema operacional de tempo real utilizado nesse trabalho é o RTEMS (*Real Time Executive for Multiprocessor Systems*). O RTEMS é um sistema operacional de tempo real desenvolvido para sistemas embarcados. Foi criado no início dos anos 80, sendo a primeira versão comercial disponibilizada em 1993. Atualmente, o projeto RTEMS é gerenciado pela OAR *Corporation*. Esse sistema operacional é amplamente utilizado em aplicações espaciais, pois dá suporte a vários microprocessadores empregados nesse tipo de aplicação, entre eles: SPARC, ERC32, LEON, MIPS Mongoose-V, Coldfire e PowerPC. Como exemplo de utilização, é possível citar um dos satélites de reconhecimento que está orbitando o planeta Marte, o qual utiliza o RTEMS integrado ao módulo responsável pela comunicação. O satélite *Mars Reconnaissance Orbiter* (MRO) é uma espaçonave produzida e lançada pela NASA (*National Aeronautics and Space Administration*) com o intuito de reconhecer e explorar a órbita de Marte (Ely, Duncan, Lightsey, & Mogensen, 2006).

O RTEMS é um sistema operacional de código aberto (*open-source*) e existem diversos compiladores disponíveis para desenvolvimento de aplicações para o mesmo, entre eles, o sparc-gcc criado pela *Aeroflex Gaisler* para geração de código de máquina para o LEON3. Além disso, é importante ressaltar que o RTEMS é o sistema operacional especificado pelo INPE para utilização em seu OBC.

4.5. CAMADA DE APLICAÇÃO

A camada de aplicação é onde ficam as diretivas de software do OBC. Essas são responsáveis por executar todas as funções do OBC, utilizando como base, o sistema operacional de tempo real e a arquitetura de hardware. Como ilustra a Figura 13, existem três grupos principais que englobam as funções do OBC, são eles: Software de

Manipulação do *Payload*, Software do Subsistema de Controle de Atitude e Órbita e o Software de Controle do OBC.

O Software de Manipulação do *Payload* é o responsável por executar as funções referentes à carga útil do satélite. Como citado anteriormente, a carga útil é onde se encontram os sistemas dependentes de cada missão, ou seja, é a razão de ser do satélite. Assim, essa parte do *software* é totalmente dependente da missão específica de cada satélite. Exemplos de aplicações que podem ser executadas pelo Software de Manipulação de *Payload* são: o envio de informações ou dados referentes à carga útil para Estação Terrestre, processamento de dados necessários para carga útil, envio de informações sobre o funcionamento da carga útil para Estação Terrestre, entre outros.

O *Software* do Sistema de Controle de Atitude e Órbita é o responsável por verificar se o satélite está na atitude e órbita corretas. É o responsável por analisar os dados coletados dos sensores e gerar comandos para os atuadores. Em geral o *software* de controle de atitude e órbita AOC (*Attitude and Orbit Control)* é bastante complexo, pois deve, a fim de determinar a atitude, comparar a direção dos eixos do satélite com referências conhecidas e disponíveis, que podem ser as estrelas, o Sol, a Terra e o campo magnético terrestre e com isso gerar sinais para que os atuadores mantenham a atitude ou reposicionem o satélite. *Software* de AOC exige alto grau de precisão e é tema de diversos trabalhos científicos (Pinheiro, 2013) (Tagawa, 2013).

O *Software* de Controle do OBC possui três funções principais: controle do satélite, comunicação interna e processamento dos dados de bordo.

O controle do satélite deve administrar os modos de operação do satélite, analisar seus parâmetros a fim de detectar possíveis falhas e manter a Estação Terrestre informada sobre problemas existentes. É também responsável por executar os comandos enviados pela Terra em forma de TCs.

A comunicação interna do satélite diz respeito à comunicação entre os diversos subsistemas, sensores e atuadores e o OBC. É importante ressaltar a comunicação entre o OBC e o subsistema de comunicação, pois é o principal meio de envio de comandos da Estação Terrestre para o OBC.

O processamento de dados de bordo é quem mantém as estruturas de dados atualizadas com os últimos eventos do satélite, sejam dados de sensores, requisições de telemetria, dados de *Payload* ou dados sobre a saúde do satélite. Nos casos em que o OBC executa o *software* de AOC,

os dados devem ser processados e armazenados para futuras tomadas de decisões.

O *Software* de Controle do OBC foi considerado o principal módulo de *software* no presente trabalho. Entre os módulos descritos anteriormente é o único que não depende totalmente do tipo de missão do satélite ou de módulos externos de difícil aquisição. Suas principais funções podem ser utilizadas em uma grande variedade de missões, assim pode-se pensar em um OBC genérico que possa ser utilizado em diversas missões. Outro fator que desestimula o desenvolvimento dos demais módulos é sua complexidade. Seria necessário um amplo estudo sobre os algoritmos de AOC para que fosse possível integrar essa funcionalidade no presente trabalho, sendo que esse não é o foco principal da pesquisa realizada.

Os detalhes do software desenvolvido para o OBC no presente trabalho serão descritos nas próximas seções.

4.5.1. Componentes do Software de Controle OBC

O *Software* desenvolvido com o intuito de executar as funções de controle do OBC segue a relação de requisitos especificada pelo INPE (FUNDEP, 2011). Para facilitar o entendimento, o bloco *Software* de Controle do OBC (Figura 13) foi dividido em Componentes de *Software* (*Computer Software Components*, CSC). Cada CSC é responsável por executar um grupo de tarefas distinto que está diretamente ligado a um grupo de requisitos especificados pelo INPE.

A Figura 14 ilustra a divisão do bloco *Software* de Controle do OBC em CSCs, onde cada bloco na parte inferior da figura representa um CSC.

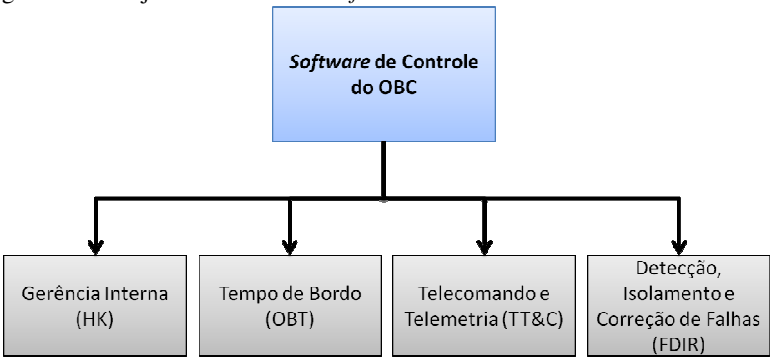
Cada CSC ilustrado Figura 14 é responsável por um grupo específico de funcionalidades que visa atender um grupo de requisitos de projeto especificado pelo INPE. Nas próximas seções, são descritas as funcionalidades de cada CSC.

4.5.1.1. Gerência Interna (*Housekeeping and Diagnose*, HK)

O HK é o CSC responsável pela gerencia das atividades do OBC. Tem como principal atribuição, a aquisição de dados de estado do satélite. Utiliza os dados para criação de relatórios sobre a saúde do mesmo; todas as informações de estado são armazenadas internamente. Assim, quando requisitado, esses dados podem ser enviados para a Estação Terrestre via TM. Entre as funções do HK estão:

- Prover meios para a aquisição dos seguintes parâmetros:
  - Configuração atual do OBC;
  - Tensões de alimentação internas ao OBC;
  - Temperaturas internas ao OBC;
  - Sinais de falha de suas unidades/módulos;
  - Dados analógicos de equipamentos e subsistemas do satélite;
  - Dados binários de equipamentos e subsistemas do satélite.
- Distribuição de comandos pulsados via RS232, para a execução de comandos ou em resposta a pedidos de serviços.

Figura 14 - Conjunto de CSCs do *Software* de Controle do OBC



4.5.1.2. Tempo de Bordo (*On Board Time*, OBT)

O OBT é o CSC responsável pelo controle de tempo do OBC. Como ilustrado na seção 4.4, esse tipo de aplicação exige elevado grau de controle dos requisitos de tempo, sendo necessária a utilização de um componente de *Software* dedicado ao controle temporal. As funcionalidades do OBT são:

- Contagem dos segundos do tempo de bordo. Utiliza um registrador de 4 bytes para armazenamento (por *software*) e sinais internos do processador para contagem (por *hardware*).
- Contagem dos sub segundos, ou *ticks* do processador, do tempo de bordo utilizando um registrador de 2 bytes.
- Utilização de um sinal de Pulsos por Segundo (PPS) gerado pelo *hardware* do OBC para controle de tempo.

- O registrador de segundos é incrementado pelo sinal PPS.
- A fonte de sinal PPS que incrementa o registrador de segundos é selecionável (interno ou externo).
- O registrador de sub segundos é incrementado por um sinal de relógio interno.
- O registrador de sub segundos é reinicializado a cada pulso do sinal PPS.

4.5.1.3. Telecomando e Telemetria (*Telemetry, Tracking & Command*, TT&C)

Componente de *Software* responsável pelo recebimento e envio de comandos para Estação Terrestre. Utiliza as interfaces com Subsistema de Comunicação para receber comandos e para envio de informações. Entre as funções do TT&C estão:

- Recepção dos pacotes de TC. O Subsistema de Comunicação deve extrair os pacotes de TC recebidos.
- Decodificação dos comandos contidos nos pacotes de TC.
- Validação dos comandos decodificados.
- Execução dos comandos validados.
- Envio de pacotes de TM para o Subsistema de Comunicação.
- Geração de relatórios de serviço sobre a aceitação de comandos a dispositivos contidos em pacotes de TCs válidos.
- Geração de relatórios de serviço sobre a rejeição dos comandos a dispositivos contidos em pacotes de TC inválidos.

4.5.1.4. Detecção de defeitos, Identificação e Reconfiguração (*Failure Detection, Identification and Reconfiguration*, FDIR)

FDIR é o CSC responsável pela indicação de falhas no OBC e demais Subsistemas. O FDIR faz monitoramento de todos os processos críticos de *Software*, das comunicações internas entre CSCs e comunicações entre o OBC e demais subsistemas. As funções do FDIR são detalhadas a seguir:

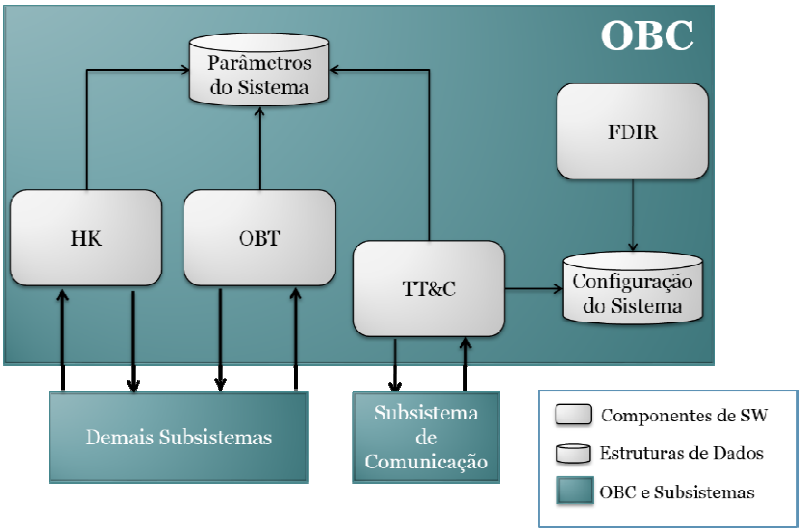
- Controle de tarefas críticas implementadas por *software*. Utiliza mecanismos de *watchdog* para reiniciar tarefas críticas que não estão sendo executadas corretamente.
- Controle de tempo de comunicações internas para detecção de falhas.

- Controle de tempo de comunicações entre o OBC e os demais subsistemas para detecção de falhas.
- Em caso de detecção de falhas, gera um sinal indicativo, para que o mesmo seja detectado externamente.
- Utiliza mecanismo de *watchdog* para verificar o recebimento do sinal de PPS externo.
- Os controles de falhas realizados pelo OBC são desabilitáveis e reabilitáveis individualmente.

4.5.1.5. Interfaces

A Figura 15 ilustra as interfaces internas desenvolvidas para o *Software* de Controle do OBC, assim como, as interfaces dos CSCs com os demais Subsistemas do Satélite. É possível identificar duas estruturas de dados utilizadas para controle das informações internas do OBC. As estruturas de dados são utilizadas como interface entre os diferentes CSCs.

Figura 15 - Interfaces entre os CSCs e demais Subsistemas



O CSC TT&C possui uma interface externa com o Subsistema de comunicação para troca de informações referentes à TCs e TMs. O CSC OBT possui uma interface externa com os demais Subsistemas do satélite para distribuição e recebimento do sinal de PPS. O CSC HK

possui uma interface externa para comunicação com os demais Subsistemas.

Com a implementação dos módulos de *software* descritos, buscou-se o desenvolvimento de uma plataforma genérica com capacidade de atender requisitos básicos para o funcionamento de um OBC. A plataforma descrita provê serviços básicos de comunicação interna entre os módulos do OBC, interfaces de comunicação entre os subsistemas, tratamento de comandos recebidos por TC e envio de informações por TM, controle temporal entre os módulos e controle de erros internos.

