PONTIFICIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
PROGRAMA DE POS-GRADUAÇÃO DE ENGENHARIA ELÉTRICA

PLATAFORMA PARA DESENVOLVIMENTO DE SoC (System-on-Chip) ROBUSTO À INTERFERÊNCIA ELETROMAGNÉTICA

JULIANO D'ORNELAS BENFICA

PORTO ALEGRE
2007

PONTIFICIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
PROGRAMA DE POS-GRADUAÇÃO DE ENGENHARIA ELÉTRICA

PLATAFORMA PARA DESENVOLVIMENTO DE SoC (System-on-Chip) ROBUSTO À INTERFERÊNCIA ELETROMAGNÉTICA

JULIANO D'ORNELAS BENFICA

Orientador: Prof. Dr. Fabian Luis Vargas

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica, da Faculdade de Engenharia da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

PORTO ALEGRE
2007

PLATAFORMA PARA DESENVOLVIMENTO DE SoC (System-on-Chip) ROBUSTO À INTERFERÊNCIA ELETROMAGNÉTICA

CANDIDATO: JULIANO D'ORNELAS BENFICA

Esta dissertação foi julgada para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

_____

Prof. Dr. Daniel Ferreira Coutinho
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

BANCA EXAMINADORA

_____

Prof. Dr. Fabian Luis Vargas – Presidente

_____

Prof. Dr. Eduardo Augusto Bezerra – PUCRS

_____

Prof. Dr. Rubem Fagundes – PUCRS

AGRADECIMENTOS

Ao professor Fabian Luis Vargas, meu orientador e meu amigo acima de tudo, que sempre acreditou e valorizou muito o meu trabalho, orientando e apoiando sempre da melhor forma possível, sendo sempre exemplo de dedicação e orientação.

A todos os integrantes do Grupo SiSC, que contribuíram para a realização deste trabalho, que nesse caso todos participaram de uma forma ou de outra e que a cada dia essas pessoas superam novas barreiras, o que reforça o que dizem por lá "*de tempos em tempos temos que chocar um ovo e sair um macaco de dentro*" e que sem essas figuras raras que geram um ambiente agradável, profissional e motivador para a realização dos projetos, nada seria possível, mas nada mesmo, pois sozinho ninguém vai a lugar algum, e hoje somos um grupo de bons profissionais e acima de tudo um grupo de amigos.

Ao pessoal do INTI – Buenos Aires (Daniel Lupi, Edmundo Gatti e Luis Garcia) e da ORT/URSEC (Fernando Hernandez), que nos possibilitaram a realização dos testes efetuados neste trabalho, fornecendo os equipamentos necessários e colaborando com suas grandes experiências no assunto.

Ao professor e amigo Daniel F. Coutinho que sempre acreditou na minha dedicação aos projetos estando sempre disposto a me receber, ajudar e aconselhar.

Aos professores Rubem D. R. Fagundes e Eduardo Augusto Bezerra que sempre quando conversávamos, sempre surgiam idéias novas, sempre um novo ponto de vista, sempre para melhor.

A minha namorada Jô pelo carinho, paciência, dedicação e motivação para seguir sempre em frente.

As pessoas que são a razão de eu estar aqui hoje escrevendo e tentando expressar o quanto eu acima de tudo os admiro, que são meus pais, Luiz Fernando Benfica, Vera Lúcia Freitas, Roberto Coradini e Evelin D'Ornelas Coradini, que sempre me incentivaram e me apoiaram nas minhas escolhas ao longo de minha vida sempre me direcionando a crescer de uma forma ou de outra.

Aos meus avós e irmãos, Rafael Coradini (O Cabelinho), Matheus Benfica e Fernanda Benfica pelo apoio e força, em especial ao grande *brother* Cabelinho que sempre estava ali literalmente no quarto ao lado, apoiando, dando força e não deixando nunca a peteca cair.

Ao Programa Nacional de Microeletrônica – CNPq, pela oportunidade e ao apoio financeiro dado ao programa de pesquisa e pós-graduação.

A PUCRS, ao Departamento de Engenharia Elétrica, aos funcionários dos laboratórios e secretarias do PPGEE pela disponibilidade dos recursos necessários para a realização desta dissertação de mestrado pelo profissionalismo e competência.

Meus sinceros agradecimentos a todos os colegas, amigos e familiares, que colaboraram e apoiaram de alguma forma no decorrer do curso....

Juliano D'Ornelas Benfica

RESUMO

O ambiente eletromagnético em que sistemas eletrônicos operam está tornando-se cada vez mais hostil. A sociedade observa com bastante entusiasmo a rápida proliferação de uma quantidade infindável de equipamentos eletrônicos sem fio (*wireless*). Infelizmente, esta tendência tem por conseqüência a poluição de forma dramática do espectro de freqüência, e portanto, aumentando o ruído intrínseco do ambiente onde vivemos.

Por outro lado, é fundamental para a aceitação e a segurança destes equipamentos eletrônicos que estes não falhem devido ao ambiente eletromagnético. Assim, é de suma importância compreender como o ruído eletromagnético (*Electromagnetic Interference*, ou EMI) impacta a confiabilidade de sistemas integrados complexos (*Systems-on-Chip*, ou SoC).

Algumas empresas em escala mundial têm demonstrado muita preocupação com este problema através do desenvolvimento de várias plataformas comerciais para o projeto e o teste de SoCs. Entretanto, estas plataformas não garantem medições adequadas da susceptibilidade dos sistemas eletrônicos à EMI.

Este cenário nos motivou a propor uma plataforma de prototipagem reconfigurável para avaliar e aprimorar projetos de SoCs levando-se em consideração sua imunidade ao ruído eletromagnético. Esta plataforma é baseada em normas internacionais IEC 62.132 para o projeto e o teste de sistemas eletrônicos, ao nível de placa. O objetivo final deste conjunto de normas é ditar regras que viabilizam a medição precisa da imunidade de circuitos integrados à EMI, tanto radiada quanto conduzida.

A plataforma desenvolvida é baseada em duas placas específicas e complementares. A primeira é dedicada para o teste de imunidade ao ruído irradiado em uma *Gigahertz Transverse Electromagnetic Cell* (GTEM Cell) de acordo com a norma IEC 62.132-2 (IEC, 2004). A segunda placa é dedicada ao teste conduzido de ruído de RF e foi implementada de acordo com as normas IEC 62.132-4 e IEC 62.132-2 (IEC, 2004), respectivamente.

Após o desenvolvimento da plataforma em questão, um estudo-de-caso baseado no processador *soft-core* da *Xilinx*, *MicroBlaze*, operando sob o controle do sistema operacional *uCOS-II* foi desenvolvido pelo **Grupo SiSC** e testado na plataforma. Os resultados dos ensaios são bastante motivadores e demonstram a capacidade e a flexibilidade da plataforma ser utilizada como ferramenta para avaliar o comportamento de SoCs em ambiente ruidoso do tipo EMI.

ABSTRACT

The electromagnetic environment in which electronic systems operate is becoming more and more hostile. The society observes with enthusiasm the rapid proliferation of a vast diversity of wireless electronic equipments. Unfortunately, this tendency yields as consequence the pollution of the frequency spectrum, and thus, increasing dramatically the environmental noise where we live.

At the same time, it is fundamental for the acceptance and reliability of these electronic equipments that the applications running on their platforms do not fail due to such noisy environment. Therefore, it is fundamental to understand how the electromagnetic noise impacts the reliability of complex electronic systems (Systems-on-Chip, SoC).

Some companies worldwide have been concerned with this situation. They have proposed several commercial platforms to design and test of SoC. However, these platforms do not allow an adequate measurement of SoC susceptibility to electromagnetic interference (EMI).

Such scenario motivated this work. In the following we propose a new platform, reconfigurable, to evaluate and improve SoC designs having in mind the electromagnetic noise immunity. This platform is based on the international standard IEC 62.132 to design and test electronic systems, at the board level. The final goal of this standard is to dictate rules to allow precise measurements of the SoCs susceptibility to (radiated and conducted) EMI.

The proposed platform is based on two specific and complementary boards. The first one is devoted to the radiated noise immunity measurement in a *Gigahertz Transverse Electromagnetic Cell* (GTEM Cell) according to the standard IEC 62.132-2 (IEC, 2004), whereas the second board is dedicated to the RF-conducted noise immunity measurement and was implemented according to the standards IEC 62.132-4 and IEC 62.132-2 (IEC, 2004), respectively.

After the development of the proposed platform, a case-study based on the Xilinx soft-core processor, *MicroBlaze*, was designed by the **Group SiSC** and tested on the platform. The obtained results are rewarding and demonstrate the capability and flexibility of the proposed platform as a tool to evaluate the behavior of SoCs in EMI-exposed environment.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS

ABS - Anti-Blocking System.
ALU - Arithmetic and Logic Unit.
AM - Amplitude Modulation.
ASIC - Application Specific Integrated Circuit.
ATM - Asynchronous Transfer Mode.
BFI - Branch Free Intervals.
BID – Branch Free Interval Identifier.
CCA - Control Flow Checking by Assertions.
CFCSS - Control Flow Checking by Software Signatures.
CFID - Control Flow Identifier.
CLB - Configurable Logic Block.
CMOS - Complementary Metal Oxide Semiconductor.
CPLD – Complex Programmable Logic Device.
CW - Continuous Wave.
DCT - Discrete Cosine Transform.
DDR - Double Data Rate.
DIMM - Dual In-line Memory Module.
DMA - Direct Memory Access.
DSP - Digital Signal Processing.
ECC - Error Correction Code.
ECCA - Enhanced Control Flow using Assertions.
EEPROM - Electrically Erasable Programmable Read Only Memory.
EM – Eletromagnético.
EMC - Electromagnetic Compatibility.
EMI - Electromagnetic Interference.
EPROM - Erasable Programmable Read Only Memory.
ESD - Electrostatic Discharge.
FCC - Federal Communications Commission.
FPGA – Field Programmable Gate Array.
FDDI - Fiber Digital Device Interface.
GTEM – GigaHertz Transverse EletroMagnetic.
HW – Hardware.
IEC - International Electrotechnical Commission.
INTI – Instituto Nacional de Tecnologia Industrial.
I/O – Input / Output.
ISDN - Integrated Services Digital Network.
ISP - Instruction Set Processor.
JTAG - Joint Test Action Group.
LAN - Local Area Network.
LED - Light Emitting Diode.
LUT - Lookup Table.
MEMS - Micro-Electro-Mechanical System.
MMC - Multi Media Card.
MOS - Metal Oxide Semiconductor.

MOSFET - Metal Oxide Silicon Field Effect Transistor.
MP3 - Moving Picture Experts Group Layer-3.
MPEG2 - Motion Picture Experts Group Layer-2.
MPGA - Mask-Programmable Gate Arrays.
MTBF - Mean Time Between Failures.
MTTF - Mean Time To Failure.
MTTR - Mean Time To Repair.
NFET – Negative Field Effect Transistor.
NMOS – Negative Channel Metal-Oxide Semiconductor.
NOC – Network on Chip.
ONO - Oxigênio-Nitrogênio-Oxigênio.
OPB - On-chip Peripheral Bus.
P&D – Projeto e Desenvolvimento.
PAL – Programmable Array Logic.
PBX - Private Branch Exchange.
PCB – Printed Circuit Board.
PFETS – Positive Field Effect Transistor.
POS - Point of Sale.
PROM - Programmable Read Only Memory.
PS/2 - Programming System 2.
RAM - Random Access Memory.
RTOS - Real-Time Operating System.
SATA - Serial Advanced Technology Attachment.
SD - Secure Digital.
SiSC – Sistemas Sinais e Computação – Laboratório da PUCRS.
SoC – System on Chip.
SONET - Synchronous Optical Network.
SDRAM - Synchronous Dynamic Random Access Memory.
SRAM - Static Random Access Memory.
SW – Software.
TMR - Triple Modular Redundancy.
UART - Universal Asynchronous Receiver Transmitter.
USB - Universal Serial Bus.
VGA - Video Graphics Array.
VHDL - Very High Speed Integrated Circuit Hardware Description Language.
VHF - Very High Frequency.
VLSI – Very Large Scale Integration.
VSWR - Voltage Standing Wave Ratio.
YACCA - Yet Another Control-Flow Checking using Assertions.