É importante ressaltar que o trabalho desenvolvido utilizou como base o modelo básico de OBC implementado em (Silva, Ferreira, & Villa, 2009). No modelo básico, foram desenvolvidos componentes básicos de *Software* para validação do fluxo de comunicação entre a UTMC e o OBC. No presente trabalho foram implementados os demais componentes de SW para controle do OBC, as estruturas de dados de controle e interfaces entre os componentes de SW.

5. MELHORIA DA CONFIABILIDADE DO COMPUTADOR DE BORDO

O computador de bordo proposto nesse trabalho visa atender requisitos especificados pelo INPE para suas próximas missões (FUNDEP, 2011). Para isso, foi levada em consideração a arquitetura especificada pelo INPE, e os respectivos requisitos funcionais (FUNDEP, 2011), tanto para o *software* quanto para o *hardware*. A maior diferença entre a arquitetura proposta no presente trabalho e a especificada pelo INPE é a utilização de FPGA SRAM para implementação do elemento central de processamento. No OBC atualmente em desenvolvimento pelo INPE, utiliza-se o processador ERC32 (Atmel Corporation, 2003). O ERC32, produzido pela Atmel, é um processador RISC de 32 bits tolerante a radiação. Na arquitetura aqui proposta, o *soft-processor* LEON3, que, diferentemente do ERC32, não é tolerante a radiação, está sendo utilizado. Ambos os processadores são compatíveis com a arquitetura SPARC V8, viabilizando a portabilidade das aplicações desenvolvidas.

Foi definido pelo INPE que a placa a ser desenvolvida para o OBC deve conter uma arquitetura redundante, ou seja, dois processadores devem ser utilizados. No OBC sendo desenvolvido pelo INPE estão sendo utilizados dois processadores redundantes ERC32, funcionando em paralelo em configuração *hot spare*. Utilizando essa configuração de redundância, toda a informação é processada simultaneamente por ambos os processadores, porém, somente a informação de um deles é utilizada. Assim, em caso de falha em um dos processadores, é possível utilizar a informação processada por sua cópia redundante, evitando que a falha se propague no sistema. Seguindo esse requisito para a placa do OBC, o presente trabalho visou à utilização de uma placa para o OBC com dois FPGAs redundantes. Como consequência, é possível que o erro em um dos FPGAs seja identificado em tempo de execução, permitindo que os dados processados pelo FPGA redundante sejam utilizados evitando que o erro se propague para o resto do sistema. A placa de protótipo atualmente utilizada conta com somente um FPGA, porém, o sistema visa utilização da placa com dois FPGAs.

A arquitetura proposta visa à simples de maneira *on-line* e em tempo de execução em cada um dos FPGAs. Assim, é possível que as falhas sejam mascaradas com utilização de duplicação de *hardware*.

Levando em consideração a arquitetura da GRLIB, é importante ressaltar que tanto o processador LEON3 quanto os outros IPs da

GRLIB são centrados no barramento AMBA. Ou seja, o barramento é a única forma de comunicação entre os módulos da GRLIB. Assim, a única maneira de uma falha ocorrida no LEON3 se propagar para o resto do sistema é através do barramento AMBA.

Para que seja possível identificar falhas em tempo de execução, foi utilizado o conceito de *controle de fluxo por observação do barramento do processador*. Uma cópia redundante e não intrusiva do *soft-processor* LEON3 foi adicionada ao barramento AMBA. Nesse caso o controle de fluxo consiste na comparação entre as saídas dos dois processadores. Quando as saídas são diferentes, uma falha é identificada e comunicada para o resto do sistema.

Um monitor de barramento foi desenvolvido para comparar o resultado dos processadores principal e redundante, para isso, o monitor verifica os dados que os processadores escrevem no barramento. Ambos os processadores respeitam o mesmo relógio, ou seja, funcionam de maneira síncrona e processam a mesma informação simultaneamente. Entretanto, as informações processadas pelo processador redundante não são escritas no barramento, por esse motivo é chamado de não intrusivo. As informações do processador redundante são utilizadas apenas pelo monitor de barramento para verificar se as informações do processador principal e redundante são iguais. A Figura 16 ilustra a arquitetura proposta com o processador redundante não intrusivo e o monitor de barramento.

Como citado anteriormente, caso o monitor de barramento identifique uma falha em um dos processadores, esse deve informar o erro para o resto do sistema. Para isso o monitor de barramento utiliza o sinal "*controlflow_error*", como ilustra a Figura 16.

A utilização da arquitetura aqui proposta permitirá que a escolha entre as informações supridas pelo FPGA principal e redundante na placa do OBC ocorra de maneira automática por um circuito seletor. O circuito seletor é responsável por selecionar as informações geradas pelo FPGA livre de falhas e repassar para os subsistemas do OBC. A Figura 17 ilustra a placa do OBC com os dois FPGAs utilizando a arquitetura proposta no presente trabalho, e o circuito seletor.

Figura 16 - Arquitetura proposta com processador redundante

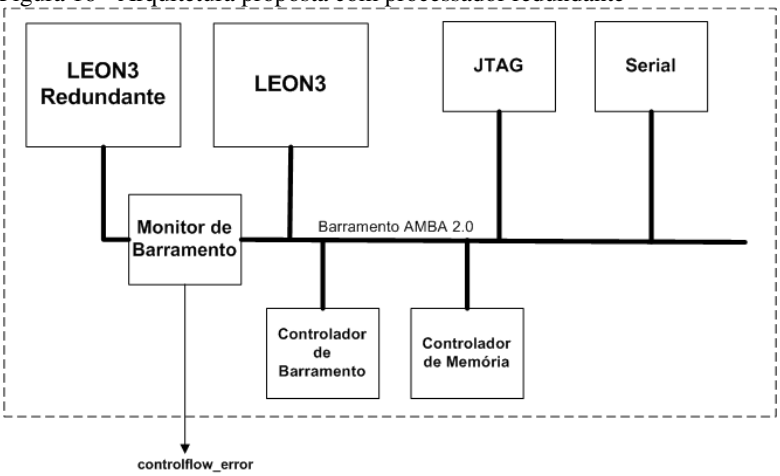
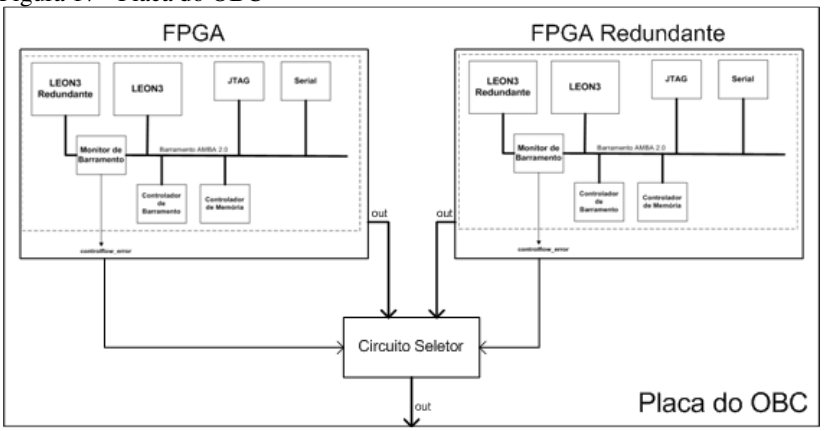


Figura 17 - Placa do OBC



5.1. TRIPLICAÇÃO DO MONITOR DE BARRAMENTO E VOTADOR DE MINORIA

O principal objetivo de técnicas que utilizam redundância para tolerância a falhas é eliminar pontos únicos de falha. Ou seja, pontos que podem gerar defeitos no sistema em caso de falha. No caso da técnica aqui proposta, foram utilizados dois processadores LEON3, assim caso

ocorra uma falha em um deles o valor errôneo será detectado quando comparado com o valor provido pelo processador livre de falhas. Porém, da maneira como foi descrita, a técnica aqui proposta gerou um novo ponto único de falha.

Caso uma falha ocorra no monitor de barramento, causando a inversão de seu valor lógico, tanto um erro inexistente pode ser sinalizado para o sistema, assim como a indicação de um erro existente pode ser mascarada. Para evitar que o monitor de barramento seja um ponto único de falhas é necessário que o módulo seja replicado. Porém, seria necessário fazer com que os caminhos lógicos redundantes, providos da replicação do monitor de barramento, voltem para somente um caminho lógico na saída do FPGA sem que um novo ponto único de falha seja criado. Isto é possível com aplicação da técnica de seleção de saídas utilizado em configurações TMR (*Triple Modular Redundancy*) com três votadores (Azevedo, 2010).

Primeiramente, para aplicar a técnica, é necessário que o monitor de barramento seja triplicado. Assim, é possível mascarar falhas ocorridas em até uma das três cópias redundantes utilizando um votador entre as saídas dos três módulos. A seleção entre as saídas dos três monitores de barramento é construída utilizando votadores de minoria combinados com *3-state buffers* (TRIBUFF). O sinal de saída de cada caminho lógico passa por um TRIBUFF antes de chegar ao pino de saída do FPGA. O TRIBUFF é responsável por habilitar a saída do caminho lógico. Caso a saída do TRIBUFF seja desabilitada o sinal é colocado em alta impedância. A habilitação do TRIBUFF é gerada pelo votador de minoria, que compara a saída dos três monitores de barramento. A Figura 18 ilustra o funcionamento da técnica de seleção de saída com três votadores. Importante ressaltar essa técnica foi utilizada devido à disponibilidade dos componentes *3-state buffers* no FPGA utilizado, facilitando assim, a aplicação da técnica.

Para cada um dos caminhos lógicos foi adicionado um votador de minoria. O votador de minoria indica quando o caminho lógico em questão (caminho primário P) está em concordância com pelo menos um dos caminhos lógicos redundantes (R1 e R2). Se o caminho primário possui o mesmo valor que algum dos caminhos redundantes, então o caminho primário faz parte da maioria, e o sinal de habilitação do TRIBUFF é ativado. Se o caminho primário não possui o mesmo valor que nenhum dos dois caminhos redundantes, então o caminho primário faz parte da minoria e a saída do TRIBUFF é desabilitada. A Figura 19 ilustra o circuito votador de minoria onde P representa o caminho lógico

primário e R1 e R2 os caminhos lógicos dos monitores de barramento redundantes.

Figura 18 - Seleção de saída com três votadores

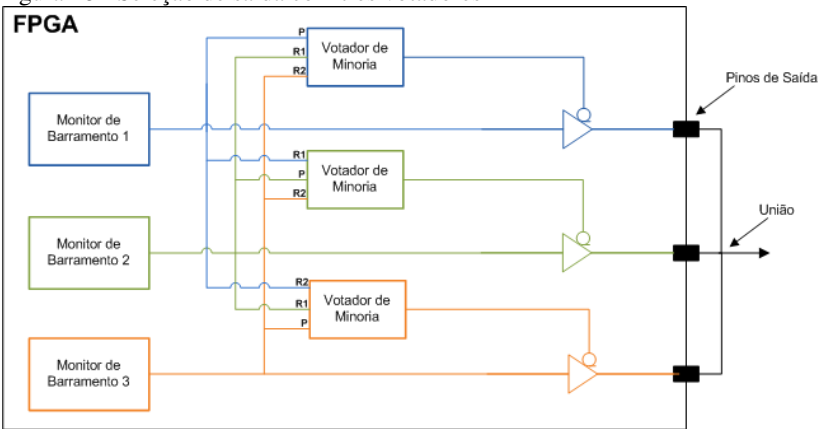
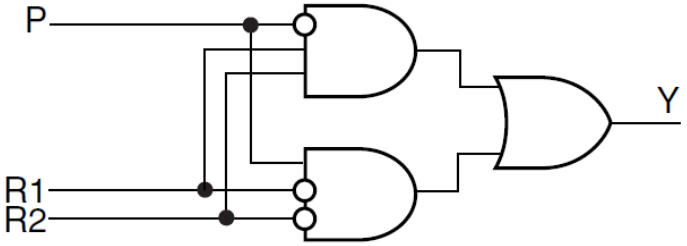


Figura 19 - Circuito Votador de Minoria



Fonte: (Azevedo, 2010)

A Tabela 1 ilustra todos os estados possíveis para a configuração com três monitores de barramento utilizando a técnica de seleção de saída com três votadores. As colunas MB1, MB2 e MB3 representam à saída dos monitores de barramento, que tem seu valor igual a “1” quando um erro foi encontrado. As colunas VM1, VM2 e VM3 representam a saída dos votadores de minoria, esses assumem o valor “0” para habilitar a saída do sinal sendo votado, e “1” para desabilitar sua saída. A coluna saída representa o resultado final após a união dos pinos do FPGA.

É importante ressaltar que os três pinos de saída do FPGA aos quais estão ligadas as saídas dos TRIBUFFs devem ser interligados externamente, fazendo com que os caminhos lógicos voltem a ser um só. Essa estrutura não causa nenhum tipo de estado transiente controverso, visto que somente as saídas que concordam entre si são diretamente conectadas e as saídas que discordam são colocadas em alta impedância. A grande vantagem deste método é que nenhum circuito adicional, externo ao FPGA, é necessário para unificar as três saídas.