PARTE I. FUNDAMENTOS

1. INTRODUÇÃO

1.1 Motivação

A área de Projeto e Desenvolvimento (P&D) envolvendo o projeto de sistemas robustos tem crescido consideravelmente nos últimos anos tanto no Brasil quanto no exterior. Porém, tem-se verificado ao mesmo tempo uma lacuna crescente nas pesquisas integrando o co-projeto de Hardware-Software com critérios de confiabilidade (detecção e diagnóstico de falhas) em ambientes de interferência eletromagnética. Neste sentido, especial atenção deve ser dedicada aos sistemas embarcados baseando seu funcionamento e desempenho em tecnologias do tipo *Systems-on-Chip* (SoCs) complexos que necessitam, além de confiabilidade, de alto desempenho, visam atender aplicações em tempo real, de confiabilidade (KATHAIL, 2002), (WOLF, 2004).

O ambiente eletromagnético em que os sistemas eletrônicos têm que operar, está tornando-se cada vez mais hostil. A necessidade da garantia para que não ocorram distúrbios nas aplicações devido ao ambiente eletromagnético, é fundamental para a aceitação e segurança de sistemas eletrônicos para este fim. Podemos citar alguns exemplos de sistemas eletrônicos que necessitam uma garantia de funcionamento de pelo menos 99% em ambientes com ruído eletromagnético:

a) Aplicações aeroespaciais onde a tolerância ao funcionamento de telefones celulares no interior da aeronave, mesmo que recomendado o não uso, que emitem ondas de rádio freqüência que podem afetar os equipamentos de controle de vôo, é obrigatória;
b) Indústria automotiva, que possui equipamentos eletrônicos de grande importância para o funcionamento e segurança do automóvel e de seus condutores tais como controle de freios ABS (*Anti-Blocking System*), injeção eletrônica, controle de tração ativa dentre outros equipamentos que tem de tolerar altos níveis de ruídos;
c) Equipamentos médico-hospitalares, que caracteriza uma aplicação extremamente crítica podendo-se perder vidas caso falhas ocorram no equipamento devido a distúrbios oriundos da rede elétrica e de interferência eletromagnética, por exemplo.

Assim é importante compreender como as tecnologias futuras impactam na nova geração de SoCs. Nota-se que, embora a redução das tensões de alimentação (pelo menos para o *core*) aumente a esperança de que ocorrerá uma menor emissão de interferência eletromagnética tanto conduzida quanto irradiada, este benefício é compensado imediatamente por: (a) o drástico aumento de transistores comutando ao mesmo tempo por CI (circuito integrado), combinado com a alta velocidade destas comutações devido às crescentes freqüências de *clock.* Assim aumentando o ruído total de RF (radiofreqüência) que pode afetar blocos de funções dentro do CI mesmo assim como outros CIs colocados nas proximidades do CI que está emitindo a interferência;

(b) a reduzida tensão de alimentação minimiza as margens de ruído o qual o CI foi projetado para operar. Assim tornando os blocos funcionais internos ao CI mais sensíveis a EMI (interferência eletromagnética).

A utilização de plataformas de teste e desenvolvimento de SoCs comerciais para fins de avaliação a imunidade à EMI, neste caso, não é uma boa opção. Pois após testes preliminares realizados com ensaios de exposição à EMI em uma destas plataformas, chegou-se à conclusão de que para poder avaliar com mais precisão o tipo de falha e onde exatamente ela ocorre, tem de se ter uma placa normalizada. Esta placa deve seguir regras de projeto de alguma norma para podermos avaliar com maior precisão os efeitos da interferência eletromagnética tanto induzida quanto irradiada. Com isso tenta-se separar os efeitos da EMI sobre o CI daqueles sobre os demais componentes da placa, assim como da própria placa (como seus conectores, pinos, *switches*, trilhas etc.).

É baseado neste cenário que nos levou a propor uma plataforma de prototipagem reconfigurável para avaliar e aprimorar projetos de SoCs levando em conta sua imunidade à ruídos eletromagnéticos (EMI) baseados em padrões, procedimentos e regras de projeto de normas internacionais tais como IEC. Assim, o conjunto de normas IEC 62.132 que é dedicada para a medição da imunidade de circuitos integrados à interferência eletromagnética induzida e irradiada, foi escolhida como base para a proposta deste trabalho.

1.2 Objetivos

Este trabalho tem como principal objetivo a implementação de uma plataforma de projeto e teste de SoCs composta por duas placas distintas, sendo uma para ensaios de imunidade a interferência eletromagnética irradiada e outra para conduzida. Tanto o desenvolvimento da plataforma, quanto o procedimento de testes são baseados no conjunto de normas IEC 62.132 (IEC, 2004) dedicado para teste de circuitos integrados.

Este trabalho tem como outros objetivos, além dos citados acima, os seguintes:

- Análise do comportamento de FPGAs do tipo SRAM quando expostos à EMI;
- Estudo e avaliação de técnicas de robustez tanto por software quanto por hardware, tais como técnicas de detecção de falhas de fluxo de controle do processador, técnicas de verificação e reconfiguração do *bitstream* do FPGA e finalmente a avaliação das técnicas e regras de projeto de placas de circuito impresso para exposição à EMI, como recomendado na norma;
- Demonstrar a aplicabilidade da plataforma para validar SoCs em desenvolvimento;
- Verificar a robustez da plataforma em função de ensaios de EMI de acordo com o conjunto de normas IEC 62.132.

O tema desta dissertação é de grande importância para a pesquisa na área de tolerância a falhas e projeto e teste de SoCs robustos a interferência eletromagnética, pois busca validar técnicas de detecção de falhas, produzidas em universidades e empresas, além de observar o comportamento de sistemas não tolerantes em ambientes reais. Através desta dissertação, será possível demonstrar e avaliar uma plataforma para projetos robustos à presença de EMI em ambientes reais.

1.3 Apresentação dos Capítulos

Este trabalho foi divido em três partes, conforme abaixo descrito:

Parte I – Fundamentos (Capítulos 1 ao 6):

- Capítulo 2: apresenta uma breve introdução relacionada as principais tecnologias que envolvem os componentes reconfiguráveis (CPLDs e FPGAs);
- Capítulo 3: apresenta os conceitos clássicos da área e as principais técnicas de detecção de falhas que podem ser implementadas em software e em hardware;
- Capítulo 4: apresenta uma breve introdução relacionada aos principais conceitos que envolvem os SoCs;
- Capítulo 5: apresenta os principais conceitos sobre a compatibilidade eletromagnética juntamente com as principais normas relacionadas a interferência eletromagnética, tais como CE, FCC e IEC 62.132;
- Capítulo 6: aborda as metodologias propostas na literatura, em hardware e em software para testes e injeção de falhas assim como as plataformas de testes e injeções de falhas.

Parte II – Metodologia (Capítulo 7):

- Capítulo 7: apresenta detalhadamente a plataforma de testes e o desenvolvimento dos SoCs propostos neste trabalho;

Parte III – Resultados e Conclusões (Capítulos 8 ao 9):

- Capítulo 8: visando avaliar a metodologia proposta no capítulo anterior, este capítulo apresenta os resultados obtidos a partir dos ensaios de imunidade a

interferência irradiada e conduzida assim como os estudos de caso utilizando a plataforma de testes proposta;

- Capítulo 9: apresenta as conclusões ao desenvolvimento deste trabalho de dissertação e finalmente sugere alguns trabalhos para serem desenvolvidos no futuro.

2. TECNOLOGIA DE COMPONENTES RECONFIGURÁVEIS

2.1 Introdução

Dependendo das aplicações computacionais, é necessária a alteração freqüente de sua funcionalidade ou grande flexibilidade de comportamento. Isto é atualmente possível não apenas via implementação em software, mas também em hardware, graças à existência de dispositivos de hardware reconfigurável.

No caso de implementações em software existe um hardware subjacente normalmente composto por um processador de conjunto de instruções (em inglês, *Instruction Set Processor* ou ISP) associado a uma memória. ISPs podem ser programados para executar uma ou mais aplicações específicas preenchendo a memória de instruções com software que implementa as aplicações.

No caso de implementação em hardware, as aplicações flexíveis são obtidas principalmente através de uso de dispositivos tais como FPGAs (*Field Programmable Gate Array*) e CPLDs (*Complex Programmable Logic Device*). De fato, estes dispositivos modificaram a tradicional distinção entre hardware e software, visto que sua funcionalidade em hardware pode ser alterada de forma total (o hardware é totalmente modificado) ou parcial (apenas uma parte do hardware é modificada) ou até mesmo de forma dinâmica (com o dispositivo em funcionamento, uma parte do hardware é modificada).

2.2 Arquitetura de um CPLD

Os dispositivos de lógica programável complexa são exatamente o que eles reivindicam a ser. São projetados essencialmente para se parecerem como um grande número de PALs (*Programmed Array Logic*) em um único *chip* (que são dispositivos de lógica programável, onde a função lógica é gravada no dispositivo para operar conforme projeto desejado) conectados através de uma chave onde usam as mesmas ferramentas programadoras e de desenvolvimento, baseados nas mesmas tecnologias, mas podem conter uma lógica muito mais complexa em um número muito maior.

O diagrama na Figura 2.1 mostra a arquitetura interna de um CPLD típico. Enquanto cada fabricante tem uma diferente variação, no geral são todos similares e consistem em:

- **Blocos de função**: são os blocos responsáveis pela operação e configuração de cada bloco do CPLD que podem operar como uma unidade lógica e aritmética para fazer operações aritméticas, por

exemplo, ou como um bloco de I/O, que é usado para conduzir sinais aos pinos do dispositivo CPLD nos níveis de tensão apropriados com a corrente apropriada (ZEIDMAN, 2001). Geralmente, os blocos de função são projetados para serem similares às arquiteturas existentes, tais como a PALs, por exemplo, de modo que o projetista possa usar ferramentas familiares ou mesmo projetos mais velhos sem grandes modificações;

- **Matriz de interconexão**: A interconexão de um CPLD é uma grande matriz de chaves programáveis que permite que sinais de todas as partes do dispositivo vão a todas as partes restantes do dispositivo. Quando nenhuma chave puder conectar todos os blocos internos da função a todos os blocos restantes da função, há bastante flexibilidade que permite muitas combinações das conexões (ZEIDMAN, 2001);
- **Elementos programáveis**: os dispositivos são programados usando os elementos programáveis que, dependendo da tecnologia do fabricante, podem ser células EPROM (*Erasable Programmable Read Only Memory*), células EEPROM (*Electrically Erasable Programmable Read Only Memory*), ou células FLASH EPROM (Erasable *Programmable Read Only Memory*) (ZEIDMAN, 2001).

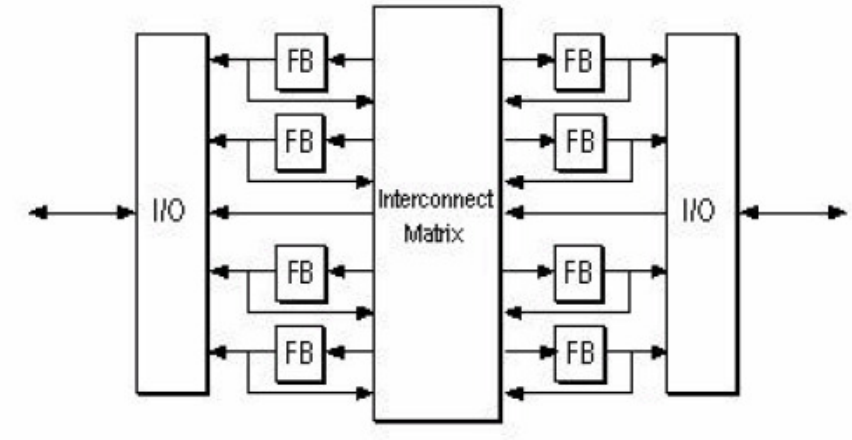


Figura 2.1 – Arquitetura CPLD (ZEIDMAN, 2001).

2.3 Arquitetura de um FPGA

A arquitetura genérica de FPGAs, ilustrada pela Figura 2.2, consiste em uma matriz de elementos agrupados em blocos lógicos configuráveis, que podem ser interconectados, por barramentos de interconexão configuráveis. Igualmente semelhante a uma PAL (*Programmable Array Logic*), as interconexões entre os elementos são implementadas por blocos de chaves configuráveis pelo usuário. Através de blocos de entradas e/ou saídas configuráveis é realizado o interfaceamento com o mundo externo.