Tabela 1 – Tabela verdade da seleção de saída com votadores de minoria

MB1	MB2	MB3	VM1	VM2	VM3	SAÍDA
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	1
1	0	0	1	0	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	1
1	1	1	0	0	0	1

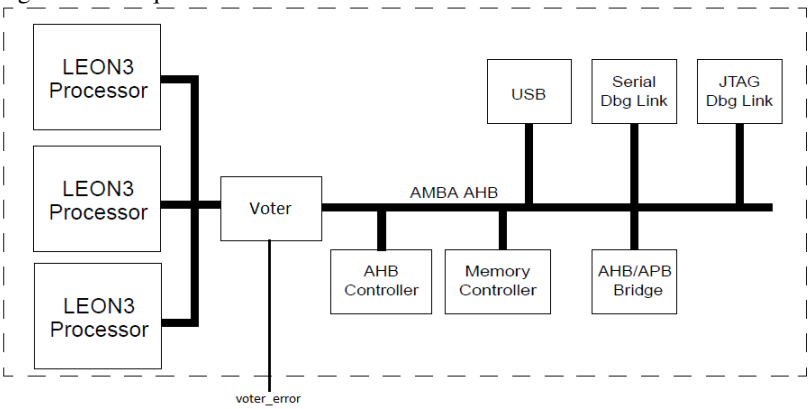
5.2. COMPARAÇÃO COM OUTRAS TÉCNICAS DE TOLERÂNCIA A FALHAS EM FPGAS

A utilização da técnica de tolerância a falhas proposta no presente trabalho mostra-se bastante eficiente para a arquitetura do computador de bordo. Quando comparada a outras técnicas encontradas na literatura (Azevedo, 2010), a utilização de um processador redundante não intrusivo demonstrou algumas vantagens. Com apenas dois processadores redundantes ao invés dos três utilizados com a técnica TMR, o aumento de área no FPGA sofre uma redução considerável. Como resultado, não há necessidade de utilização de FPGAs com maior disponibilidade de área. Os quais usualmente são mais caros e não contam com versões tolerantes a radiação. Além disso, em algumas aplicações a utilização de TMR pode gerar um aumento de área tão significativo a ponto de não haver FPGAs comerciais grandes o suficiente para comportá-lo.

Para demonstrar a diferença entre as duas técnicas, foi desenvolvida uma versão do OBC utilizando TMR com um votador. A técnica apresenta como principal problema a ocupação de área em

FPGA, pois o processador deve ter uma versão principal e duas redundantes. A grande diferença da técnica é a utilização de um votador. Esse deve comparar os resultados gerados pelos três processadores e selecionar o resultado correto, ou seja, o que foi calculado por dois ou mais processadores, caso os três resultados sejam diferentes um sinal de erro é gerado pelo votador. Assim como na técnica com um processador redundante, os valores calculados pelos processadores são verificados em seus barramentos. A Figura 20 ilustra a arquitetura do OBC com TMR e votador.

Figura 20 – Arquitetura do OBC com TMR e votador



Os relatórios de síntese listados nas figuras a seguir ilustram a utilização do FPGA do kit de desenvolvimento ML403 da Xilinx, em três casos: a Figura 21 ilustra a arquitetura original do OBC sem nenhuma técnica de tolerância a falhas; a Figura 22 ilustra a arquitetura utilizada no presente trabalho, com redundância simples e monitor de barramento; a Figura 23 ilustra a arquitetura do OBC utilizando TMR, com triplicação do processador e votador.

Ambas as técnicas de tolerância a falhas tem um aumento significativo de ocupação do FPGA. Comparando as técnicas com a versão original do OBC, temos que:

- A técnica com redundância simples causou um aumento de aproximadamente 154% de ocupação do FPGA, utilizando cerca de 94% dos recursos disponíveis.
- A utilização de TMR causou um aumento de aproximadamente 228%. É importante ressaltar que a utilização de TMR se tornou inviável para o FPGA

utilizado, pois mais de 100% dos recursos disponíveis seriam necessários.

A utilização do votador de minoria, eliminando um ponto único de falhas na arquitetura, agrega mais uma vantagem em comparação a TMR. Com a utilização do votador da TMR, é possível que uma falha única, causada por radiação, altere o valor lógico do sinal "*vote_error*", indicando que as três cópias redundantes do processador LEON3 apresentaram resultados distintos. A utilização do votador de minoria elimina esse tipo de problema, pois mesmo sob efeito de falhas únicas, o funcionamento do votador de minoria não é afetado.

Outra vantagem da técnica proposta é a manutenção do *software* executado no processador. Como a técnica utiliza somente recursos de *hardware,* para implementar a tolerância a falhas, não há necessidade de modificações no *software*. Técnicas que adicionam instruções especiais ao programa em execução para implementar tolerância a falhas (*Software Implemented Hardware Fault Tolerance*, SIHFT) (Piotrowski, Makowski, Jablonski, & Napieralski, 2008) podem prejudicar o desempenho do processador, que gasta ciclos de relógio na execução dessas instruções.

Figura 21 – Utilização do FPGA sem técnicas de tolerância a falhas

```
Device utilization summary:
---------------------------

Selected Device : 4vsx35ff668-10

 Number of Slices:                     9434  out of  15360    61%
 Number of Slice Flip Flops:           6248  out of  30720    20%
 Number of 4 input LUTs:              16456  out of  30720    53%
    Number used as logic:             16169
    Number used as Shift registers:      39
    Number used as RAMs:                248
 Number of IOs:                         257
 Number of bonded IOBs:                 254  out of    448    56%
    IOB Flip Flops:                     322
 Number of FIFO16/RAMB16s:               26  out of    192    13%
    Number used as RAMB16s:              26
 Number of GCLKs:                        16  out of     32    50%
 Number of DCM_ADVs:                      5  out of      8    62%
 Number of DSP48s:                        4  out of    192     2%
```

Figura 22 – Utilização do FPGA com um processador redundante e monitor de barramento

```
Device utilization summary:
---------------------------

Selected Device : 4vsx35ff668-10

 Number of Slices:                    14507  out of  15360    94%
 Number of Slice Flip Flops:           9450  out of  30720    30%
 Number of 4 input LUTs:              25415  out of  30720    82%
    Number used as logic:             25095
    Number used as Shift registers:      72
    Number used as RAMs:                248
 Number of IOs:                         257
 Number of bonded IOBs:                 254  out of    448    56%
    IOB Flip Flops:                     322
 Number of FIFO16/RAMB16s:               46  out of    192    23%
    Number used as RAMB16s:              46
 Number of GCLKs:                        16  out of     32    50%
 Number of DCM_ADVs:                      5  out of      8    62%
 Number of DSP48s:                        8  out of    192     4%
```

Figura 23 – Utilização do FPGA com TMR

```
Device utilization summary:
---------------------------

Selected Device : 4vsx35ff668-10

 Number of Slices:                    21292  out of  15360   138% (*)
 Number of Slice Flip Flops:          13531  out of  30720    44%
 Number of 4 input LUTs:              37283  out of  30720   121% (*)
    Number used as logic:             36690
    Number used as Shift registers:     105
    Number used as RAMs:                488
 Number of IOs:                         257
 Number of bonded IOBs:                 254  out of    448    56%
    IOB Flip Flops:                     321
 Number of FIFO16/RAMB16s:               66  out of    192    34%
    Number used as RAMB16s:              66
 Number of GCLKs:                        16  out of     32    50%
 Number of DCM_ADVs:                      5  out of      8    62%
 Number of DSP48s:                       12  out of    192     6%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used
```

6. FERRAMENTAS E AMBIENTE DE DESENVOLVIMENTO

Nesse capítulo são abordados os detalhes de implementação do sistema, os métodos adotados para seu desenvolvimento bem como as dificuldades encontradas. Nas seções a seguir, é ilustrado como cada componente do sistema foi desenvolvido e os passos necessários para que o processo seja reproduzido.

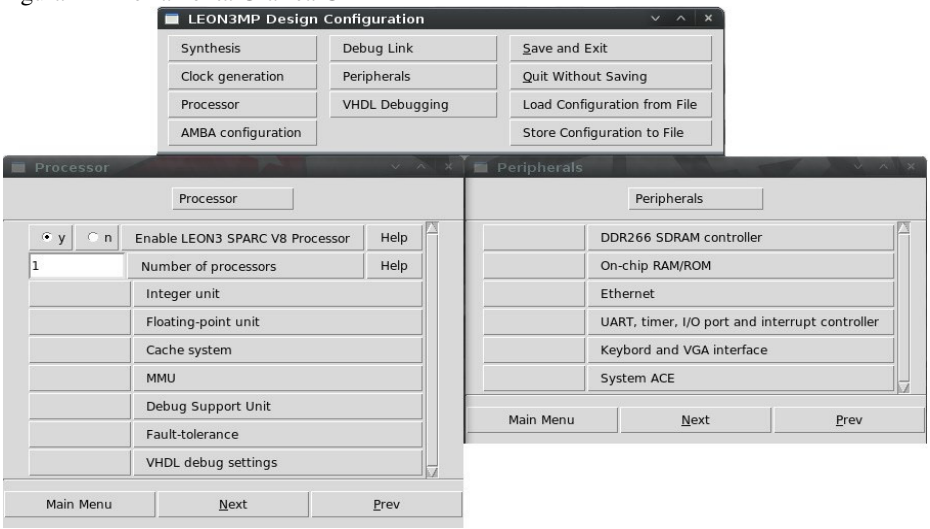
6.1. PROCESSADOR

Conforme descrito na Seção 4.1, o processador selecionado para o projeto foi o LEON3, da empresa Aeroflex Gaisler que fornece uma série de recursos para utilização desse processador. O modelo do processador LEON3, juntamente com a biblioteca de IPs (GRLIB), é sintetizável para uma grande variedade de FPGAs. A biblioteca GRLIB consiste em uma variedade de IPs reutilizáveis portados para desenvolvimento em SoC, suportando uma grande variedade de ferramentas de CAD e FPGAs.

A preparação do ambiente de desenvolvimento de sistemas baseados no processador LEON3 inclui a detenção e instalação da biblioteca GRLIB, utilizando uma distribuição do sistema operacional Linux. O arquivo "grlib-gpl-1.0.20-b3403.tar.gz", que pode ser obtido gratuitamente, deve ser extraído em uma pasta qualquer. Com os arquivos extraidos, deve-se acessar a pasta grlib-gpl-1.0.20-b3403/designs/leon3-digilent-xup, pasta com o projeto para a placa da utilizada nesse trabalho. A GRLIB suporta diversas placas de desenvolvimento, caso a placa utilizada não seja a leon3-digilent-xup, deve-se selecionar a placa desejada.

Antes de realizar a síntese do VHDL é necessário configurar o projeto (*design*), para isso a GRLIB conta com uma ferramenta gráfica onde podem ser selecionadas as configurações da arquitetura, adição e remoção de periféricos, configuração da quantidade de processadores entre outros. Essa ferramenta de configuração é basicamente em um script que altera o arquivo “config.vhd”, responsável pela configuração do sistema como um todo, caso seja necessário o arquivo pode ser modificado manualmente. A Figura 24 ilustra a ferramenta gráfica de configuração.

Figura 24 - Ferramenta Gráfica GRLIB



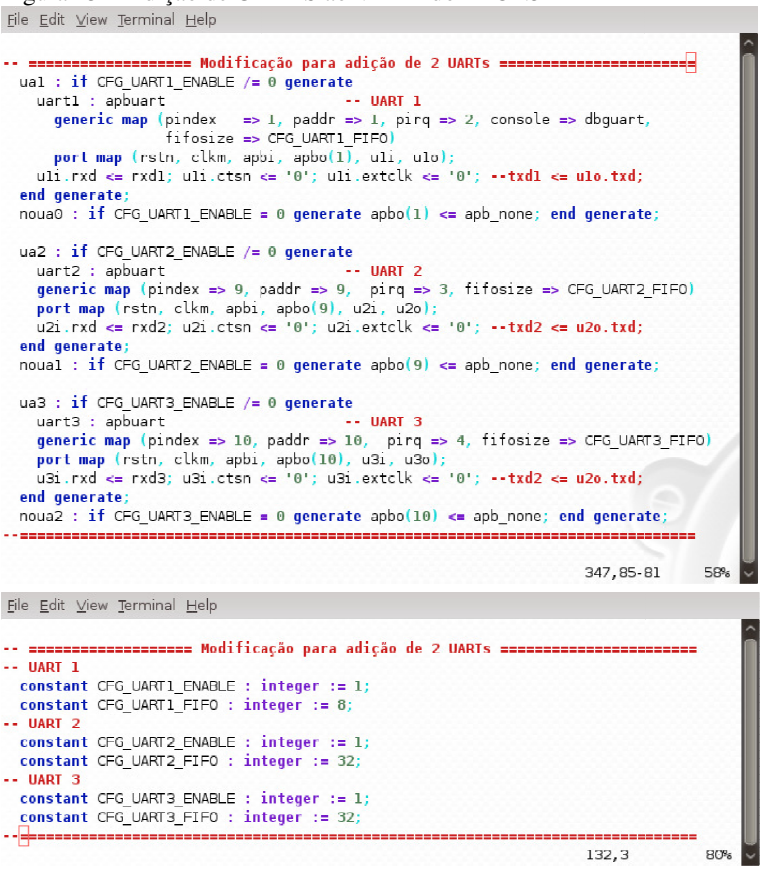
Nas configurações padrão do projeto do LEON3 utilizado nesse trabalho estão disponíveis os seguintes periféricos:

- Controlador para SDRAM DDR266;
- Ethernet MAC, UART e
- Interface PS2

Como a placa XUP disponibiliza apenas um conector DB9, conector padrão para comunicação serial RS-232C, o design do processador para a placa de desenvolvimento não permite adicionar novas UARTs através da ferramenta de configuração ilustrada na Figura 24. Na arquitetura proposta, são necessárias três portas seriais (três UARTs): uma para possibilitar ao LEON3 o recebimento de informações dos sensores inerciais (placa com 8051); uma para envio de comandos para os instrumentos (painel de instrumentos); e outra para comunicação com a UTMC por onde o LEON3 irá receber Telecomandos e enviar Telemetrias. Dessa forma, foi necessário adicionar outras duas UARTs diretamente no código VHDL.