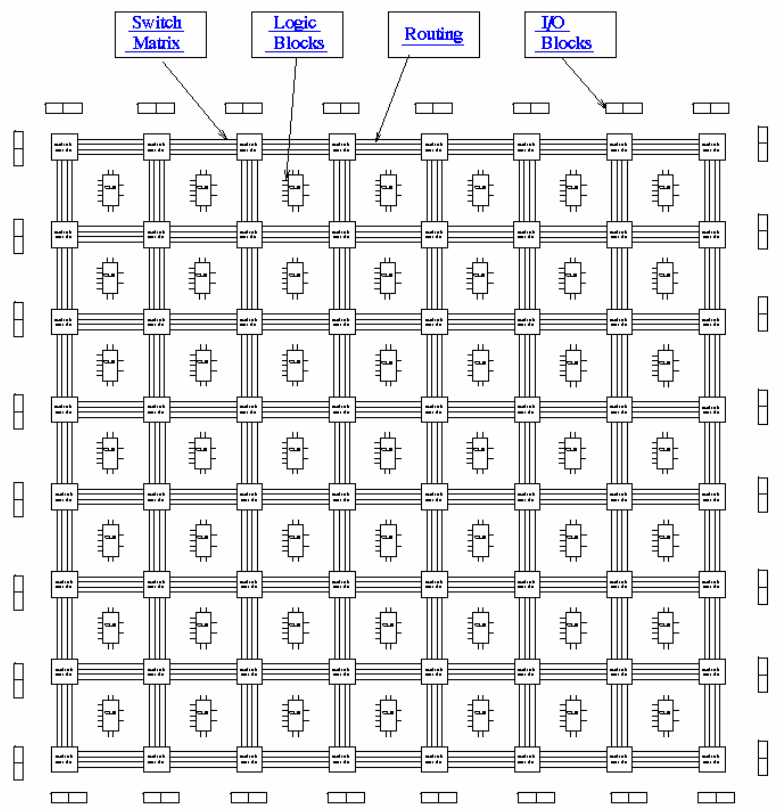


Figura 2.2 - Arquitetura de um FPGA genérico (ZEIDMAN, 2001).

Cada fabricante de FPGA tem sua própria arquitetura, mas em termos gerais são todos uma variação da arquitetura mostrada na Figura 2.2. A arquitetura consiste em:

- **Blocos de configuração lógica**: Os blocos lógicos configuráveis contêm a lógica para o FPGA. O CLB (*Configurable Logic Block*) conterá bastante lógica para criar uma máquina de estados, que contem uma memória RAM para criar as funções de lógicas combinatórias. Contem também *flip-flops* para elementos de armazenamento chaveados por *clock*, e multiplexadores a fim de distribuir a lógica dentro do bloco e a dos recursos externos. Os multiplexadores permitem também a seleção da polaridade do *reset* e a seleção da entrada (TOROK, 2001).
- **Blocos de I/O configurável**: Um bloco de I/O configurável é usado para receber sinais do *chip* e emiti-los para fora outra vez. Ele consiste em um buffer de entrada e um buffer de saída *three-state* (alta impedância) e controles de saída em coletor aberto. Tipicamente há resistores de *pull up* nas saídas e às vezes dependendo da arquitetura há resistores *pull down*. A polaridade da saída pode geralmente ser programada para ser ativo baixo ou ativo alto e frequentemente a taxa de saída pode ser programada para ter tempos de subida (*rise*) e queda

(*fall*) altos e baixos. Além disso, há frequentemente um flip-flop na saída de modo que os sinais controlados pelo *clock* possam ter saída diretamente aos pinos sem encontrar algum atraso (*delay*) significativo. (TOROK, 2001)

- **Interconexão programável**: A interconexão de um FPGA é muito diferente do que aquela em um CPLD, mas é particularmente similar a um vetor de portas ASIC. Na Figura 2.3, uma hierarquia de recursos de interconexão pode ser vista, onde observa-se linhas longas que podem ser usadas para conectar CLBs críticos onde estão fisicamente distantes dentro do chip sem provocar atrados maiores que o desejado. Podem também ser usados como barramento dentro do chip. Há também as linhas curtas que são usadas para conectar CLBs individuas fisicamente perto um dos outros. As chaves programáveis dentro do chip, permitem a conexão de CLBs para interconectarem linhas umas as outras e à matriz de chaves (TOROK, 2001).

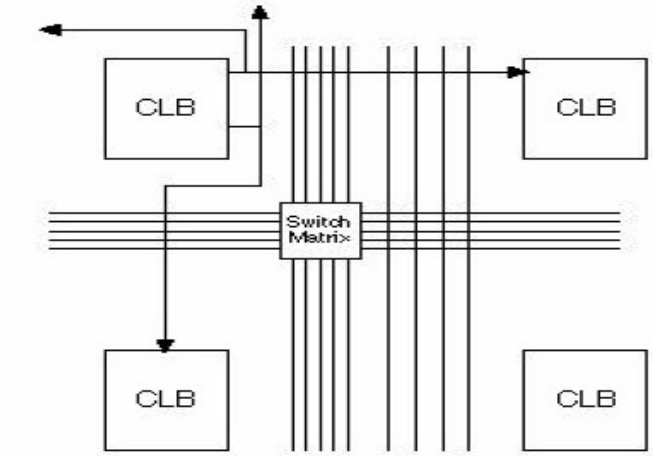


Figura 2.3 – Interconexão programável de um FPGA (TOROK, 2001).

- **Elementos programáveis**: Em um FPGA, a matriz de blocos lógicos é configurada e interconectada eletricamente através de chaves eletrônicas configuráveis. As propriedades destas chaves, tais como tamanho, resistência (em ohms), e capacitância (em farads), delimitam as características elétricas destes circuitos integrados (ROSE, 1993). Existem basicamente três tipos de tecnologias de programação:

  (1) - **Tecnologia SRAM** (*Static Random Access Memory*), usa bits de memória RAM (*Random Access Memory*) estática para programar e controlar as chaves eletrônicas, baseadas em transistores de tecnologia CMOS (*Complementary Metal Oxide Semiconductor*) ou multiplexadores (TOROK, 2001). Considerando que a SRAM é volátil, o FPGA deverá ser configurado a cada vez que for alimentado. Isto exige uma memória externa permanente como PROM (*Programmable Read Only Memory*), EPROM (*Erasable Programmable Read Only*

Memory), EEPROM (*Electrically Erasable Programmable Read Only Memory*) ou outro meio de armazenamento, para prover o vetor de configuração (TOROK, 2001). A tecnologia SRAM é usada em dispositivos da Xilinx (XILINX, 1999), Plessey (PLESSEY, 1989), Algotronix (ALGOTRONIX, 1989), Altera (ALTERA, 2001), Concurrent Logic (CONCURRENT LOGIC, 1991), Toshiba (MUROGA, 1991) e Atmel (ATMEL, 1999);

(2) – Tecnologia antifusível: é um dispositivo que apresenta uma resistência muito alta entre seus terminais. Quando uma tensão de 11 a 20 volts (dependendo do tipo de antifusível) é aplicada entre de seus terminais, o antifusível "queima," criando uma ligação de baixa de resistência (contrário ao fusível convencional que se interrompe), tornando esta ligação permanente (TOROK, 2001). A programação de um antifusível requer um circuito extra para fornecer a tensão de configuração e uma corrente de 5mA ou mais dependendo da família do dispositivo. Dois tipos de antifusíveis são mais utilizados, os fabricados pelo processo Oxigênio-Nitrogênio-Oxigênio (ONO) e o de Silício amorfo. A tecnologia de antifusível é usada nos dispositivos FPGAs da Actel (EL GAMAL, 1989), Quicklogic (BIRKNER, 1991) e Crosspoint (MARPLE, 1992);

(3) - **Tecnologia de porta flutuante**: Esta tecnologia é baseada em configuração por armazenamento de cargas, da mesma forma que utilizada nas memórias EPROM, e Flash RAMs. Cada bit da memória possui um transistor MOS (*Metal Oxide Semiconductor*) com duas portas, uma delas flutuante, não conectada ao barramento da memória (S0) e isolada por material de alta impedância. Esta tecnologia é semelhante a das EPROMs, a não ser pela remoção das cargas da porta flutuante, que pode ser feita eletricamente, no circuito, sem luz ultravioleta. Isto dá uma vantagem a mais, a fácil e mais rápida reconfigurabilidade, que pode ser muito útil em algumas aplicações. Porém, existe uma desvantagem, a célula de EEPROM é aproximadamente duas vezes o tamanho de uma célula de EPROM. A tecnologia de porta flutuante baseada em EEPROM é usada em dispositivos fabricados pela AMD (AMD, 1990), Lattice (BAKER, 1991) e Altera (WONG, 1989).

Também, haverá circuitos de *clock* para chavear pulsos de sinais a cada bloco lógico. Pode haver recursos de lógica adicional tais como ALUs (*Arithmetic and Logic Unit*), memórias e decodificadores. Os dois tipos básicos e mais utilizados de elementos programáveis para um FPGA são SRAM e antifusível, mas explanaremos ainda a tecnologia de configuração de porta flutuante (TOROK, 2001).

Os FPGAs foram introduzidos em 1985 pela empresa Xilinx. Desde então, grande variedade de FPGAs foi desenvolvida por várias outras companhias, entre elas: Actel, Altera, Atmel, Plessey, Plus Logic, Advanced Micro Devices (AMD), Quicklogic, Algotronix, Concurrent Logic, e Crosspoint Solutions (ZEIDMAN, 2001).Os FPGA's são chamados assim porque melhor que ter uma estrutura similar a um PAL ou a outro dispositivo programável, são muito bem estruturados como uma disposição de portas ASIC (*Application Specific Integrated Circuit*). Isto faz dos FPGAs muito bons para o uso em prototipação de ASICs, ou em lugares onde ASICs serão usados eventualmente. Por exemplo, um FPGA pode ser usado em um projeto que necessita entrar no mercado rapidamente não deixando o custo de lado onde mais tarde um ASIC pode ser usado no lugar do FPGA quando o volume da produção aumenta a fim de reduzir o custo (ZEIDMAN, 2001).

3. TÉCNICAS DE DETECÇÃO DE FALHAS

3.1 Introdução

Esse tópico aborda os conceitos clássicos da área de testabilidade de circuitos integrados, para uma maior compreensão desta dissertação. A seguir, apresentam-se alguns conceitos formais retirados da literatura (LAPRIE, 1985), (ANDERSON, 1981), (PRADHAN, 1996), (IYER, 2002) e (BOLZANI, 2004):

Falha: pode ocorrer no hardware ou no software, sendo assim a causa do erro.

Componentes velhos e interferências externas são exemplos de fatores que podem levar à falha.

As falhas podem ser classificadas em transientes, permanentes e intermitentes:

Falhas Permanentes: podem ser geradas durante o processo de fabricação bem como durante a vida útil do circuito. Elas podem ser representadas fisicamente por curtos ou interconexões abertas, que consistem em uma condição falha permanente para o circuito.

Falhas Transientes: aparecem durante a vida útil do circuito e são geradas por fenômenos aleatórios tais como interferência eletromagnética.

Falhas Intermitentes: é caracterizada pela ocorrência temporária e repetida do defeito a partir de alguma variação nas condições externas ao circuito, como por exemplo, ocorrência de vibrações, variações da temperatura etc.

Stuck-at: tipo de falha permanente onde um nó específico do circuito está sempre com o mesmo valor seja ele zero (*stuck-at-zero*) ou um (*stuck-at-one*).

Bit flip: tipo de falha transiente ocasionada por interferências externas, que resultam em uma mudança temporária no valor em um determinado nó do circuito. Esta mudança pode ocorrer nos dois sentidos de zero para um ou de um para zero.

Erro: define-se que um sistema está em estado errôneo, ou em erro, se o processamento posterior a partir desse estado pode levar a um defeito.

Defeito: ocorre quando existe um desvio da especificação. Esse não pode ser tolerado e deve ser evitado.

Latência: período de tempo desde a ocorrência da falha até a manifestação da mesma.

Dependabilidade: esse termo é uma tradução literal do inglês *dependability*. Indica a qualidade e a confiança depositada no serviço fornecido por um dado sistema. Confiabilidade e disponibilidade são dois dos principais atributos da dependabilidade.

Confiabilidade: capacidade de atender à especificação dentro de condições definidas, durante certo período de funcionamento, e estar operacional no início desse período.

Disponibilidade: é a probabilidade de o sistema estar operacional quando a utilização deste seja necessária.

3.2 Falha, Erro ou Defeito

Estamos interessados no sucesso de determinado sistema no atendimento da sua especificação. Um defeito (*failure*) é definido como um desvio da especificação. Define-se que um sistema está em estado errôneo, ou em erro, se o processamento posterior a partir desse estado pode levar a um defeito. Finalmente define-se falha ou falta (*fault*) como a causa física ou algorítmica do erro.

Falhas são inevitáveis. Componentes físicos envelhecem e sofrem com interferências externas, sejam ambientais ou humanas. O software, e também os projetos de software e hardware, são vítimas de sua alta complexidade e da fragilidade humana em trabalhar com grande volume de detalhes ou com deficiências de especificação.

Defeitos são evitáveis usando técnicas de tolerância a falhas.