O código VHDL do processador LEON3 foi alterado adicionando-se as novas UARTs ao barramento AMBA-2.0 AHB/APB, em endereços livres e com posições próprias no vetor de interrupção. Os trechos do código VHDL alterado para que as novas UARTS funcionem corretamente é ilustrado na Figura 25.

Figura 25 – Adição de UARTS ao VHDL do LEON3



Os pinos de transmissão e recepção das UARTs adicionadas foram mapeados nos conectores de expansão da XUP. Como os pinos de expansão fornecem níveis de tensão TTL (5 volts) foi necessário utilizar um componente que converte os níveis de tensão de TTL para RS-232C, ou seja, de +12v para 0v e -12v para +5v. Assim é necessário utilizar dois componentes conversores de tensão, um para cada UART adicionada manualmente.

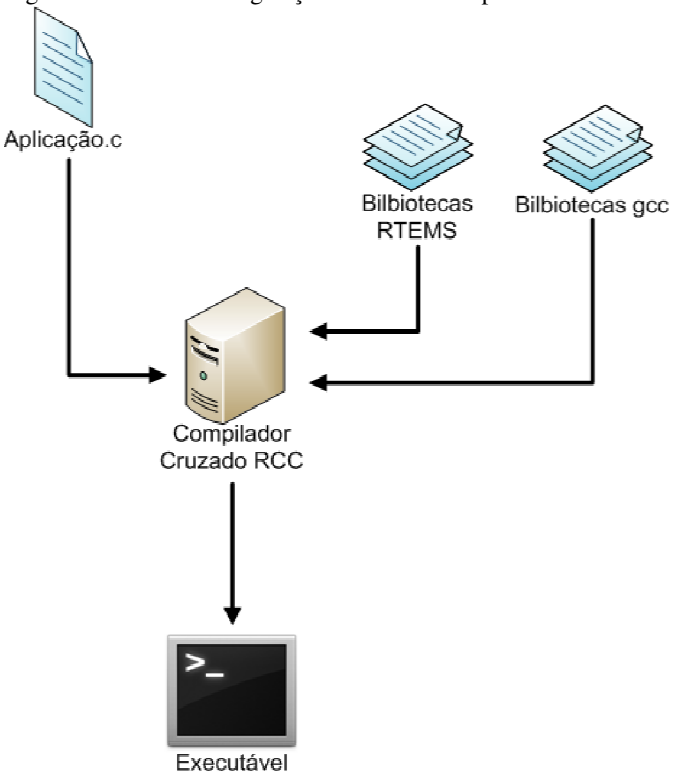
6.2. SISTEMA OPERACIONAL

O RTEMS Cross Compiler system (RCC) é um sistema de desenvolvimento multiplataforma que gera, entre outros, executáveis

para os processadores das famílias LEON e ERC32 com as diretivas de tempo real do sistema operacional RTEMS. Por se tratar de um compilador cruzado permite gerar códigos executáveis para o LEON3 utilizando outro sistema operacional, como por exemplo, o Linux.

Com um compilador próprio baseado no gcc, o RCC permite que códigos escritos em linguagem de programação C/C++ sejam compilados utilizando bibliotecas padrão para Linux e algumas bibliotecas exclusivas para o RTEMS. Os executáveis gerados pela ferramenta podem ser executados diretamente no processador LEON3, pois já incluem as diretivas do sistema operacional RTEMS, ou seja, a aplicação em linguagem C é compilada e ligada aos módulos do RTEMS gerando um código único em linguagem de máquina do processador, dispensando a instalação do sistema operacional. O processo de geração do código executável da aplicação, integrado ao RTEMS está ilustrado na Figura 24.

Figura 26 - Processo de geração do executável pelo RCC



Por se tratar de um sistema operacional de tempo real o RTEMS exige que sejam definidas algumas configurações das tarefas executadas no sistema. Configurações como, por exemplo, número máximo de tarefas executadas, tamanho máximo de pilha das tarefas, número de descritores de arquivo gerenciados, entre outras.

O compilador cruzado RCC é distribuído de forma gratuita sob a licença GNU e pode ser obtido no site da Gaisler. O RCC funciona em conjunto com a GRLIB facilitando o desenvolvimento deste trabalho. Devem-se seguir os passos descritos a seguir para instalação e utilização do RCC no sistema operacional Linux.

1. Extrair o arquivo de instalação do RCC na pasta "/opt/rtems-4.10"
    a. # cd /opt
    b. # bunzip2 -c sparc-rtems-4.10-gcc-4.3.3-1.1.99.x-linux.tar.bz2 | tar xf -

2. Para ter acesso direto as ferramentas disponíveis deve-se adicionar o diretório "/opt/rtems-4.10/bin" ao caminho de busca de executaveis:
    a. # export PATH=$PATH:/opt/rtems-4.10/bin

3. Na pasta "/opt/rtems-4.10/src/samples" estão disponíveis uma série de códigos fonte para o RTEMS. Os exemplos podem ser utilizados para garantir que o LEON3 está funcionando perfeitamente.

4. Para compilar os códigos fontes deve-se executar os seguintes comandos:
    a. # cd /opt/rtems-4.10/src/samples
    b. # make

5. É necessário que o caminho do compilador "sparc-rtems-gcc" seja adicionado a variável de ambiente PATH, como ilustrado no Passo 2, para que o Makefile de compilação dos arquivos exemplo funcione corretamente.

6.3. TESTE E DEPURAÇÃO DO LEON3

Para efetuar os testes e depuração do software executado pelo LEON3 (SO e Aplicação) foi utilizada a ferramenta GRMON, distribuída de forma gratuita pela mesma empresa responsável pela GRLIB.

O GRMON é um monitor de depuração para os processadores da família LEON e para sistemas embarcados baseados na biblioteca GRLIB. A depuração do sistema embarcado no FPGA é possível devido à capacidade do GRMON de se comunicar com a unidade de suporte a depuração (Debug Support Unit - DSU) do LEON3, permitindo depuração não intrusiva do sistema.

Com essa ferramenta é possível adicionar executáveis criados com o RCC ao LEON3 e executá-los; visualizar as configurações do processador e os periféricos conectados ao barramento AMBA; permite conexão remota com o depurador GDB através das interfaces: Serial, Ethernet, JTAG, PCI e USB; tornando possível que o sistema seja depurado de maneira confiável.

Para os fins desse trabalho foi utilizado o cabo de testes JTAG para configurar o FPGA com o *bitstream* contendo a imagem do processador, para envio dos executáveis do sistema e depuração do mesmo. Os drivers do cabo JTAG utilizados na instalação do Xilinx ISE não funcionam na maioria das distribuições Linux, pois são específicos para o Red Hat Enterprise Linux. Portanto, é necessário instalar um driver USB alternativo que pode ser obtido na internet. Os seguintes passos devem ser seguidos para efetuar a instalação:

1. Antes de instalar os drivers é necessario ter a biblioteca libusb-dev instalada, para isso utilizou-se o comando:
   ```
   # apt-get install libusb-dev
   ```

2. Após, deve-se extrair os arquivos do driver para a pasta onde a ferramenta ISE está instalada e executar o comando make para efetuar a instalação:
   ```
   # tar xzf usb-driver-HEAD.tar.gz -C /opt/Xilinx/10.1
   # cd /opt/Xilinx/10.1/usb-driver
   # make
   ```

3. Com o comando "ls" verificamos se o driver foi instalado:
   ```
   # ls libusb-driver.so
   ```

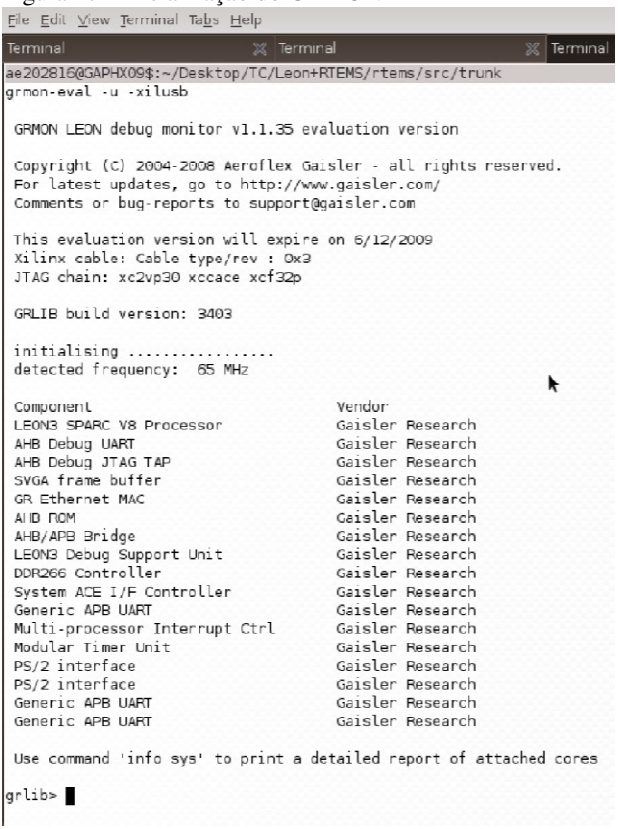
4. Para testar o dispositivo deve-se utilizar o comando lsusb e verificar se o endereço é igual ao ilustrado a seguir, ou seja, que o firmware inicial do dispositivo foi iniciado corretamente:
   # lsusb | grep "Xilinx"
   # Bus 006 Device 017: ID 03fd:0008 Xilinx, Inc.

5. Finalizando é necessário configurar a seguinte variavél de ambiente antes de executar o GRMON:
   # export LD_PRELOAD=/opt/Xilinx/10.1/usb-driver/libusb-driver.so

Quando a placa de desenvolvimento é ligada é necessário enviar o *bitstream* com a imagem previamente compilada do processador LEON3 ao FPGA, para isso foi utilizado o software Impact.

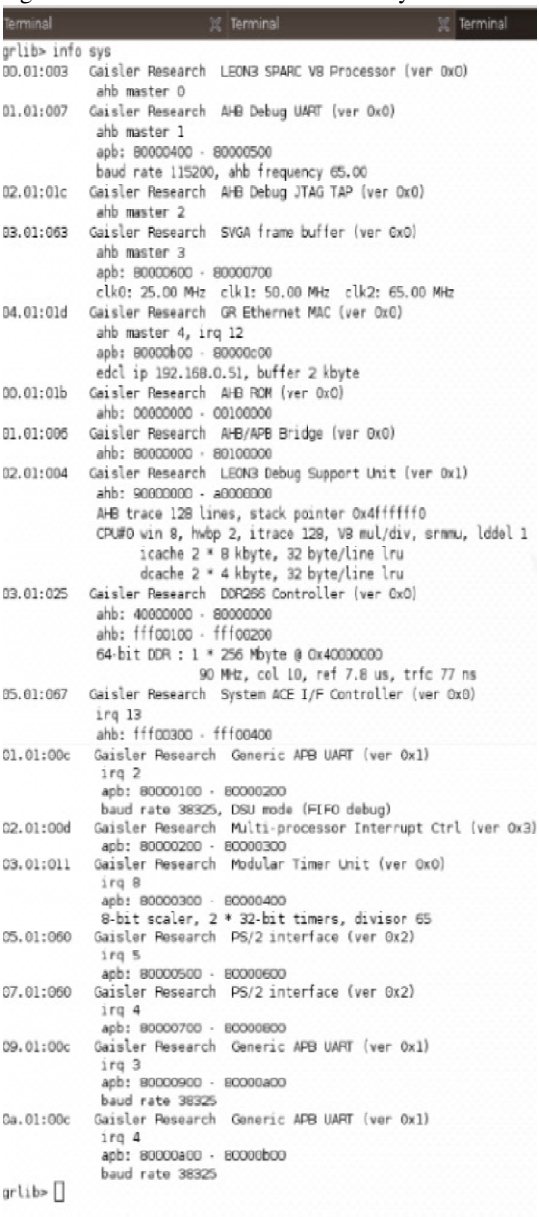
Após instalar todos os componentes e *drivers* necessários, é possível executar o software GRMON. O GRMON permite verificar algumas configurações do processador e os periféricos conectados ao barramento AMBA-2.0 AHB/APB. Utilizando o cabo de testes JTAG, o GRMON se conecta com a unidade de depuração do processador (DSU) permitindo acesso não intrusivo ao processador. A Figura 27 demonstra o funcionamento inicial do GRMON, com as informações sobre o JTAG, conexão com a DSU e os principais módulos e componentes encontrados no barramento. Assim, é possível verificar que o processador embarcado no FPGA está funcionando corretamente.