Na Figura 3.1 é mostrada uma simplificação, sugerida por Laprie (LAPRIE, 1985) e Anderson Lee (ANDERSON, 1981), e também adotada nesse texto, para os conceitos de falha, erro e defeito. Falhas estão associadas ao universo físico, erros ao universo da informação e defeitos ao universo do usuário.

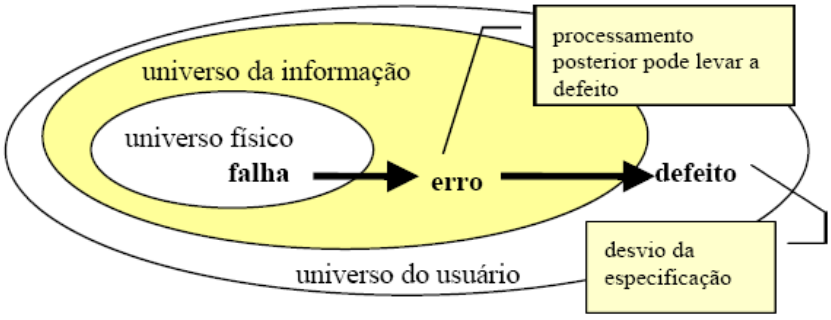


Figura 3.1 Modelo dos três universos, falha, erro e defeito (PRADHAN, 1996).

Por exemplo: um chip de memória, que apresenta uma falha do tipo *stuck-at-zero* em um de seus bits (falha no universo físico), pode provocar uma interpretação errada da informação armazenada em uma estrutura de dados (erro no universo da informação) e como resultado o sistema pode negar autorização de embarque para todos os passageiros de um vôo (defeito no universo do usuário). É interessante observar que uma falha não necessariamente leva a um erro (aquela porção da memória pode nunca ser usada), e um erro não necessariamente conduz a um defeito (no exemplo, a informação de vôo lotado poderia eventualmente ser obtida a partir de outros dados redundantes da estrutura).

3.2.1 Latência

Define-se latência de falha como o período de tempo desde a ocorrência da falha até a manifestação do erro devido àquela falha. Define-se latência de erro como o período de tempo desde a ocorrência do erro até a manifestação do defeito devido aquele erro.

Baseando-se no modelo de 3 universos, o tempo total desde a ocorrência da falha até o aparecimento do defeito é a soma da latência de falhas e da latência de erro.

3.2.2 Classificação de Falhas

Existem várias classificações para falhas na literatura (ANDERSON, 1981), (LAPRIE, 1995), e (PRADHAN, 1996). As falhas aparecem geralmente classificadas em falhas físicas, aquelas de que padecem os componentes, e falhas humanas. Falhas humanas compreendem falhas de projeto e falhas de interação.

As principais causas de falhas são problemas de especificação, problemas de implementação, componentes defeituosos, imperfeições de manufatura, fadiga dos componentes físicos, além de distúrbios externos como radiação, interferência eletromagnética, variações ambientais (temperatura, pressão, umidade) e também problemas de operação.

Além da causa, para definir uma falha considera-se ainda:

- **Natureza:** falha de hardware, falha de software, de projeto, de operação;
- **Duração ou persistência:** permanente ou temporária (intermitente ou transitória);
- **Extensão:** local a um módulo, global;
- **Valor:** determinado ou indeterminado no tempo.

Vem crescendo a ocorrência daquelas falhas provocadas por interação humana maliciosa, ou seja, por aquelas ações que visam propositadamente provocar danos aos sistemas. Essas falhas e suas conseqüências são tratadas por técnicas de segurança computacional (*security*) e não por técnicas de tolerância a falhas. Deve ser considerado, entretanto, que um sistema tolerante a falhas deve ser também seguro as intrusões e ações maliciosas.

Falhas de software e também de projeto são consideradas atualmente o mais grave problema em computação crítica. Sistemas críticos, tradicionalmente, são construídos de forma a suportar falhas físicas. Assim é compreensível que falhas não tratadas, e não previstas, sejam as que mais danos causam, pois possuem um grande potencial de comprometer a confiabilidade e disponibilidade do sistema. Um exame de estatísticas disponíveis (LAPRIE, 1998) confirma essas considerações conforme mostrado na Tabela 3.1.

Sistemas tradicionais				Redes cliente-servidor (não tolerantes a falhas)	
Não tolerantes a falhas		**Tolerantes a falhas**			
MTTF: 6 a 12 semanas Indisponibilidade após defeito: 1 a 4 horas		MTTF: 21 anos (Tandem)		Disponibilidade média: 98%	
Defeitos:		**Defeitos:**		**Defeitos:**	
hardware	50%	software	65%	projeto	60%
software	25%	operações	10%	operações	24%
comunicação/ambiente	15%	hardware	8%	físicos	16%
operações	10%	ambiente	7%		

Tabela 3.1 – Causas usuais em sistemas computacionais (LAPRIE, 1998).

3.3 Defeitos e Modelos de Falhas

O teste de circuitos ou sistemas eletrônicos é realizado com o intuito de detectar falhas eventualmente presentes. Entretanto, para a realização do teste é necessário definir modelos de falhas baseados em falhas reais definidas a partir de mecanismos físicos e leiautes reais. Segundo (BARDELL, 1987) um modelo de falha especifica a série de defeitos físicos que podem ser detectados através de um procedimento de teste. Um bom modelo de falha, segundo (STROUD, 2002), deve ser computacionalmente eficiente em relação ao dispositivo de simulação e refletir fielmente o comportamento dos defeitos que podem ocorrer durante o processo de projeto e manufatura bem como o comportamento das falhas que podem ocorrer durante a operação do sistema. Estes modelos são utilizados na emulação de falhas e defeitos durante a etapa de simulação do projeto.

Assim, nos últimos anos surgiram vários modelos de falhas baseados nos principais defeitos físicos dos circuitos (LAPRIE, 1998).

3.3.1 Modelo de Falha *Gate-Level Stuck-at*

Este modelo define que as portas de entrada e saída podem estar coladas em 0 (*stuck-at*-0) ou coladas em "1" (*stuck-at*-1). Salienta-se que as falhas *stuck-at* são emuladas como se as portas de entradas e saídas estivessem desconectadas e assim amarradas ao valor "0" (*stuck-at*-0) ou ao valor "1" (*stuck-at*-1) (LAPRIE, 1998).

3.3.2 Modelo de Falha Transistor-*Level Stuck*

Este modelo reflete o comportamento exato das falhas de transistores em circuitos NMOS (*Negative Channel Metal-Oxide Semiconductor*) e define que qualquer transistor pode estar *stuck-on* (também denominado *stuck-short*) ou *stuckoff* (também denominado *stuck-open*) (LAPRIE, 1998).

Salienta-se que as falhas *stuck-on* (*s-on*) podem ser emuladas através de um curto circuito entre o *source* e o *drain* do transistor e as falhas *stuck-off* (*s-off*) desconectando-se o transistor do circuito. Alternativamente, falhas stuck-on podem ser emuladas desconectando a porta MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*) do sinal e conectando-a a lógica "1" para transistores NFETS (*Negative Metal Oxide Semiconductor Field Effect Transistor*) ou a lógica "0" em transistores PFETS (*Positive Metal Oxide Semiconductor Field Effect Transistor*).

Este procedimento fará com que o transistor esteja sempre conduzindo. Já no que diz respeito a falhas *stuck-off*, elas podem ser emuladas conectando a porta MOSFET a lógica "0" para transistores NFET e a lógica "1" para transistores PFET, assim o transistor nunca conduzirá.

3.3.3 Modelo de Falha *Bridging*

Este modelo inclui outro importante conjunto de defeitos tais como curtos entre trilhas e rompimento de trilhas (trilhas abertas). Basicamente estes tipos de defeitos resultam de um *overetching* ou *under-etchi*ng durante o processo de fabricação do VLSI (*Very Large Scale Integration*) ou PCB (*printed circuit board*).

Outro modelo de falha *bridging* definido a partir do comportamento observado em curtos que ocorrem em ASIC's (*Application Specific Integrated Circuit*) e FPGAs é definido como *dominant*-AND/OR *bridging*.

Observa-se que apesar das falhas transistor-*level* e *bridging* refletirem mais fielmente o comportamento dos defeitos presentes em circuitos, sua emulação e avaliação em simuladores é computacionalmente mais complexa em relação as tradicionais falhas *stuck-at* (LAPRIE, 1998).

3.3.4 Modelo de Falha *Delay*

Este modelo representa outra importante classe de falhas que ocorrem quando a operação executada pelo circuito é logicamente correta, mas não é executada na freqüência de operação requerida. Este tipo de falha origina-se a partir de um *over* ou *under-etching* durante o processo de fabricação que origina MOSFETs com canais muito mais estreitos ou longos do que os pretendidos.

Assim, o teste de *Delay* concentra-se em encontrar e expor todo e qualquer defeito que possa existir no dispositivo fabricado. O objetivo básico deste tipo de teste é verificar os caminhos entre *flip-flops*, entre entradas primárias e *flip-flops* e finalmente entre *flip-flops* e saídas primárias, ou seja, verificar através da lógica combinacional se durante a operação na velocidade requerida, algum caminho do sistema falhou.

Tipicamente, o teste de *Delay* consiste na aplicação seqüencial de dois vetores tal que o caminho através da lógica combinacional é carregado com o primeiro vetor enquanto o segundo vetor gera a transição através dos caminhos para detecção da falha (LAPRIE, 1998).

3.3.5 Modelo de Falha Simples versus Múltiplos

Durante o processo de fabricação de um determinado dispositivo VLSI ou PCB múltiplos defeitos podem ser inseridos. Para ilustrar com mais clareza, as diferenças em termos de tempo de simulação dos modelos simples versus múltiplos, observe os exemplos abaixo descritos. Em um circuito com N portas de entradas e saídas, diante do modelo múltiplo de falha *stuck-at* deve-se emular 3N-1 e do modelo de falha simples apenas 2N diferentes falhas. A mesma análise pode ser feita diante dos modelos transistor-*level stuck* e para o modelo *wired*-AND/OR ou *dominant bridging*. Já para o modelo *dominant*-AND/OR *bridging* múltiplo é necessário simular 5N-1 falhas e no simples 4N falhas.

Entretanto, a alta cobertura de falhas obtida a partir de modelos de falhas simples garante sua ampla utilização no desenvolvimento e avaliação de testes (LAPRIE, 1998).

3.4 Medidas Relacionadas ao Tempo Médio de Funcionamento

As medidas para avaliação de dependabilidade mais usadas na prática são: taxa de defeitos, MTTF (*mean time to failure*), MTTR (*mean time to repair*), MTBF (*mean time between*). Todas essas medidas estão relacionadas a confiabilidade R(t) (PRADHAN, 1996). A Tabela 3.2 mostra uma definição informal dessas medidas.

Os fabricantes deveriam fornecer medidas de dependabilidade para os seus produtos, tanto para os componentes eletrônicos, como para os sistemas de computação mais complexos. Tais medidas são determinadas pelo fabricante estatisticamente, observando o comportamento dos componentes e dispositivos fabricados.

MEDIDA SIGNIFICADO	TAXA DE DEFEITOS - *FAILURE RATE*
Taxa de defeitos - *failure rate, hazard function, hazard rate*	Número esperado de defeitos em um dado período de tempo; é assumido um valor constante durante o tempo de vida útil do componente.
MTTF	Tempo esperado até a primeira ocorrência de defeito
MTTR	Tempo médio para reparo do sistema
MTBF - *failure*	Tempo médio entre os defeitos do sistema

Tabela 3.2 - Medidas de confiabilidade (PRADHAN, 1996).

A taxa de defeitos de um componente é dada por defeitos por unidade de tempo e varia com o tempo de vida do componente.

Uma representação usual para a taxa de defeitos de componentes de hardware é dada pela curva da banheira. Na Figura 3.2 pode-se distinguir três fases:

- Mortalidade infantil: componentes fracos e mal fabricados;
- Vida útil: taxa de defeitos constantes;

- Envelhecimento: taxa de defeitos crescentes.

Os componentes de hardware só apresentam taxa de defeitos constante durante um período de tempo chamado de vida útil, que segue uma fase com taxa de defeitos decrescente chamada de mortalidade infantil. Para acelerar a fase de mortalidade infantil, os fabricantes recorrem a técnicas de *burn-in*, onde é efetuada a remoção de componentes fracos pela colocação dos componentes em operação acelerada antes de colocá-los no mercado ou no produto final.