Figura 27 - Inicialização do GRMON



O GRMON disponibiliza alguns comandos para verificar informações sobre o processador, entre eles, o de maior relevância para os fins desse trabalho é o “info sys”. Esse comando retorna informações detalhadas sobre os módulos conectados ao processador como, por exemplo: endereços no barramento de dados, endereços no vetor de interrupção, relógios utilizados pelo componente, buffers, entre outros. A Figura 28 demonstra o retorno da função “info sys”. Vale ressaltar que as UARTs adicionadas para os fins desse trabalho aparecem na lista de dispositivos, com os endereços de barramento previamente definidos pelo grupo.

Figura 28 - Retorno do comando "infosys"



7. METODOLOGIAS DE TESTE ADOTADAS

Em aplicações críticas e de alta complexidade como o OBC, é importante garantir que o sistema atende todas as funcionalidades requeridas. No presente trabalho, foram aplicados testes que tem como objetivo verificar o funcionamento correto do *software* desenvolvido e também, a capacidade da arquitetura modificada de identificar falhas em tempo de execução. Devido às modificações na arquitetura descritas no Capítulo 5, é necessário que os testes comprovem o funcionamento da arquitetura tolerante a falhas e que mostrem que as modificações não afetaram o funcionamento do OBC.

Os testes executados no presente trabalho são divididos em dois grupos: Testes de *software* e Simulação Funcional.

7.1. TESTE DE SOFTWARE

Os Testes de Software do OBC foram executados de maneira a cobrir os requisitos especificados para cada um dos CSCs ilustrados na Seção 4.5.1. Devido à ausência dos demais subsistemas e periféricos utilizados em um OBC, foram utilizadas rotinas de *software* e componentes disponíveis no FPGA para validar alguns dos requisitos.

O fluxo de Telecomando e Telemetria, executado pelo CSC TT&C, foi o mais exaustivamente testado. Devido à reutilização dos componentes de *software* e do fluxo de informações utilizados na Arquitetura Inicial (Seção 3.2), os testes previamente executados são considerados suficientes para validação do CSC.

O Fluxo de TC e TM foi validado utilizando o OBC conectado ao Subsistema de Comunicação (Unidade de Telemetria e Telecomando, UTMC), realizando o fluxo de dados no satélite. O ESE foi utilizado para geração dos TCs e interpretação das TMs, fazendo o papel da Estação Terrestre, como ilustrado na Figura 12.

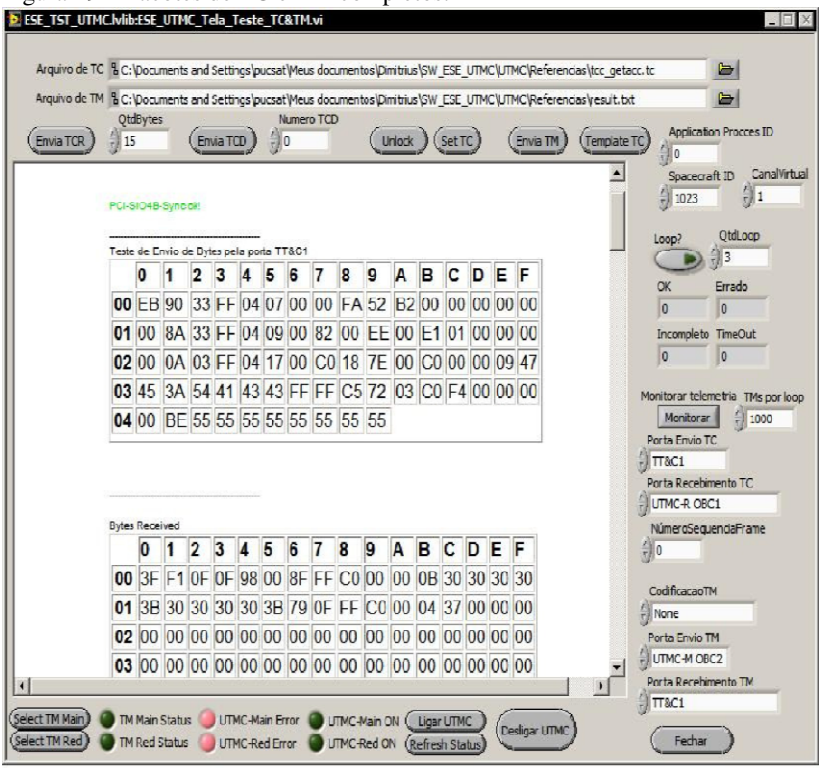
A comunicação entre o OBC e a UTMC foi considerada válida no momento em que os pacotes de Telecomando e Telemetria foram reconhecidos pelo OBC e pelo ESSE, respectivamente. Os Telecomandos utilizados foram criados com auxílio da ferramenta de geração de Telecomandos disponibilizada pelo INPE, como ilustrado em (Silva, Ferreira, & Villa, 2009).

Assim, é possível que sejam enviados Telecomandos previamente definidos a partir do ESE e que as Telemetrias recebidas sejam verificadas pelo mesmo. O funcionamento dos TCs pode ser visualmente comprovado utilizando comandos para acionamento dos

componentes do Painel de Instrumentos, com acionamento dos LEDs e Buzzer disponíveis. Os Telecomandos de aquisição de dados dos sensores foram validados através do ESE, ou seja, as Telemetrias com os dados dos sensores são exibidas na tela do ESE sendo possível verificar os dados providos pelos sensores na área de dados do pacote de Telemetria. Todos os Telecomandos utilizados no presente trabalho foram exaustivamente enviados, garantindo o funcionamento proposto.

A Figura 29 demonstra uma das telas do *Labview* (software utilizado no ESE) com o sistema completo em funcionamento.

Figura 29 – Pacotes de TC e TM completos.



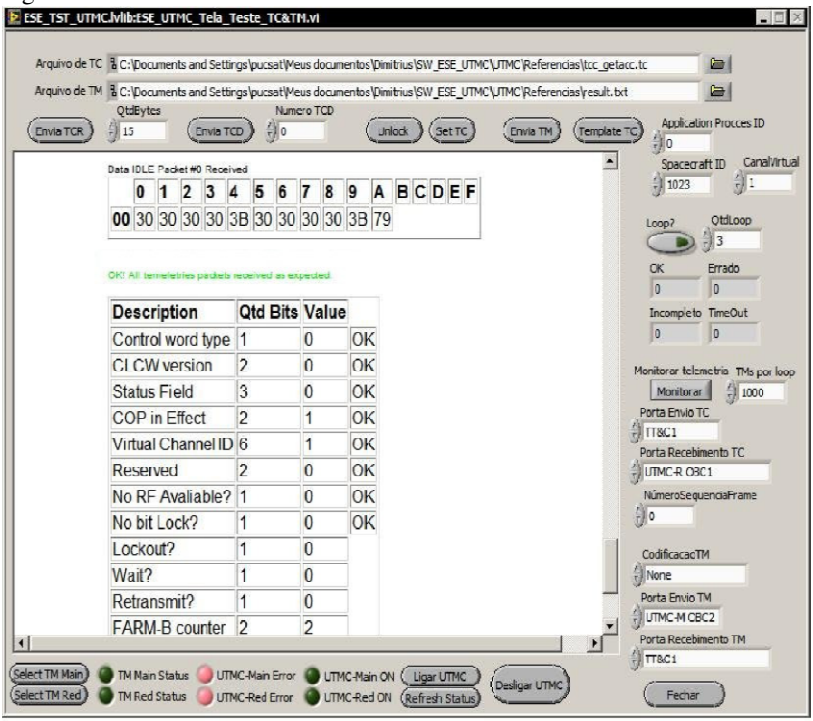
Fonte: (Silva, Ferreira, & Villa, 2009)

No primeiro campo de dados, é demonstrado o Telecomando enviado pelo ESE, contendo os campos de cabeçalho e controle utilizados pela UTMC e a área de dados e cabeçalho que será enviada para o OBC. No segundo campo, é demonstrada a Telemetria recebida,

que novamente passou pela UTMC e teve seus campos de cabeçalho e controle adicionados antes de retornar ao ESE.

O campo de dados da Telemetria tem maior relevância para depuração do OBC, pois nele estão contidos os dados providos pelos sensores. Na Figura 30, ilustra-se o campo de dados da Telemetria que contém as informações de inclinação requeridas pelo Telecomando.

Figura 30 - Área de Dados da TM.



Fonte: (Silva, Ferreira, & Villa, 2009)

O controle de tempo de bordo (OBT) foi realizado utilizando registradores de tempo disponíveis no Sistema Operacional. O RTEMS disponibiliza contadores de pulso de relógio por segundo, que são utilizados no controle das diretivas de tempo real. Os mesmos contadores foram utilizados para executar as funções do OBT. Como especificado nos requisitos do OBT (Seção 4.5.1.2) o contador de PPS é selecionável, sendo possível utilizar uma fonte externa para o mesmo. Um pino externo do FPGA utilizado com fonte externa do PPS, sendo

que a seleção entre PPS externo ou interno é feita através de Telecomando.

A gerencia interna (HK) provê acesso às estruturas de dados que contém informações de saúde do OBC; as estruturas podem ser requisitadas via Telecomando. Entre as informações disponíveis na estrutura estão:

- Registradores de segundos do OBT
- Registradores de sub segundos do OBT
- Fonte selecionada para PPS (interno ou externo)
- Indicação de erro dos módulos de *software* (FDIR, OBT, HK e TT&C)
- Indicação de *watchdog* dos módulos de *software* (FDIR, OBT, HK e TT&C)
- Indicação de *timeout* de comunicações (UTMC, Painel de Instrumentos e PPS externo)
- Desativação dos *watchdogs*

Os dados de indicação de erros, *timeouts* e *watchdog*, providos na estrutura de dados, são atualizados pelo CSC FDIR. Para que o CSC fosse validado, cada um dos módulos controlados foi desativado por *software*, forçando a invocação de cada um dos *watchdogs* e indicação dos *timeouts* de comunicação. Todos os eventos de falha foram devidamente indicados na estrutura de dados. Um LED foi utilizado para validar indicação externa de falha do OBC.

Os Testes de *Software* (Seção 7.1) demostram que as modificações na arquitetura do LEON3 não alteraram o funcionamento do OBC. Assim, pode-se considerar que na ausência de eventos externos, como a incidência de SEE, o OBC está funcionando corretamente. O Teste de *Software* utilizou o sistema completo para o teste. Sendo esse formado pelo *Software* do OBC juntamente com as primitivas do sistema operacional RTEMS, e as primitivas de *Hardware* do processador LEON3, ambos embarcados no FPGA.

7.2. SIMULAÇÃO FUNCIONAL

Em aplicações desenvolvidas utilizando lógica programável, é necessário que o circuito lógico resultante da síntese do VHDL passe por fases de simulação que garantam que o funcionamento é o esperado. Usualmente, somente após as fases de Simulação Funcional, onde o circuito lógico é avaliado, e Simulação Temporal, onde os atrasos de

cada uma das unidades lógicas do circuito são considerados, o circuito está pronto para testes em FPGA.

Existem diversas ferramentas que permitem a realização de simulação funcional em circuitos lógicos gerados a partir de VHDL. Essas ferramentas permitem a inserção de estímulos ao circuito, assim como monitoramento de todos os sinais internos do mesmo. No presente trabalho, a ferramenta utilizada para realizar a Simulação Funcional *foi o Modelsim* (Mentor Graphics Corporation, 2010).

Com o intuito de avaliar a técnica de detecção de falhas utilizada no trabalho, é necessário que o funcionamento do circuito seja afetado por um evento externo, que simule a incidência de SEE. Em outras palavras, é necessário que uma falha seja injetada no circuito. Utilizando a capacidade de inserção de estímulos nos sinais internos do circuito provida pelo *Modelsim*, é possível alterar o estado de um sinal interno simulando a incidência de um SEE.

Com os resultados da simulação do OBC, sem injeção de falhas, previamente armazenados, é possível repetir a simulação injetando uma falha em um dos sinais, e comparar os valores de ambas as simulações. Assim, é possível analisar as alterações comportamentais do OBC sob influência de uma falha. Como definido na Seção 3.3, existem diferentes maneiras de manifestação das falhas no sistema: Silencioso, Latente e Erro. Quando ocorre um Erro, ou seja, a falha injetada se propagou para pelo menos uma saída do circuito, é necessário que o Monitor de Barramento identifique o Erro e indique-o para o sistema evitando que o mesmo gere um defeito.