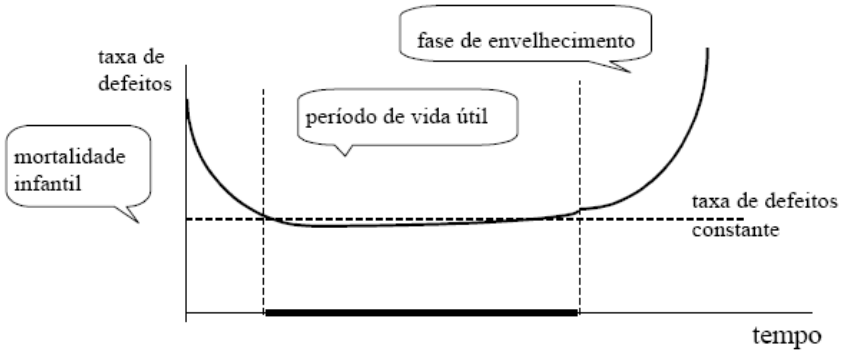


Figura 3.2 – Curva da banheira (PRADHAN, 1996).

É questionável se a curva da banheira pode ser aplicada também para componentes de software. Pode ser observado, no entanto, que os componentes de software também apresentam uma fase de mortalidade infantil ou taxa de erros alta no início da fase de testes, que decresce rapidamente até a entrada em operação do software. A partir desse momento, o software apresenta um taxa de erros constante até que, eventualmente, precise sofrer alguma alteração ou sua plataforma de hardware se torne obsoleta. Nesse momento, a taxa de erros volta a crescer. Intencionalmente mencionamos taxa de erros para software e não defeitos, porque erro é o termo usualmente empregado quando se trata de programas incorretos.

3.5 Detecção de Falhas de Fluxo de Controle

3.5.1 Técnica CFCSS (Control Flow Checking by Software Signatures)

A técnica de CFCSS, proposta por McCluskey et al (MCCLUSKEY, 2002), é uma solução via software que provê a detecção de erros de fluxo de controle. A CFCSS baseia-se fundamentalmente na divisão do programa em blocos básicos, na atribuição de assinaturas para cada um deles e no monitoramento das assinaturas em tempo de execução.

A técnica determina que o programa alvo seja dividido em blocos básicos e que a partir desta divisão seja construído um grafo que representa o fluxo de execução do mesmo. A Figura 3.3 mostra um programa e seu respectivo grafo de execução.

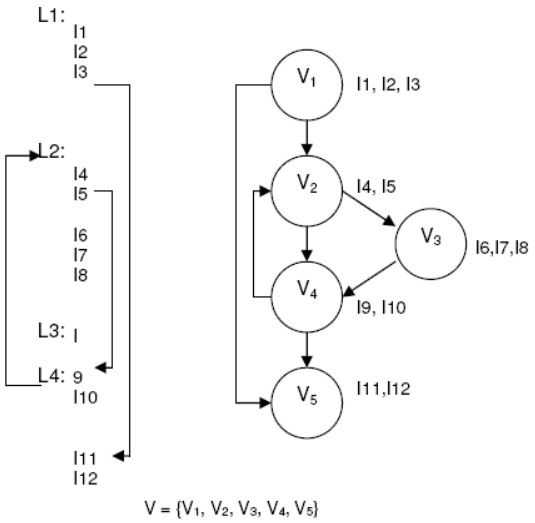


Figura 3.3 – Seqüência de instruções e seu respectivo grafo (LOPES, 2005).

Descrição detalhada da técnica de CFCSS:

A técnica de CFCSS verifica em tempo real o fluxo de controle do programa através do monitoramento e verificação das assinaturas, sendo definida pelos seguintes passos:

1. Dividir o programa em blocos básicos (*vi*);
2. Construir o grafo que representa o fluxo de execução do programa;
3. Atribuir aleatoriamente um valor único de assinatura (*si*) para cada bloco básico em tempo de compilação para representar o bloco básico *vi*; essa assinatura é atribuição de um valor a uma constante no código da aplicação;
4. Calcular em tempo de compilação a assinatura diferença (*di*), definida com base nas assinaturas do(s) predecessor (es) de *vi* (pred(*vi*)) e na assinatura de *vi*. Este cálculo é realizado através da seguinte fórmula: $di = ss \oplus sd$, onde *ss* representa a assinatura do nó fonte (predecessor) e *sd* representam a assinatura do nó destino (no caso o próprio nó *vi*). Assim, dado o bloco básico *vi* e seu conjunto de predecessores igual a pred(*vi*)={*vj*}, a assinatura *di* é calculada a partir da formula: $di = sj \oplus si$;
5. Quando necessário, calcular em tempo de compilação a assinatura de ajuste (*D*). Esta assinatura é utilizada quando um determinado nó *vi* possui mais de um predecessor.

Além dos passos acima descritos, uma assinatura *G* deve ser calculada em tempo de execução e armazenada em uma variável dedicada, denominada registrador de assinatura global (Global Signature Register - GSR). Assim, toda vez que o controle é transferido de um bloco básico para outro, a assinatura *G* é atualizada através da função de assinatura f dada pela equação (3.1).

$f = f(G, di) = G \oplus dd$, *onde* $dd = ss \oplus sd$ (3.1)

A Figura 3.4 ilustra um exemplo de um desvio legal que ocorre do nó *v1* para *v2*. Observe que antes do desvio, $G = G1 = S1$ e após o desvio, *G* é atualizado a partir do seguinte cálculo: $f = f(G1,d2)=G1 \oplus d2$, onde $d2 = s1 \oplus s2$ e assim $G2 = s1 \oplus (s1 \oplus s2)=s2$, ou seja, para o nó *v2*, $G2 = s2$.

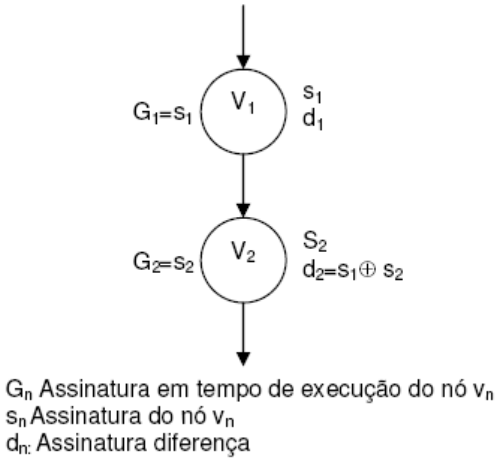


Figura 3.4 - Exemplo de um desvio legal de v1 para v2 (LOPES, 2005).

A Figura 3.5 ilustra um exemplo de desvio ilegal de *v1* para *v4*. Observe que antes do desvio, $G = G1 = S1$ e após o desvio, *G* é atualizado a partir do seguinte cálculo:

$f(G1,d4)=G1 \oplus d4$, onde, $d4=s3 \oplus s4$ e assim $G4 = s1 \oplus (s3 \oplus s4) \neq s4$, ou seja, o erro de fluxo de controle será detectado já que *G4* é diferente de *s4*.

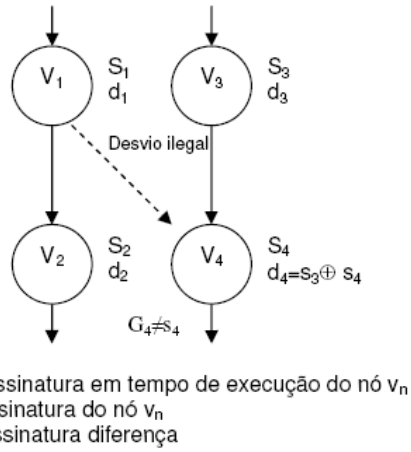


Figura 3.5 - Exemplo de um desvio ilegal de v1 para v4 (LOPES, 2005).

Dessa forma, no topo de cada bloco básico devem ser inseridas as instruções de verificação abaixo definidas:

- Função assinatura: $G = (G \oplus dk)$;
- Instrução de verificação de desvio: "$Br\ (G \neq sk)\ error$".

A Figura 3.6 ilustra, respectivamente, a representação gráfica das instruções, do bloco básico, do bloco básico acrescido das instruções de verificação e a representação utilizada para um nó em um grafo de fluxo de execução.

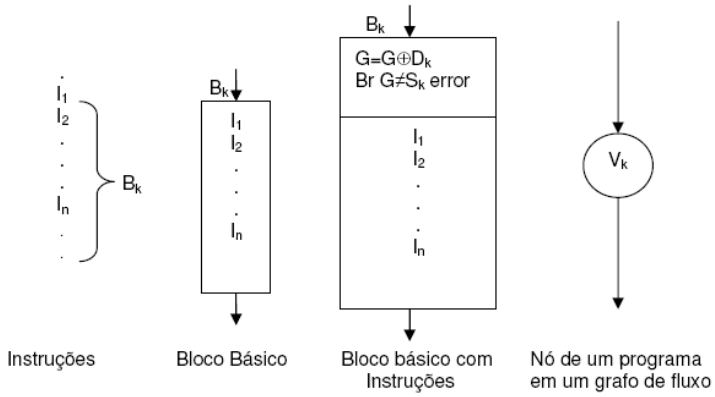


Figura 3.6 - Representação gráfica (LOPES, 2005).

Entretanto, quando um determinado bloco básico possui mais de um predecessor (grafos com nós convergentes), é necessário inserir no código uma assinatura extra denotada como *D*. A Figura 3.7 ilustra claramente o problema que surge quando um determinado bloco básico possui mais de um predecessor. Observe que os nós *v1* e *v3* desviam para o nó *v5*. Logo, *d5* seria um resultado de $s1 \oplus s5$. Caso o desvio seja *br1,5,* o $G5 = G1 \oplus d5 = s1 \oplus s1 \oplus s5 = s5$, ou seja, o *G5* será igual a *s5* e conseqüentemente nenhum erro será observado. Contudo, ocorrerá um erro se o desvio for *br3,5*, pois o $G5 = G3 \oplus d5 => s3 \oplus s1 \oplus s5 \neq s5$ devido a $s3 \neq s1$.

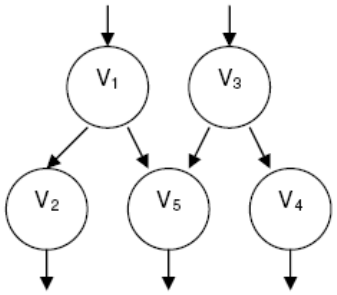


Figura 3.7 - Exemplo de um bloco básico com mais de um predecessor (LOPES, 2005).

Neste contexto, a fim de solucionar o problema acima ilustrado, surge a assinatura *D*, que deve ser inserida após a instrução que realiza o cálculo de atualização de *G*. A Figura 3.8 ilustra um exemplo de utilização da assinatura *D*. Observe que no bloco básico *B5* é adicionada a instrução $G = G \oplus D$ após a instrução que calcula a atualização da assinatura *G*, dada por $G = G \oplus d5$. *D* é determinado a partir dos nós fontes *v1* e *v3*, ou seja, $D = s1 \oplus s3$. Portanto, inicialmente, $d5 = s1 \oplus s5$ e *D* no bloco *B1* apresentam o valor 0000. Assim, após o desvio $br1, 5, 5 = G \oplus d5$ e $G5 = G \oplus D = s5 = 0000 = s5$ e após o desvio $br3,5, G5 = G3 \oplus d5 = s3 \oplus (s1 \oplus s5)$ $G = G5 \oplus D = s3 \oplus (s1 \oplus s5) \oplus (s1 \oplus s3) = s5$.

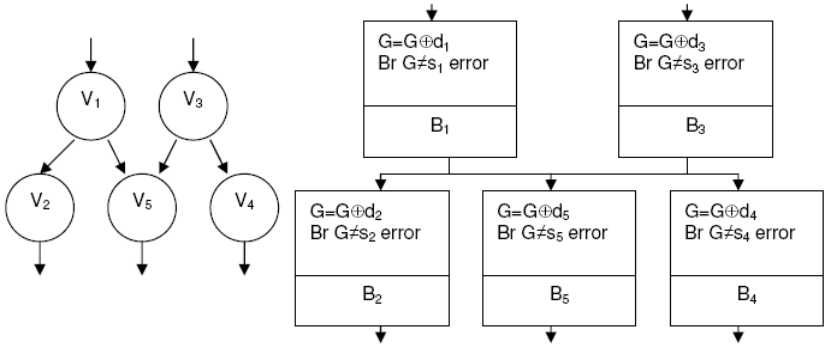


Figura 3.8 - Exemplo de utilização da assinatura D utilizada para solucionar o problema de nós convergentes (LOPES, 2005).

Então, surge o algoritmo abaixo, denominado algoritmo A, para nortear a implementação da técnica de CFCSS em um determinado programa.

Algoritmo A:

Passo 1: Identificar todos os blocos básicos, o grafo de fluxo do programa e o número de nós do grafo;

Passo 2: Atribuir um *si* para cada *v*i, em que $si \neq sj$ *se* $i \neq j$, *i,j=1,2,...,N;*

Passo 3: Para cada *vi,j = 1,2,...,N*;

1.1. Para *vj* cujo pred(vj) é somente um nó *vi*, então $dj = si \oplus sj$;

1.2. Para *vj* cujo pred(vj) é um conjunto de nós vi, 1, vi, 2,...,vi, M, dj é determinado por um dos nós como dj = si,1 ⊕ sj. Para vi,m, m=1, *2,..., M*, inserir uma instrução $Di, m = si,1 \oplus si,m$ em *vi,m*;

1.3. Inserir uma instrução $G = G \oplus dj$ no começo de *vj*;

1.4. Se *vj* é um nó com mais de um predecessor, então inserir a instrução $G = G \oplus D$ após $G = G \oplus dj$ no nó *vj*;

1.5. Inserir uma instrução "$br(G \neq sj)$ error" após as instruções dos dois últimos passos;

Passo 4: Fim Algoritmo.

3.5.2 Técnica CCA (Control Flow Checking by Assertions)

A técnica CCA proposta por G. A. Kanawati et al (KANAWATI, 1996) é uma solução que provê a detecção de erros de fluxo de controle do software. Esta consiste na divisão do código do programa em intervalos livres de desvio (*Branch Free Intervals* - BFIs) na inserção de dois identificadores para cada um dos BFIs, e na verificação dos identificadores durante o período de execução do código.

O primeiro identificador é denominado identificador do intervalo livre de desvio (*branchfree interval Identifier* - BID) e armazena um valor único que identifica o BFI. Já o segundo, denominado identificador de fluxo de controle (*Control Flow*

Identifier - CFID), representa os desvios permitidos, ou seja, indica se os desvios de fluxo de controle estão ocorrendo na seqüência correta. O CFID é armazenado em duas filas de elementos que são inicializadas com o CFID do primeiro BFI. Na entrada do BFI, o CFID do próximo BFI é colocado na fila e na saída do mesmo, o CFID é retirado da fila. A fila que monitora o valor CFID provê a redução da média de latência de detecção da falha.

A Figura 3.9 mostra a estrutura IF-THEN-ELSE acrescida das instruções de verificação da técnica CCA.

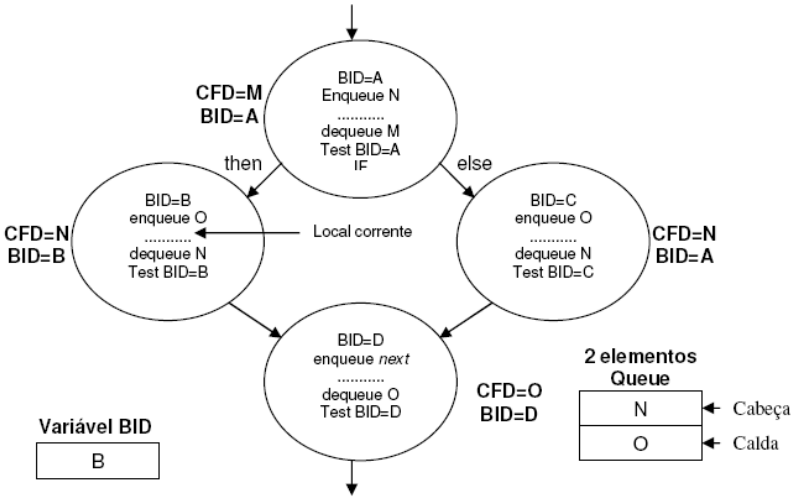


Figura 3.9 - Instruções e verificação dos IDs para estrutura IF-THEN-ELSE na CCA (ALKHALIFA, 1999).

3.5.3 Técnica ECCA (Enhanced Control Flow using Assertions)

Segundo Z. Alkhalifa (ALKHALIFA, 1997), a técnica ECCA é um melhoramento da CCA, que é uma solução em software para detecção de erros no fluxo de controle. Esta técnica insere instruções de teste e atualização no código da aplicação, para torná-lo tolerante a falhas e, conseqüentemente, mais confiável e robusto. São três os passos para a aplicação da técnica ECCA:

Passo 1: dividir o programa em um conjunto de intervalos livres de desvio (BFIs);

Passo 2: atribuir um número primo único, denominado identificador de intervalos livres de desvio (BID), para cada BFI;

Passo 3: inserir no início do BFI a instrução (3.2) e no fim a instrução (3.3).

$$id \leftarrow \frac{BID}{\overline{(id \bmod BID)}.(id \bmod 2)} \qquad (3.2)$$

$$id \leftarrow NEXT + \overline{\overline{(id - BID)}} \qquad (3.3)$$

Onde: id é a variável atualizada durante o tempo de execução do algoritmo na entrada e na saída do BFI, ou seja, ela monitora o fluxo de execução do algoritmo; BID representa a assinatura de cada BFI, ou seja, BID armazena um número primo único para cada BFI gerado em tempo de pré-processamento; NEXT representa o somatório de todos os BID dos BFI que podem ser sucessores do BFI atual gerado em tempo de pré-processamento.

Assim, um programa escrito em linguagem C possui a seguinte estrutura após ser pré-processado pela técnica ECCA:

```
/* Início do BFI */
id = <BID> / (((!(id%<BID>)) * (id%2));
… corpo do BFI …
id = <NEXT> + !!(id-<BID>);
/* Fim do BFI */
```

A Figura 3.10 apresenta a estrutura de um programa em C dividido em BFIs. A Figura 3.11 representa o mesmo programa acrescido das instruções de controle definidas pela técnica, ou seja, após o pré-processamento. Finalmente, a Figura 3.12 apresenta o diagrama de bloco do mesmo programa.

```
/* Inicio do código original*/
... BFI1 ...
if foo
{
... BFI2 ...
}
else
{
... BFI3 ...
}
... BFI4 ...
/* Fim do código orignal */
```

Figura 3.10 - Representação do código original (ALKHALIFA, 1997).

```
/* Início do código pré-processado*/
...
... BFI1...
id = 35+!!(id-3);                              /* <BID> é 3 */
/* Observe que NEXT vale 35 e resulta da multiplicação do BID =
5 do BFI 2 com o BID = 7 do BFI 3. */
if foo
{           id = 5 / (((!(id%5)*(id%2));       /* <BID> é 5 */
            ... BFI2 ...
            id = 11+!!(id-5);}
else
            id = 7/(((!(id%7))*(id%2));       /* <BID> é 7 */
            … BFI3 …
             id = 11+!!(id-7);}
id = 11/(((!(id%11))*(id%2));                  /* <BID> é 11 */
... BFI4 ...
/* Fim do código pré-processado */
```

Figura 3.11 - Representação do código tolerante a falhas de acordo com a ECCA (ALKHALIFA, 1997).

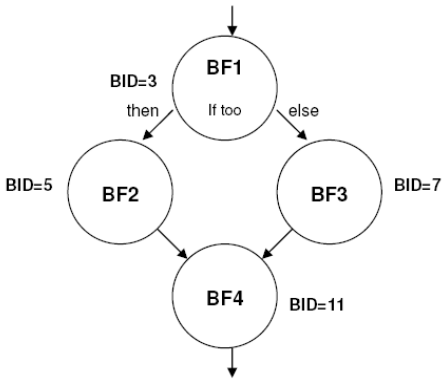


Figura 3.12 - Diagrama de bloco do código tolerante a falhas de acordo com a ECCA (ALKHALIFA, 1997).

A CCA e a ECCA são técnicas portáveis: podem ser implementadas para múltiplas plataformas, e são soluções implementadas puramente em software. Quanto à cobertura de falhas, ambas apresentam a mesma capacidade de detecção. Entretanto, a ECCA apresenta algumas vantagens em relação à CCA:

- Menor overhead, pois são inseridas apenas duas linhas de código por BFI;
- ECCA utiliza apenas uma variável ao invés de duas filas e uma variável.

3.5.4 Técnica YACCA (Yet Another Control-Flow Checking using Assertions)

A técnica YACCA, proposta por O. Goloubeva et al (GOLOUBEVA, 2003), consiste em uma solução via software capaz de detectar erros de fluxo de controle através do monitoramento de assinaturas inseridas nos programas. A técnica define um conjunto de assinaturas que são geradas em tempo de compilação e uma assinatura atualizada em tempo de execução. O conjunto de assinaturas geradas em tempo de compilação são os identificadores dos blocos básicos, e a assinatura gerada em tempo de execução é armazenada em uma variável dedicada, e está associada ao valor do bloco básico atual.

Os passos para aplicar a YACCA podem ser vistos a seguir:

Passo 1: dividir o programa em blocos básicos e definir seu grafo de fluxo de execução;

Passo 2: associar, em tempo de compilação, a cada bloco básico uma assinatura única;

Passo 3: inserir em cada bloco básico as seguintes instruções: uma instrução de teste, que controla a assinatura do bloco básico anterior (predecessor) e verifica se o desvio ocorrido é válido de acordo com o grafo do programa. Uma instrução que calcula

a variável CODE (assinatura) com o novo valor referente ao bloco atual. Essas instruções pertencem respectivamente ao conjunto de teste e ao conjunto das assinaturas.

Regras para definir o conjunto de teste: durante a execução do programa, quando ocorre uma transição do bloco básico *vj* para *vi*, é necessário verificar se esta transição é legal, ou seja, se *brj,i* pertence ao conjunto *Ei*.

Para que este controle seja realizado, a técnica sugere a criação de uma variável denominada PREVIOUS que contém o produto de todas as assinaturas dos nós predecessores de *vi*, equação (3.4), e a inserção da instrução de teste no início de cada bloco básico, equação (3.5).

$$PREVIOUS = \Pi B_j, \ \forall v_j \ com \ desvio \ br_{j,i} \in E_i \qquad (3.4)$$

Onde: *Bj* corresponde a assinatura(s) do(s) nó(s) predecessor(es) de *vi*; *Ei* é o conjunto que contém os desvios legais para *vi*; *brj,i* representa o desvio de *vj* para *vi*.

if (PREVIOUS % CODE) error() (3.5)

Devido à complexidade da instrução da equação (3.5), seu processamento torna-se bastante crítico, por isto, são sugeridas duas soluções alternativas representadas pelas equações (3.6) e (3.7).

if ((CODE!= Ba) && (CODE!= Bb) && (...) && (CODE!= Bn)) error() = (3.6)

Onde: *Ei = {bra,i; brb,i; ...; brn,i}*

ERR_CODE |= ((CODE!= Ba) && (CODE!= Bb) && (...) && (CODE!= Bn)) (3.7)

Onde*: Ei = {bra,i; brb,i; ...; brn,i}*

Regras para definir o conjunto de assinaturas: Dado um determinado bloco básico *vi*, a variável CODE será igual a *Bi*, onde *Bi* corresponde à assinatura de *vi*. A fórmula genérica para o cálculo de CODE é dada pela equação (3.8).

CODE = (CODE & M1) ⊕ M2 (3.8)

Onde, *M1* representa uma constante calculada a partir das assinaturas dos nós que formam o conjunto dos predecessores (*vi*). Já *M2* representa uma constante gerada a partir da assinatura do nó atual e dos nós que formam o conjunto dos predecessores (*vi*).

Os exemplos abaixo demonstram claramente o cálculo da variável code.

Exemplo 1: Dado *vi*, seu conjunto de predecessores é:

pred(vi) = {vj}

M1 = -1 e *M2 = Bj ⊕ Bi*

Assim, *code = code ⊕ (Bj ⊕ Bi)*

Exemplo 2: Dado *vi*, seu conjunto de predecessores é:
pred(vi) = {vj, vk}
M1 = (Bj ⊕ Bk) e M2 = (Bj &(Bj ⊕ Bk) ⊕ Bi)
E assim, *code = (code&(Bj ⊕ Bk)) ⊕ (Bj &(Bj ⊕ Bk) ⊕ Bi))*

4. SYSTEM ON CHIP (SoC)

4.1 Introdução

O aumento do número de transistores e da freqüência de operação, o curto tempo de projeto e a redução do ciclo de vida dos produtos eletrônicos caracterizam o cenário atual da indústria de semicondutores. Em 1980, a maioria dos circuitos integrados (CIs) ditos complexos era composta por dezenas de milhares de transistores. Atualmente, é possível encontrar CIs que contenham dezenas de milhões de transistores (REIS, 2000). Contudo, estima-se que até 2012 existam CIs contendo 4 bilhões de transistores, operando a uma freqüência de 10 GHz (SEMATECH, 2002). Esse avanço tecnológico permite implementar sistemas computacionais completos em um único CI, denominados SoCs (do inglês, *Systems on Chip*) (BERGAMASCHI, 2000) (BERGAMASCHI, 2001).