Para que seja possível analisar todos os possíveis cenários de ocorrência de SEE, é necessário que se analise o comportamento do circuito sob influência de SEE em cada um dos sinais internos. Para o caso do LEON3 existem 1090 sinais internos que podem ter seu valor alterado em simulação. Os sinais internos podem ser sinais simples, representados por um *bit,* e também vetores de sinais simples, que podem ser representados por até 64 *bits*. É necessário executar o cenário completo de simulação para cada um dos sinais alterados, resultando em 1090 simulações completas do programa de testes. Para facilitar a execução de todos os casos de teste, foi desenvolvido um *script* de simulação. O *script* automatiza a injeção de falhas e a análise do comportamento do circuito ao final da simulação.

O *script* de simulação utiliza a linguagem Tcl juntamente com o *Modelsim*. Assim, é possível acessar as funções de controle de simulação providas pelo *Modelsim* diretamente do *script*. Para que seja possível simular diferentes modelos de injeção de falhas, o *script* utiliza

as funções "*force*" e "*noforce*" providas pelo *Modelsim*. A função "*force*" pode alterar o valor de qualquer um dos valores internos do circuito em qualquer tempo de simulação, o sinal sob influência do "*force*" não pode ter seu valor alterado até a execução da função "*noforce*". A flexibilidade das funções "*force*" e "*noforce*" permite que diferentes modelos de falhas sejam utilizados. Controlando o tempo em que o sinal está sob influência do "*force*", é possível simular diferentes cenários de SEU, assim como FP.

O funcionamento do *script* segue o seguinte fluxo:

1. Execução da simulação do OBC sem injeção de falhas;
2. Ao final da simulação, todos os valores dos sinais internos do processador são salvos em um arquivo de texto;
3. Declaração de um monitor para o sinal "*controlflow_error*". Qualquer alteração no estado do sinal é armazenada em um arquivo de log;
4. Formação de uma lista com todos os sinais internos do processador. Para cada um dos sinais, os seguintes passos são executados:
    a. Reinicialização da Simulação;
    b. Simulação é executada até o momento em que se deseja inserir a falha.
        i. O tempo mínimo é 80us, tempo necessário para inicialização do componente *Hardware* e início da execução do *Software* do OBC;
    c. Valor do sinal sob teste é capturado;
    d. Valor do sinal sob teste é alterado, causando a injeção da falha. Para alterar o valor, utiliza-se a função "*force*";
    e. Simulação é executada pelo período definido no modelo da falha;
    f. Sinal é liberado da influência da falha, utilizando a função "*noforce*";
        i. Caso a falha sendo injetada seja um FP, a etapa "f" não é executada.
    g. Simulação é executada até o final;
    h. Todos os valores dos sinais internos do processador são capturados em um arquivo de texto;

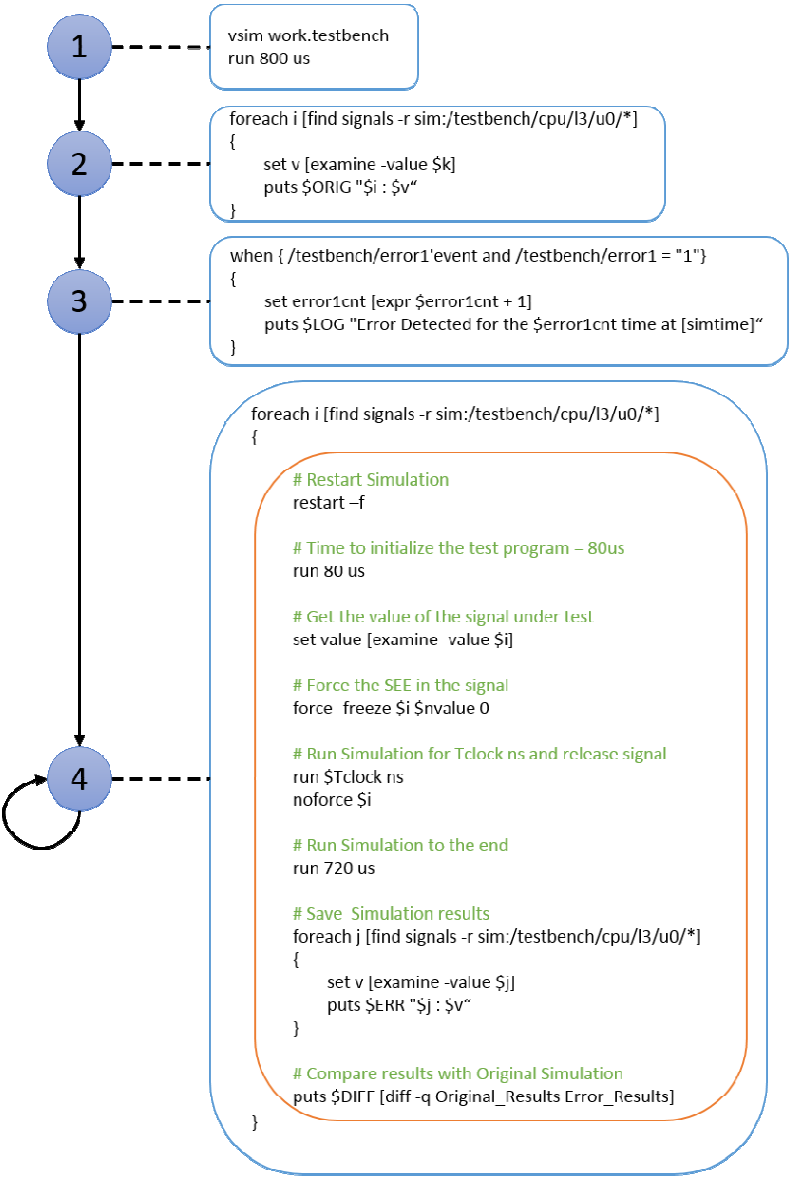
i. Os valores dos sinais (após a simulação com e sem injeção de falhas) são comparados. Os resultados são armazenados em um arquivo de log.
j. Caso ainda existam sinais a serem testados, repetir passo 4 com o próximo sinal da lista.

A Figura 31 representa o fluxo de execução do *script* de simulação ilustrando os principais comandos executados. Os números em destaque, referem-se à sequência de passos descrita anteriormente.

Diversas técnicas de injeção de falhas por simulação são descritos em outros trabalhos (da Silva & Sanchez, 2010) (Ziade, Ayoubi, & Velazco, 2004), e poderiam ser aplicadas no presente trabalho. Porém, todas as técnicas apresentam o mesmo problema, ou seja, o tempo necessário para que a simulação seja concluída. Projetos complexos que utilizam centenas de portas lógicas representam uma quantidade enorme de possibilidades de falha, que devem ser levadas em consideração. No caso da arquitetura do OBC descrita no presente trabalho, a simulação de 800 ms (utilizando um *clock* de 100 MHz), tempo de execução do programa de teste, leva cerca de 9 minutos. Levando em consideração que a simulação deve ser executada em cada um dos 1090 casos de teste, o tempo total é de cerca de 164 horas, ou cerca de 7 dias. No presente trabalho, foi analisada a ocorrência de SEUs em três diferentes tempos de execução do programa de testes, e também utilizando FP como modelo de falhas. Assim, todos os casos de teste foram simulados quatro vezes, sendo que o tempo necessário para execução de todas as simulações foi de aproximadamente quatro semanas. Os resultados de simulação serão apresentados no Capítulo 1.

Devido a todos esses fatores, o *script* de simulação mostrou-se essencial para que a técnica de tolerância a falhas fosse avaliada. Seria inviável avaliar todos os casos de teste executados sem a automatização de injeção de falhas e análise de resultados proporcionada pelo *script*.

Figura 31 - Fluxo de execução do *script* de simulação



8. RESULTADOS OBTIDOS E DISCUSSÃO

Os resultados obtidos com a simulação de injeção de falhas demonstram o comportamento do OBC sob influência de SEE. Analisando os resultados, é possível verificar as alterações comportamentais causadas pelos SEE em cada um dos módulos do processador, assim como verificar que a técnica apresentada no presente trabalho foi capaz de identificar qualquer propagação de falha injetada para os pinos externos do processador.

Durante a execução do programa de testes, diferentes áreas do processador são utilizadas em momentos também diferentes. Por exemplo, o módulo do processador responsável pela multiplicação é ativado somente para execução de instruções que exijam multiplicação. Assim, a injeção de uma falha enquanto o bloco não está sendo utilizado pode não afetar o funcionamento normal do processador. Portanto, é necessário que a injeção da falha em cada módulo ocorra enquanto o mesmo está sendo ativado pelo programa de testes. Com o intuito de aumentar o número de erros oriundos da injeção de falhas, repetiu-se a simulação alterando o tempo na qual a falha é injetada.

Quatro modelos de falhas foram utilizados:

- FP: a injeção da falha ocorre no início da execução do programa de testes (80 ciclos de relógio) e o sinal não pode ser sobrescrito durante toda a simulação;
- SEU 80: a injeção da falha ocorre no início da execução do programa de testes (80 ciclos de relógio), após um ciclo de relógio o sinal pode ser sobrescrito;
- SEU 300: a injeção da falha ocorre durante a execução do programa de testes (300 ciclos de relógio), após um ciclo de relógio o sinal pode ser sobrescrito;
- SEU 600: a injeção da falha ocorre na parte final da execução do programa de testes (600 ciclos de relógio), após um ciclo de relógio o sinal pode ser sobrescrito.

Para validar os tempos utilizados para injeção de falhas é necessário que os módulos do processador sejam analisados separadamente. Assim, é possível verificar os efeitos causados pela injeção de falhas nos diferentes tempos de simulação. A descrição comportamental do processador LEON3 consiste de cinco módulos principais:

- Núcleo de Processamento (*Processor Core*, P0);
- Memória *Cache* (*Cache Memory*, CMEM);

- Unidade de Ponto Flutuante (*Floating Point Unit*, FPU);
- Registrador de três portas (*3-Ports Register File,* RF0), é a entidade responsável pelo acesso à memória RAM;
- *Buffer* de Rastreamento (*Trace Buffer Memory*, TBMEM), buffer que armazena informações sobre a comunicação entre a CPU e o barramento AMBA.

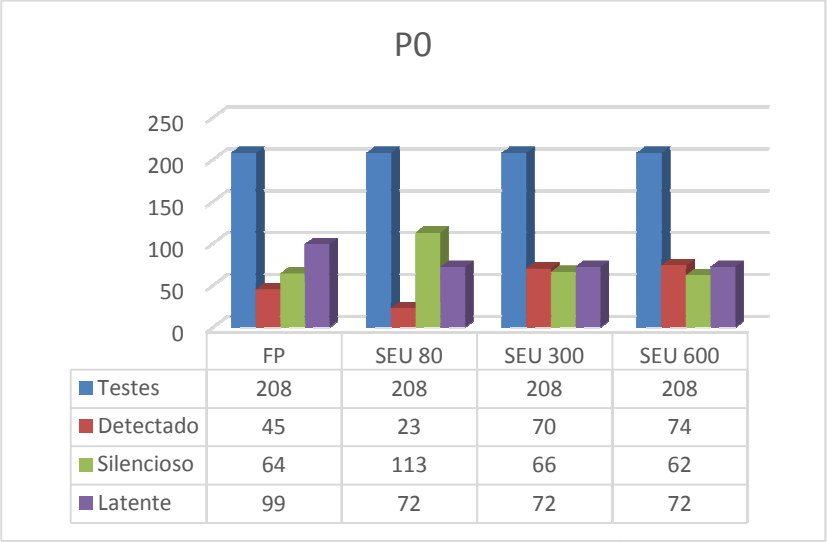
A seguir, são apresentados os resultados obtidos nas simulações com injeção de falhas em cada um dos módulos do processador. Os gráficos ilustram os resultados obtidos em cada um dos modelos de falha descritos. Para cada um dos modelos de falhas, as seguintes informações são providas:

- Testes: Número de testes executados. Todos os sinais internos do módulo são testados.
- Detectado: Número de erros gerados pela injeção de falhas que foram detectados pelo Monitor de Barramentos. Todos os erros gerados foram detectados.
- Silencioso: Injeção da falha foi sobrescrita e não alterou nenhum valor interno do modulo.
- Latente: A falha injetada alterou sinais internos do modulo, porém a falha não foi propagada para nenhuma saída do processador.

A Figura 32 ilustra os resultados de simulação com injeção de falhas no Núcleo do Processador. Analisando o gráfico é possível identificar que a injeção de falhas em 600 ciclos de relógio foi a que gerou maior número de erros, cerca de 35,5% das falhas injetadas geraram um erro. No modelo de injeção de falhas em 80 ciclos de relógio, cerca de 54,3% dos testes geraram um sinal Silencioso. Isso se deve ao fato de grande parte dos sinais do módulo serem sobrescritos após a inicialização.