SoCs podem ser compostos por processadores, memórias, módulos dedicados em hardware para realização de funções específicas, módulos de software e inclusive tecnologias mais recentes como MEMs (do inglês, *Micro-Electro-Mechanical Systems*) integrados no mesmo CI (MARTIN, 2001) (JUNEIDI, 2001) (SEMATECH, 2002). A heterogeneidade apresentada por esses sistemas, ou seja, a possibilidade de combinar diferentes tecnologias no mesmo CI tem sido aproveitada em diversos setores da indústria. Tal fato é evidenciado pelo sucesso de produtos eletrônicos como telefones celulares, PDAs, aparelhos de jogos eletrônicos, aparelhos de DVDs, máquinas fotográficas digitais (integração de sensores ópticos com lógica digital e analógica), entre outros produtos (BENINI, 2001) (RUNNER, 2000) (SIA, 1999). Porém, combinar diferentes tecnologias no mesmo CI é um processo difícil, que implica o aumento de tempo e custo de projeto de novos produtos. Por outro lado, existe a necessidade da diminuição no tempo de lançamento do produto no mercado (*time-to-market*), o que vêm a contribuir para o domínio de mercado e aumento dos lucros.

Ambos os fatores, evolução tecnológica e o curto *time-to-market* possuem grande efeito na diminuição do ciclo de vida dos produtos. Segundo Bergamaschi (BERGAMASCHI, 2002), a diminuição do ciclo de vida dos produtos pode ser evidenciada pelo tempo que um dado produto leva para ter o seu volume de vendas igual a 1 milhão de unidades, como ilustrado na Figura 4.1, que ilustra a diminuição do ciclo de vida de produtos eletrônicos frente ao volume de vendas após a introdução dos mesmos ao mercado. A mesma figura mostra que um televisor a cores levou aproximadamente 10 anos para vender 1 milhão de unidades, enquanto, que em um ano vendeu-se o mesmo volume de aparelhos de DVD.

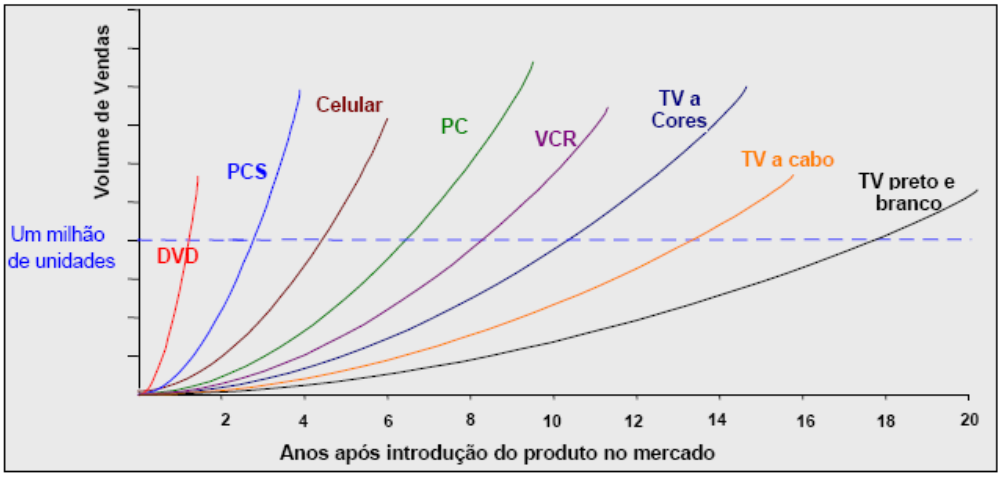


Figura 4.1 - Diminuição do ciclo de vida dos produtos (BERGAMASCHI, 2002).

Sendo assim, é possível constatar que existe uma grande pressão em relação à diminuição do tempo de desenvolvimento e de lançamento de um produto ao mercado antes que o mesmo já esteja ultrapassado tecnologicamente. Dentro deste cenário, as companhias que projetam e vendem SoCs priorizam em seus projetos a utilização de núcleos de propriedade intelectual, ou simplesmente núcleos (do inglês, *IP cores*), a fim de aumentar a produtividade, minimizando o tempo de desenvolvimento, e conseqüentemente, o *time-to-market* de seus produtos (BERGAMASCHI, 2000).

4.2 Linguagens para Descrição de SoCs

O projeto concorrente de hardware e software é uma importante característica dos SoCs. O procedimento usual para o projeto destes sistemas é utilizar uma dada linguagem para descrever os módulos de hardware e outra linguagem para descrever os procedimentos de software. Linguagens de descrição de hardware, tais como Verilog (THOMAS, 1991) e VHDL (PERRY, 1998), têm como características comuns hierarquia (descrição estrutural com utilização de componentes) paralelismo e temporização. Linguagens para descrição de software, como C ou C++, são baseadas em um modelo de execução seqüencial, adaptadas para a execução em processadores de propósito geral. Estas linguagens para descrição de software geralmente não têm suporte para modelar paralelismo e temporização.

Exemplos de linguagens utilizadas para modelagem concorrente de hardware e software são *SystemC* (SYSTEMC, 1993), *SystemVerilog* (THOMAS, 1991) e *SpecC* (GAJSKI, 2000). Estas linguagens são derivadas de linguagens de descrição de software, acrescentando-se bibliotecas que simulam as características de paralelismo e temporização. A descrição de sistemas computacionais através destas linguagens permite também aumentar o nível de abstração de projeto. A elevação dos níveis de abstração de projeto para os chamados níveis sistêmicos permite que detalhes de baixo nível sejam abstraídos, tornando mais fácil descrever a funcionalidade de cada núcleo e suas interconexões.

4.3 Arquitetura Genérica de um SoC

Como descrito anteriormente, o mercado de semicondutores é caracterizado por produtos eletrônicos cada vez mais complexos, com reduzido tempo de vida. Com isso, tornou-se comum desenvolver SoCs a partir de núcleos heterogêneos. A Associação das Indústrias de Semicondutores (SIA, 1999) estima que em 2005 90% da área dos circuitos integrados sejam preenchidas por núcleos. Logo, é possível inferir que a produção de SoCs depende da criação e da validação de núcleos que possam ser reutilizados em projetos distintos. Estima-se que criar um núcleo que possa ser reutilizado por outros projetistas é substancialmente mais difícil (por um fator estimado entre 2 e 5 vezes maior) que desenvolvê-lo para um único projeto (SEMATECH, 2002).

Como ilustrado na Figura 4.2 (MADISETTI, 1997), um SoC é composto por núcleos não programáveis, processadores e memórias que comunicam-se através de uma estrutura de interconexão e interfaces com o mundo externo.

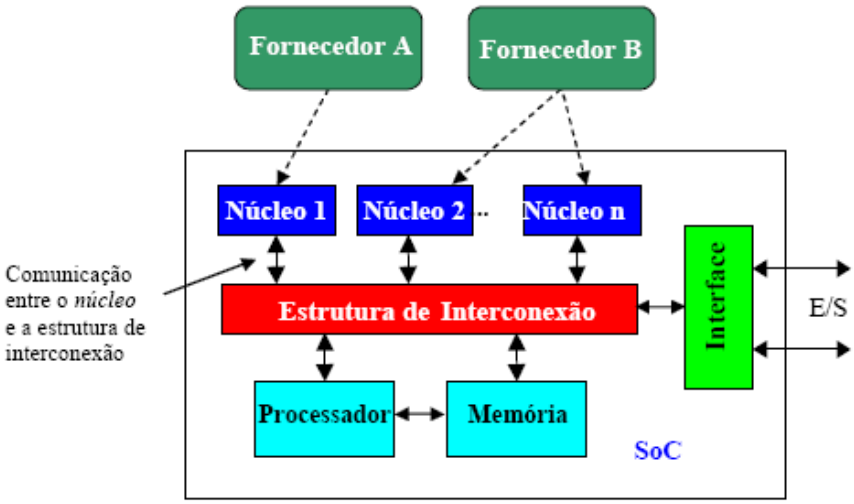


Figura 4.2 - Arquitetura genérica de um SoC (OST, 2004).

A arquitetura acima é composta por núcleos IP. Estes núcleos podem ser módulos de telecomunicação, processadores DSP (*Digital Signal Process*), decodificadores de MPEG2 (*Motion Picture Experts Group Layer-2*) ou MP3 (*Motion Picture Experts Group Layer-3*), entre outros. Os núcleos provêm de fornecedores distintos (A, B) e estão integrados a uma estrutura de interconexão que pode ser um barramento ou uma rede intra-chip. A(s) interface(s) com o mundo externo (I/O) é(são) utilizada(s) para interconectar periféricos, como: porta USB (*Universal Serial Bus*) ou uma UART (*Universal Assynchronous Receiver Transmiter*).

Gupta et al. (GUPTA, 1997) definem um núcleo como um módulo de hardware pré-projetado e pré-verificado, que pode ser usado na construção de uma aplicação maior ou mais complexa em um CI. Estes núcleos podem ser classificados em três categorias: (i) *soft core*, (ii) *firm core* e (iii) *hard core*.

i. ***Soft Core*** é a descrição de um núcleo em uma linguagem de descrição de hardware (e.g. VHDL, Verilog, SystemC). As principais vantagens apresentadas por um *soft core* são: a independência de tecnologia e a flexibilidade. É possível que o projetista/usuário modifique o mesmo, visando adequar a funcionalidade deste em prol do sistema desejado. Em contrapartida a esta flexibilidade, é responsabilidade do projetista atender às restrições temporais. Normalmente, os *soft cores* são acompanhados de scripts de síntese, que guiam o projetista na etapa de concepção.

ii. ***Firm Core*** é um *netlist* que apresenta menos flexibilidade em relação ao *soft core* e na maioria dos casos dependem de uma tecnologia específica (e.g. *netlist* EDIF obtido a partir de síntese lógica). As vantagens em relação aos *soft cores* são: melhor proteção da propriedade intelectual e estimativa de desempenho mais próxima da realidade.

iii. ***Hard Core*** é geralmente uma descrição de um *layout* posicionado e roteado de um ASIC/FPGA. Para atingir desempenho de processamento, baixo consumo e menor área, estes são otimizados para uma dada tecnologia. Logo, os *hard cores* garantem os tempos de propagação do núcleo (*timing*), além de proverem alta proteção à propriedade intelectual. Como conseqüência, a flexibilidade é mínima e este é fortemente dependente da tecnologia.

Os núcleos são usualmente interconectados em um SoC através de duas estruturas de interconexão: fios ponto-a-ponto dedicados e barramentos1, simples ou hierárquicos. Antes de descrever as estruturas de interconexão citadas acima, torna-se necessário definir alguns conceitos que caracterizam as mesmas.

- **Paralelismo:** relaciona-se à possibilidade de transferência e/ou recepção de dados entre dois ou mais pares de núcleos simultaneamente;
- **Consumo de energia:** determina a quantidade de energia consumida por um determinado circuito;
- **Escalabilidade:** refere-se à capacidade de interconectar componentes adicionais à estrutura de interconexão, sem comprometimento significativo no desempenho global do sistema (KUMAR, 2003);
- **Reusabilidade:** é a capacidade de utilizar uma dada estrutura de interconexão em projetos distintos. Essa estrutura deve proporcionar facilidades para que um grande número de núcleos possam trocar informações eficientemente. Isso, tanto para pares de núcleos como para comunicações concorrentes entre vários pares.

4.4 Interconexão de Núcleos Baseada em Fios Ponto-a-Ponto Dedicados

Na abordagem baseada em fios ponto-a-ponto dedicados (do inglês, *dedicated wires*), os núcleos são interligados diretamente um ao outro, ou seja, conexão ponto-a-ponto. O desempenho oferecido por essa estrutura pode ser considerado bom, pois cada comunicação ocorre independentemente das demais. Este tipo de estrutura de interconexão é eficaz se cada núcleo tem que se comunicar com um número pequeno de núcleos. Caso pretenda-se interligar um núcleo a vários outros, o número de fios dedicados aumenta proporcionalmente ao número de núcleos, o que pode gerar o congestionamento de fios em volta do mesmo (ZEFERINOa, 2003). Tal característica pode ser considerada uma limitação, à medida que se projeta SoCs com dezenas a centenas de núcleos (SEMATECH, 2002) (KUMAR, 2003). Outro fator limitante está no fato que o projeto deste tipo de estrutura é específico e, portanto, a reusabilidade é limitada. A Figura 4.3 ilustra a abordagem de interconexão descrita acima.

Figura 4.3 - Estrutura de Interconexão baseada em fios ponto-a-ponto dedicados (OST, 2004).