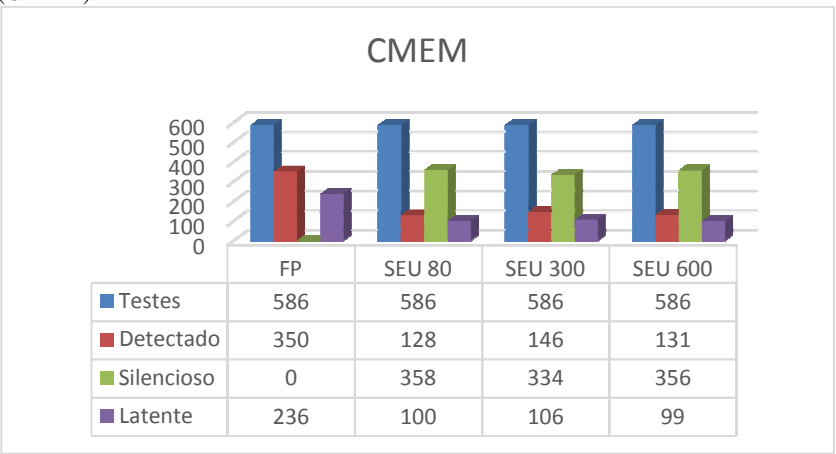
A Figura 33 ilustra os resultados de simulação com injeção de falhas na Memória *Cache*. Devido às características das Memórias *Cache*, que usualmente tem maior número de acessos para leitura e escrita que os demais módulos, pode-se verificar que a maioria (cerca de 56%) das falhas injetadas foram sobrescritas, tornando-se Silenciosas. No caso do FP, onde a sobrescrita não é possível, a mesma proporção de falhas, cerca de 56%, foram propagadas para alguma saída e identificadas pelo monitor de barramentos.

Figura 32 - Resultados de Simulação com Injeção de Falhas no Núcleo do Processador (P0)



	FP	SEU 80	SEU 300	SEU 600
Testes	208	208	208	208
Detectado	45	23	70	74
Silencioso	64	113	66	62
Latente	99	72	72	72

Figura 33 - Resultados de Simulação com Injeção de Falhas na Memória *Cache* (CMEM)



	FP	SEU 80	SEU 300	SEU 600
Testes	586	586	586	586
Detectado	350	128	146	131
Silencioso	0	358	334	356
Latente	236	100	106	99

A Figura 34 ilustra os resultados de simulação com injeção de falhas na Unidade de Ponto Flutuante. A Unidade de Ponto Flutuante foi

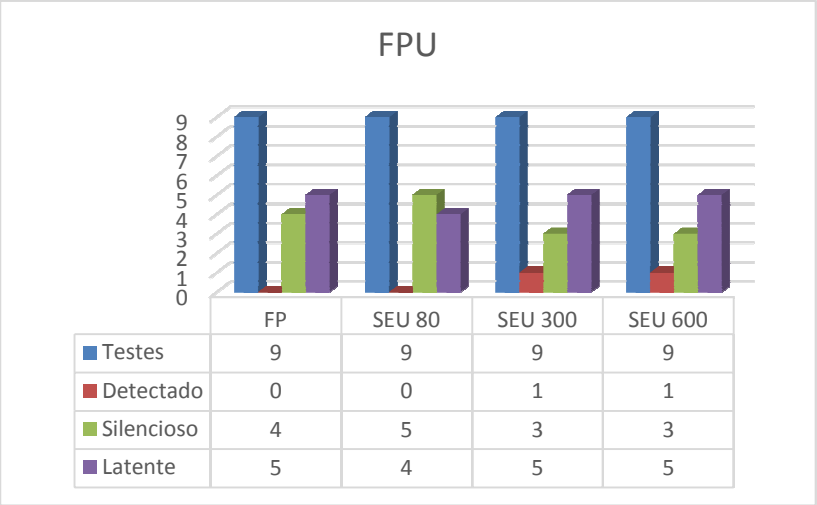
o módulo que menos propagou erros para as saídas do processador. Como o módulo é menos utilizado no início da execução do programa de testes, a propagação de um erro foi detectada somente nos modelos que injetaram a falha após a metade da execução do programa de testes.

A Figura 35 ilustra os resultados de simulação com injeção de falhas no Registrador de três portas. Por se tratar de um controlador de acesso a memória, o módulo RF0 apresenta algumas características próximas às apresentadas pelo CMEM. Porém, devido a menor frequência de acesso a memória RAM quando comparada com a *Cache*, o efeito de mascaramento das falhas devido à sobrescrita torna-se ainda mais evidente. O mesmo efeito reflete-se no modelo de injeção de falhas FP, onde devido a não possibilidade de sobrescrita, cerca de 53% das falhas foram propagadas para fora do processador.

A Figura 36 ilustra os resultados de simulação com injeção de falhas no *Buffer* de Rastreamento. Os dados de controle armazenados no *Buffer* de Rastreamento não são externados pelo programa de testes. Assim, todas as falhas injetadas no modulo foram Silenciosas ou Latentes.

A Figura 37 ilustra os resultados de simulação com injeção de falhas em todos os módulos. Analisando os resultados, é possível verificar o comportamento do processador de maneira geral.

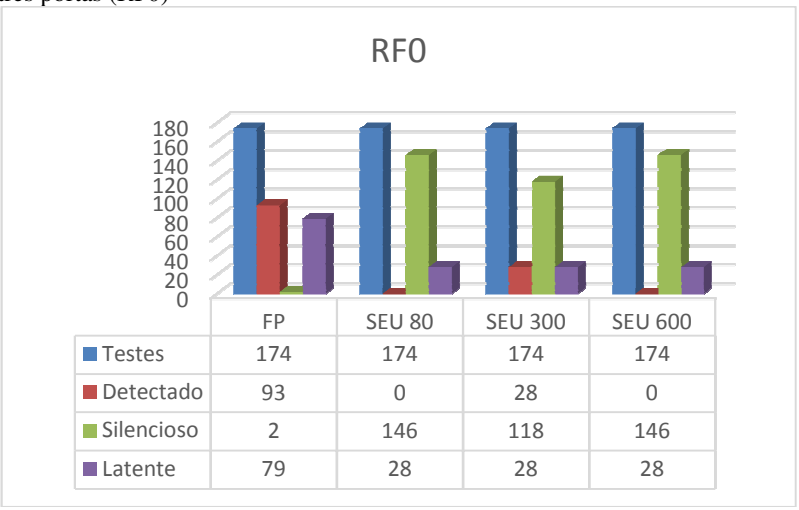
Figura 34 - Resultados de Simulação com Injeção de Falhas na Unidade de Ponto Flutuante (FPU)



	FP	SEU 80	SEU 300	SEU 600
Testes	9	9	9	9
Detectado	0	0	1	1
Silencioso	4	5	3	3
Latente	5	4	5	5

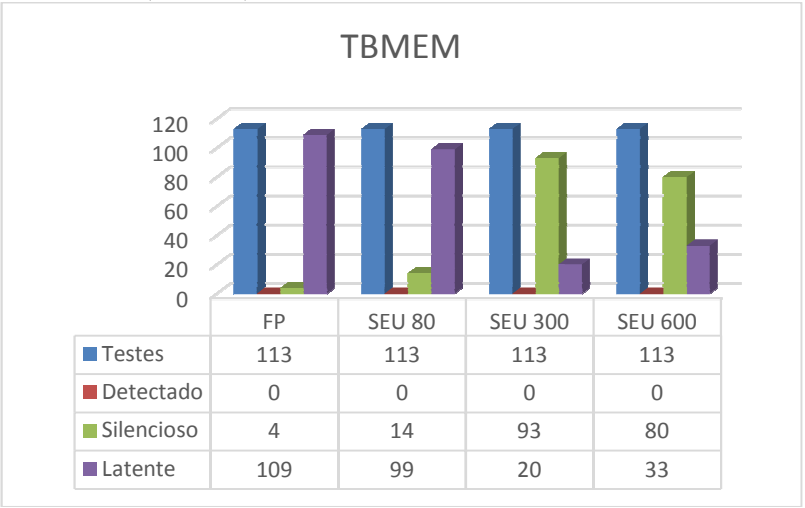
Com os resultados combinados é possível notar uma discrepância no número de erros gerados pelo modelo de injeção FP, quando comparado com os demais modelos. Utilizando FP, cerca de 44,7% das falhas injetadas geraram erros, enquanto os outros modelos apresentaram no máximo cerca de 22%. Isso se deve a característica do FP, que não permite que o valor escrito seja sobrescrito, diminuindo a chance de mascaramento da falha injetada.

Figura 35 - Resultados de Simulação com Injeção de Falhas no Registrador de três portas (RF0)



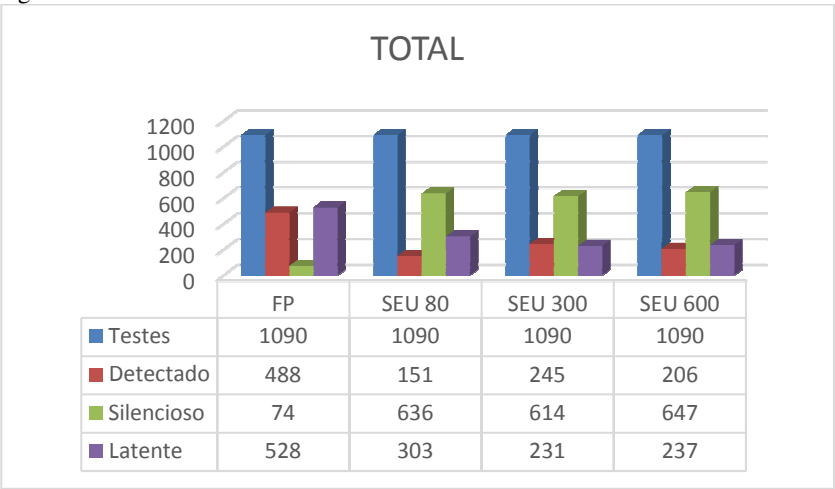
	FP	SEU 80	SEU 300	SEU 600
Testes	174	174	174	174
Detectado	93	0	28	0
Silencioso	2	146	118	146
Latente	79	28	28	28

Figura 36 - Resultados de Simulação com Injeção de Falhas no *Buffer* de Rastreamento (TBMEM)



	FP	SEU 80	SEU 300	SEU 600
Testes	113	113	113	113
Detectado	0	0	0	0
Silencioso	4	14	93	80
Latente	109	99	20	33

Figura 37 - Resultados Combinados



	FP	SEU 80	SEU 300	SEU 600
Testes	1090	1090	1090	1090
Detectado	488	151	245	206
Silencioso	74	636	614	647
Latente	528	303	231	237

Dentre os modelos de injeção de falhas do tipo SEU, que permite sobrescrita do valor, nota-se que o maior número de erros detectados ocorreu a 300 ciclos de relógio. Por estar no meio da execução do programa de testes, onde mais recursos do processador são utilizados, a injeção de falhas a 300 ciclos de relógio tem maiores chances de afetar um módulo em uso no processador, gerando assim um erro. Utilizando os resultados de todos os modelos de falhas aplicados, cerca de 31,5% das falhas geraram um erro. Levando em consideração apenas os modelos de injeção de falhas do tipo SEU, maior preocupação nesse tipo de aplicação, cerca de 18,3% das falhas injetadas propagaram-se para alguma saída do processador. É importante ressaltar que em todos esses casos o monitor de barramento foi capaz de identificar a propagação da falha e indicar para o sistema que um erro havia ocorrido. Em uma aplicação real, utilizando a indicação de erro gerada pelo monitor de barramento, seria possível mitigar a falha causada pelo SEU sem que a mesma se propagasse para o resto do sistema.

Devido ao modelo de simulação utilizado, onde, falhas únicas são injetadas em apenas um dos processadores, a simulação da arquitetura utilizando TMR não é necessária. Ao se injetar falhas únicas em apenas uma das três copias redundates do processador LEON3 na arquitetura com TMR, o compotamente seria o mesmo apresentado pelo processador sob efeito de falhas na arquitetura com monitor de barramentos. Assim como nos resultados apresentados anteriormente, o

votador da TMR seria capaz de identificar as diferenças entre os resultados gravados no barramento e mascarar os resultados errôneos.

A grande diferença entre as arquiteturas pode ser observada quando a injeção de falhas ocorre nos votadores de ambas as arquiteturas. Na utilização do votador de minorias, nenhuma falha única pode afetar o comportamento do votador, mostrando-se uma solução robusta para eliminação do ponto único de falhas gerado. Porém, ao utilizar o votador da TMR, existe um ponto ao qual uma falha pode gerar um alarme falso no sistema. O sinal "*vote_error*", responsável por indicar que os três processadores redundantes apresentaram resultados diferentes e que não é possível indicar qual o resultado correto, não apresenta nenhuma redundância. Assim, caso uma falha altere seu valor lógico, um alarme falso seria indicado, sendo necessário que os resultados dos três processadores sejam descartados.

9. CONCLUSÃO

O Brasil é um dos poucos países do mundo que possui um programa espacial atuante. As iniciativas para criação do programa espacial brasileiro dotam de 1961, quando uma comissão foi nomeada para estudar a implementação de um programa de pesquisas espaciais, resultando na criação da Comissão Nacional de Atividades Espaciais (CNAE) (Gouveia, 2003). Atualmente existem várias entidades que se dedicam ao desenvolvimento espacial no âmbito nacional, entre as principais pode-se citar: Instituto Nacional de Pesquisas Espaciais (INPE), Centro Tecnológico da Aeronáutica (CTA), Instituto de Aeronáutica e Espaço (IAE) e a Agência Espacial Brasileira (AEB), que é a coordenadora das atividades.

Para que o Brasil continue se destacando como um importante competidor mundial no desenvolvimento de tecnologias para construção de satélites e veículos lançadores, é importante que o meio acadêmico mantenha os trabalhos de pesquisas na área espacial. A competência em áreas como tolerância a falhas causadas pela radiação é de suma importância para utilização em componentes críticos como o OBC.

No presente trabalho foi desenvolvida uma arquitetura de computador de bordo para utilização em aplicações espaciais, utilizando uma técnica não intrusiva, e em tempo de execução, para identificação de falhas simples em soft processors embarcados em FPGAs. O modelo de OBC utilizado baseou-se em especificações do INPE para o OBC da Plataforma Multimissão.

Para que as funções básicas pertinentes ao OBC fossem executadas, foram desenvolvidos componentes de *software* embarcado no processador. Devido aos requisitos temporais necessários nesse tipo de aplicação, todos os componentes de *software* desenvolvidos utilizam um Sistema Operacional de tempo real como base para execução. Os componentes de *software*, juntamente com o Sistema Operacional de tempo real, foram validados realizando o fluxo completo de comunicação e utilizando um modulo de telecomando e telemetria real.

Componentes de lógica programável foram desenvolvidos para permitir a comunicação do processador LEON3 com os demais subsistemas e para aplicação da técnica de identificação de falhas simples no FPGA. Para garantir que as funcionalidades do processador LEON3 não fossem alteradas com a modificação do seu fonte VHDL e com a adição dos blocos para monitoramento do barramento, foram executadas simulações funcionais e testes utilizando o *software* desenvolvido.

Scripts de simulação para verificação funcional da técnica de tolerância a falhas proposta foram desenvolvidos. Os resultados obtidos com a simulação funcional mostram a capacidade da técnica utilizada em identificar falhas no processador do OBC, não permitindo que as mesmas sejam propagadas para os demais módulos do sistema. É importante ressaltar que a simulação funcional utilizou o modelo de falhas simples, injetadas em um dos módulos LEON3, e verificando a capacidade de identificação da falha injetada.

Para que a técnica seja totalmente validada é necessário que trabalhos de emulação de falhas em FPGAs sejam executados com sucesso. Em (Ferlini, 2012), foram utilizadas técnicas de emulação de falhas, em tempo de execução, tendo a arquitetura apresentada no presente trabalho como caso de estudo. Os resultados obtidos em (Ferlini, 2012) mostram uma maior ocorrência de falhas propagadas para uma das saídas do processador LEON3. A técnica de emulação utilizada não permite o mesmo controle, sobre as falhas injetadas, obtido com a simulação. Assim, é necessário que a utilização de emulação seja analizada para que a validação ocorra utilizando o mesmo modelo de falhas. Posteriormente, visando à validação completa, o OBC deve ser exposto a uma câmara capaz de reproduzir os efeitos de radiação aos quais o OBC estaria exposto no espaço.

REFERÊNCIAS

Aeroflex Gaisler. (2001). GRLIB IP Library User's Manual.

Almeida, G. M. (2007). Códigos Corretores de Erros em Hardware para Sistemas de Telecomando e Telemetria em Aplicações Espaciais. *Dissertação de Mestrado - PUCRS* .

Almeida, G., Bezerra, E., Cargnini, L. V., Fagundes, R. D., & Mesquita, D. G. (2007). A Reed-Solomon algorithm for FPGA area optimization in space applications. *Second IEEE NASA/ESA Conference on Adaptive Hardware and Systems* .

Aranci, G., Maltecca, L., & Ranieri., R. (1997). The On Board Data Handling Subsystem for XMM/Integral Space Missions. *DASIA 97 Conference on Data System in Aerospace* .

Atmel Corporation. (2003). *TSC695F SPARC 32-bit Space Processor.* Acesso em 16 de November de 2011, disponível em Atmel Corporation: http://www.atmel.com/dyn/resources/prod_documents/doc4148.pdf

Avizienis, A., Laprie, J.-C., & Randell, B. (2000). Fundamental Concepts of Dependability. *ISW-2000* .

Azevedo, L. R. (2010). Aspectos de Confiabilidade na Implementação da Unidade de Telecomando e Telemetria para Plataformas Orbitais. *Dissertação de Mestrado - PUCRS* .

B. zalke, J., & Pandey, S. K. (2009). Dynamic Partial reconfigurable embedded system to achieve Hardware flexibility using 8051 based RTOS on Xilinx FPGA. *IEEE Proceding* .

Battezzati, N., Sterpone, L., & Violante, M. (2011). Reconfigurable Field Programmable Gate Array for Mission-Critical Applications. *New York, NY: Springer, 2011* .

Battezzati, N., Sterpone, L., & Violante, M. (2011). *Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications.* ISBN 978-1-4419-7595-9.

Benfica, J., Poehls, L. B., Vargas, F., Lipovetzky, J., Lutenberg, A., Gatti, E., et al. (2012). Customized Platform for TID and EMI IC Combined Measurements: Case-Study and Experimental Results. *Journal of Electronic Testing, v. 28, p. 803-816, 2012* .

Bernardi, P., Bolzani, L., Rebaudengo, M., Reorda, M., Vargas, F., & Violante, M. (2006). A New Hybrid Fault Detection Technique for Systems-on-a-Chip. *IEEE TRANSACTIONS ON COMPUTERS , 55*.

CCSDS. (2003b). *Communications Operation Procedure-1. Blue Book. Issue 1.* Washington: CCSDS Recommended Standard.

CCSDS. (2003c). Space Packet Protocol. Blue Book. Issue 1. *CCSDS Recommended Standard* .

CCSDS. (2003a). *TC Space Data Link Protocol. Blue Book. Issue 1.* Washington: CCSDS Recommended Standard.

CCSDS. (1987). Telemetry Summary of Concept and Rationale. Green Book. Issue 1. *CCSDS Recommended Standard* .

da Silva, A., & Sanchez, S. (2010). LEON3 ViP: A Virtual Platform with Fault Injection Capabilities. *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools* .

ECSS. (2003). Ground systems and operations - Telemetry and telecommand packet utilization. *EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS)* .

Ely, T., Duncan, C., Lightsey, E. G., & Mogensen, A. (2006). Real Time Mars Approach Navigation aided by the Mars Network. *American Institute of Aeronautics and Astronautics* .

Entrena, L., Garcia-Valderas, M., Fernandez-Cardenal, R., Lindoso, A., Portela, M., & Lopez-Ongil, C. (2010). Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection. *IEEE Transactions on Computers, no. 99, pp. 1–1.*

Epiphanio, J. C. (2013). Satélites brasileiros de observação da Terra: balanço e perspectivas. *Anais XVI Simpósio Brasileiro de Sensoriamento Remoto - SBSR* .
ESA. (2012). Acesso em 20 de August de 2013, disponível em European Space Agency: http://www.esa.int/Our_Activities/Operations/ESA_satellites_flying_in_formation

Ferlini, F. (2012). PLAESER - Plataforma De Emulação De Soft Errors Visando A Análise Experimental De Técnicas De Tolerância A Falhas: Uma Prototipação Rápida Utilizando Fpgas. *Dissertação de Mestrado - UFSC* .

Ferlini, F., Silva, F., Bezerra, E., & Lettnin, D. (2012). Non-intrusive fault tolerance in soft processors through circuit duplication. *Latin American Testability Workshop - LATW* .

Fiethe, B., Michalik, H., Dierker, C., Osterloh, B., & Zhou, G. (2007). Reconfigurable System-on-Chip Data Processing Units for Space Imaging Instruments. *2007 Design, Automation & Test in Europe Conference & Exhibition* .

FUNDEP. (2011). Especificação De Requisitos De Hardware Do Computador De Bordo Da Plataforma Orbital. *Edital Concorrencia Nº.01/2011 – FUNDEP* .
Gaisler Research. (2002). Packet Telemetry Encoder. *Convolutional-ESA* .

Gaisler, A. (2013). *Leon3*. Retrieved 11 26, 2013, from Aeroflex Gaisler: http://www.gaisler.com/index.php/products/processors/leon3?task=view&id=13

García, M. P., Lindoso, A., Entrena, L., Garcia-Valderas, M., Ongil, C. L., Pianta, B., et al. (2012). Evaluating the Effectiveness of a Software-Based Technique Under SEEs Using FPGA-Based Fault Injection Approach. *Journal of Electronic Testing (Dordrecht. Online), v. 28, p. 777-789, 2012.*

George, J., Koga, R., Swift, G., Allen, G., Carmichael, C., & Tseng, C. W. (2007). Single Event Upsets in Xilinx Virtex-4 FPGA Devices. *IEEE Radiation Effects Data Workshop, pp. 109-114, Jan 2007* .

Gouveia, A. (2003). Esboço Histórico Da Pesquisa Espacial No Brasil. *INPE* .

INPE. (2008). Amazonia-1 Satellite ACDH Subsystem Specification. *Documento de Especificação - A12700-SPC-01 Rev 01* .

INPE. (2001a). Multi-mission Platform Attitude Control and Data Handling (ACDH) Subsystem Specification. *Documento de Especificação: A822700-SPC-01/06* .

INPE. (2001b). *PRR – Multi-mission platform specification.* Acesso em 2 de Maio de 2008, disponível em Documento de Especificação - A822000: http://www.inpe.br/noticias/arquivos/pdf/especificacao.pdf

K, R., & A., S. J. (1994). Scheduling Algorithms and Operating Systems Support for Real-Time Systems. *IEEE Proceedings* .

Malcolm, N., & Zhao, W. (1994). The Timed-Token Protocol for Real-Time Communications. *IEEE Computer (Volume:27 , Issue: 1)* .

Mentor Graphics Corporation. (2010). *ModelSim User Manual.* Acesso em 13 de November de 2011, disponível em http://www.actel.com/documents/modelsim_ug.pdf

Pessota, F. (1999). Analise de Arquiteturas de Computador de Bordo para Missões Espaciais de Longa Duração. *Dissertação Mestrado - ITA* .

Pinheiro, E. R. (2013). Aplicação do Controlador Misto H2/H∞ no Controle da Atitude de um Microssatélite na Presença de Incertezas Paramétricas. *Dissertação de Mestrado - INPE* .

Piotrowski, A., Makowski, D., Jablonski, G., & Napieralski, A. (2008). The automatic implementation of Software Implemented Hardware Fault Tolerance algorithms as a radiation-induced soft errors mitigation technique. *Nuclear Science Symposium Conference Record, 2008. NSS '08. IEEE* .

Schröder, R., Puls, J., Hajnsek, I., Jochim, F., da Silva, M. M., Paradella, W., et al. (2005). The MAPSAR Mission: Objectives, Design and Status. *Anais XII Simpósio Brasileiro de Sensoriamento Remoto* .

Şerban, G., Anton, C., Tutănescu, I., Ionescu, L., & Mazăre, A. (2010). An implementation of BCH codes in a FPGA. *2010 International Conference on Applied Electronics (AE)* .

Silva, F., Ferreira, C., & Villa, P. (2009). Concepção do On-Board Data Handling com Sensor Inercial para Aplicações Espaciais. *Trabalho de Conclusão de Curso - PUCRS* .

Straka, M., Kastil, J., & Kotasek, Z. (2010). Modern Fault Tolerant Architectures Based on Partial Dynamic Reconfiguration in FPGAs. *2010 IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)* .

Straka, M., Kotasek, Z., & Winter, J. (2008). Digital Systems Architectures Based on On-line Checkers. *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools* .

Straka, M., Tobola, J., & Kotasek, Z. (2007). Checker Design for On-line Testing of Xilinx FPGA Communication Protocols. *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems* .

Surrey Space Centre. (2013). *Surrey Space Centre*. Acesso em 26 de January de 2013, disponível em http://www.ee.surrey.ac.uk/SSC/

Tagawa, G. (2013). Estudo de um Controlador de Atitude em um Simulador de Aviônica Modular Integrada (IMA) Aplicado ao Satélite Amazônia-1. *Dissertação de Mestrado - INPE* .

Thomson, D. (1991). *The 8051 Microcontroller: Architecture, Programming and Applications.* Eagan: West Publishing Company.

Velazco, R., Fouillat, P., & Reis, R. (2007). *Radiation Effects on Embedded Systems.* ISBN 978-1-4020-5646-8.

Vladimirova, T., & Curiel, A. d. (2004). *A System-on-a-Chip for Small Satellite Data Processing and Control ("ChipSat").* Surrey Space Centre.

Ziade, H., Ayoubi, R., & Velazco, R. (2004). A Survey on Fault Injection Techniques. *Int. Arab J. Inf. Technol , 1*.