4.5 Interconexão de Núcleos Baseada em Barramentos Compartilhados

Uma estrutura de interconexão mais reutilizável é a baseada em barramentos compartilhados (do inglês, *shared data bus*). Além da reusabilidade, a baixa área de silício e a baixa latência contribuem para que esta abordagem seja a mais utilizada para interconexão de núcleos nos SoCs atuais (GUERRIER, 2000).

Um barramento consiste em um conjunto de fios que conectam diferentes núcleos do SoC, e sobre o qual dados são transmitidos e recebidos. Estes núcleos podem ser classificados como mestres e/ou escravos do barramento. Um núcleo mestre é uma unidade que controla a transferência em um barramento, ou seja, pode solicitar a transmissão ou a recepção dos dados através do barramento. Por outro lado, um componente escravo é a unidade que apenas responde às solicitações desses mestres (HWANG, 1993). Como exemplo, pode-se citar um microprocessador, atuando como mestre, e uma memória que efetua o papel de escravo do barramento. As informações são lidas ou escritas da/na memória a partir dos sinais gerados pelo microprocessador.

A maioria dos barramentos define um método de arbitragem responsável pelo controle de acesso dos núcleos mestres a si. Dentre os vários métodos citam-se o centralizado e o distribuído (HWANG, 1993). No método centralizado, um dispositivo denominado como árbitro ou controlador de barramento é responsável pela atribuição de acesso ao barramento. Neste caso, existem sinais de requisição e de permissão.

Quando o árbitro percebe uma requisição de direito de acesso, ele gera um sinal de permissão, quando for possível, ao mestre que solicitou o acesso ao barramento. No método de arbitragem descentralizado/distribuído, não há um árbitro. Uma das formas utilizadas para descentralizar a arbitragem é de delegar o monitoramento das linhas de requisição aos próprios núcleos do barramento. Desta maneira, cada núcleo sabe sua prioridade na ordem destas requisições e se podem ou não utilizar o barramento. A Figura 4.4 ilustra uma estrutura de interconexão baseada em barramento.

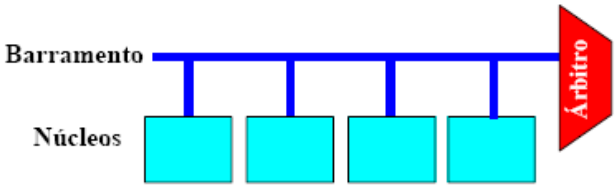


Figura 4.4 - Núcleos interligados através de um barramento, com arbitragem centralizada (OST, 2004).

Apesar de barramentos serem reutilizáveis, pode-se enumerar 3 desvantagens desta forma de interconexão: (i) ausência de paralelismo; (ii) baixa escalabilidade (iii) alto consumo de energia. O paralelismo inexiste em barramentos simples, pois apenas uma transação de comunicação é permitida por vez, dado que todos os núcleos compartilham o mesmo canal de comunicação.

Barramentos hierárquicos podem aumentar o número de transações simultâneas, porém de uma forma limitada (duas a três transações simultâneas). A escalabilidade é limitada a dezenas de núcleos, segundo (BENINI, 2002) (KUMAR, 2002) (GUERRIER, 2000). O consumo de energia é elevado devido à existência de fios longos (BENINI, 2001). O acréscimo de núcleos ao barramento aumenta a capacitância total do sistema, o que acaba reduzindo o desempenho elétrico.

Entre as arquiteturas de barramento intra-chip encontradas na literatura, destacam-se: AMBA da ARM (ARM, 2007), Avalon da Altera (ALTERA, 2007), CoreConnect da IBM (IBM, 2007) e OPB da Xilinx (XILINX, 2007). Geralmente, estas arquiteturas de barramento estão vinculadas à arquitetura de um processador, tal como o AMBA vinculado ao processador ARM, o CoreConnect vinculado ao processador PowerPC, o Avalon vinculado ao processador Nios e o OPB vinculado ao processador MicroBlaze. Diversos autores prevêem que as estruturas de interconexão citadas acima tornar-se-ão fatores limitantes para grandes projetos, em função do aumento da complexidade dos sistemas e do aumento dos requisitos de largura de banda. Uma possível solução para a integração de núcleos são as redes intra-chip.

5. COMPATIBILIDADE ELETROMAGNÉTICA (EMC)

Compatibilidade Eletromagnética é definida como a habilidade de um dispositivo, equipamento ou sistema funcionar satisfatoriamente em seu ambiente sem provocar perturbações intoleráveis a este ambiente ou a seus componentes (BOSCH, 2004).

O uso de sistemas eletrônicos em sistemas embarcados vem assumindo um papel indispensável, para o aumento do conforto, segurança e diminuição das emissões. Por outro lado, equipamentos e produtos eletrônicos estão presentes na vida diária de milhões de pessoas: telefones celulares, dispositivos sem-fio de entrada e saída de computadores, como mouses, teclados, redes *wi-fi*, sem contar as transmissões de rádio e televisão. Estes sistemas "sem-fio" (do inglês *wireless*) operam de maneira silenciosa e invisível, ininterruptamente para a conveniência e satisfação de necessidades humanas modernas (BIENERT, 2004). Apesar dos grandes benefícios e facilidades desses sistemas, eles introduzem ondas eletromagnéticas aos diversos ambientes em que operam. Neste contexto, compatibilidade eletromagnética vem se tornando uma habilidade cada vez mais crítica para garantir o correto funcionamento de todos esses sistemas em seus respectivos ambientes.

Num sistema embarcado, a fim de garantir o funcionamento adequado de cada componente e a operação dos sistemas como um todo, faz-se necessária a análise da interação entre esses componentes do ponto de vista elétrico e eletromagnético. Um componente não deve interagir com outro de forma que a operação de qualquer um deles seja prejudicada.

Essa interação pode ser abordada do ponto de vista de emissão ou de susceptibilidade à interferência. No primeiro caso o foco se concentra no que emana elétrica e eletromagneticamente do componente para o meio e no segundo, como o componente é afetado por sinais elétricos e eletromagnéticos que chegam até ele.

Determinados equipamentos são naturalmente fontes ou receptores de sinais eletromagnéticos. Transmissores e receptores de áudio ou dados, por exemplo. Outros sabidamente geram ruídos ou são sensíveis. Circuitos digitais, circuitos de chaveamento e sensores, respectivamente, são exemplos.

Os dispositivos podem ser separados em classes de acordo com o seu comportamento durante e após serem submetidos a um ambiente eletromagnético. Existem dispositivos que operam normalmente, mesmo quando submetidos a elevados níveis de potência. Outros podem ter sua operação degradada ou podem não operar enquanto estão submetidos à irradiação. Outros ainda podem passar a operar de forma degradada, mesmo depois de cessada a interferência.

O projeto dos equipamentos já deve prever o controle da emissão e da susceptibilidade elétrica e eletromagnética. Diversas técnicas já são conhecidas usando

leiautes, terras, blindagem, conectores apropriados, e filtragem nos pinos dos dispositivos.

Cada fornecedor e fabricante têm seus requisitos de EMC, assim como cada componente usualmente tem seus requisitos de EMC específicos, definidos por normas. O objetivo das normas é estabelecer padrões mínimos a serem atendidos dentro da indústria.

A Compatibilidade Eletromagnética (EMC) é uma matéria cada vez mais preocupante para qualquer pessoa que opere equipamentos e sistemas elétricos, eletrônicos ou de telecomunicações e, ainda que a sua designação seja algo crítico (pelo menos à primeira abordagem para pessoas sem formação específica em eletricidade), está associada a alguns efeitos que fazem parte do nosso dia a dia, e que são do conhecimento geral, decorrentes do fato de qualquer aparelho elétrico gerar perturbações radioelétricas. Exemplos desses efeitos são as perturbações visíveis na imagem de um televisor quando um veículo motorizado ruidoso (em radiação eletromagnética) passa nas proximidades ou quando ouvimos no nosso receptor de rádio perturbações oriundas de um aspirador elétrico.

Existem muitas outras causas dificilmente identificáveis, mas capazes de gerar efeitos imprevisíveis e que existem potencialmente em qualquer local ou ambiente, nomeadamente o lar, a indústria, os hospitais e os transportes aéreos, terrestres e marítimos. Nas últimas cinco décadas assistiu-se a uma preocupação relativamente crescente a este tema, comprovada na edição de publicações e normas técnicas sobre esta matéria e mais recentemente através dos requisitos das Diretivas Comunitárias Européias relacionadas à EMC, ou nos regulamentos das companhias de aviação comercial, que proíbem a utilização de aparelhos eletrônicos aos passageiros durante os vôos, para impossibilitar a ocorrência de fenômenos que interfiram com os sistemas de navegação aérea. Atualmente o tema Compatibilidade Eletromagnética, relaciona-se com a medição e a definição de limites para as várias perturbações geradas pelo aparelho ‘perturbador’, por um lado, e com a influência dessas perturbações sobre o aparelho ‘perturbado’, por outro.

5.1 Ambiente Eletromagnético

Um ambiente eletromagnético é definido por vários elementos, tais como a rede de energia elétrica, o tipo de edificação ou instalação onde o equipamento está inserido, outros equipamentos eletro-eletrônicos instalados e até o ambiente externo. Esse ambiente eletromagnético é alterado à medida que ocorrem reformulações no leiaute dos equipamentos, na edificação e/ou instalações e, principalmente, na instalação elétrica.

As fontes de EMI (*Electromagnetic Interference*) podem ser divididas em naturais e não naturais, ou seja, produzidas pelo homem (WESTON, 2001).

As fontes naturais podem ser desde ruído atmosférico, decorrente de descargas elétricas, até ruídos cósmicos provocados por explosões do sol. No caso de quedas de raios sobre a rede de distribuição de energia elétrica, o distúrbio é propagado pelos fios até a instalação interna, provocando diversos danos.

As fontes de EMI não naturais são geradas tanto dentro do ambiente em que o equipamento está instalado, como fora dele, em acionamentos de cargas indutivas como motores elétricos, cargas resistivas como aquecedores, lâmpadas fluorescentes, equipamentos médicos de diatermia por radiofreqüência (RF), aparelhos de micro-ondas, equipamentos de comunicação móvel, etc.

Um exemplo bastante conhecido é o de interferência causada por certos motores elétricos, resultante de arcos gerados nas escovas do motor. Como o comutador faz e desfaz o contato através das escovas, a corrente nos enrolamentos do motor (uma indutância) é interrompida, causando uma grande variação de tensão (e = L di/dt) através dos contatos (PAUL, 1992).

Em qualquer caso, sejam as fontes de ruídos naturais ou não, a qualidade da energia elétrica é comprometida e conseqüentemente a qualidade do atendimento ao consumidor também.

5.2 Interferência Eletromagnética (EMI)

O campo da Interferência Eletromagnética (EMI) e da Compatibilidade Eletromagnética (EMC) está ganhando importância com o avanço da tecnologia e a proliferação de um parque de equipamentos eletrônicos que emitem radiações eletromagnéticas muitas vezes não conhecidas, coabitando com outros muito susceptíveis.

Os equipamentos modernos são compostos de uma eletrônica que varia desde amplificadores analógicos sensíveis a microprocessadores sofisticados. Estes equipamentos podem então ser adversamente afetados por problemas de EMI (KIMMEL e GERKE, 1995).

Para entender como ocorre a EMI, as suas conseqüências e as possíveis medidas para minimizar ou extinguir seu efeito, é necessário compreender o modo de acoplamento[1] ou a ligação condutiva, qual a fonte causadora, os receptores dessas energias (equipamento vítima), os níveis de energia e a freqüência envolvida.

5.2.1 Emissão e Imunidade

A Figura 5.1 mostra as fontes de IEM, conduzida e irradiada, que podem estar presentes em um ambiente onde está instalado um equipamento (vítima).

[1] Acoplamento é definido como o caminho pelo qual parte ou toda energia eletromagnética de uma fonte especificada é transferida a outro circuito ou dispositivo (IEC 60050 (161) de 1990, pág. 17)

Os equipamentos utilizados em de transmissão de dados por onda-curtas ou microondas (ex. carga de tabelas de preços de produtos para balanças eletrônicas instaladas em supermercados, onde essa comunicação é feita com o servidor por micro-rádios RF) foram projetados para emitir intencionalmente energia eletromagnética de radiofreqüência. Os equipamentos de condicionamento de sinais analógicos (ex. circuito condicionamento de sinais de células de carga em balanças eletrônicas) que podem estar localizados próximos aos equipamentos acima citados, foram projetados para detectar e trabalhar com sinais que possuem amplitudes que variam de µV a mV. Essa combinação tem um potencial elevado para gerar problemas de EMC (ZEVZIKOVAS, 2004).

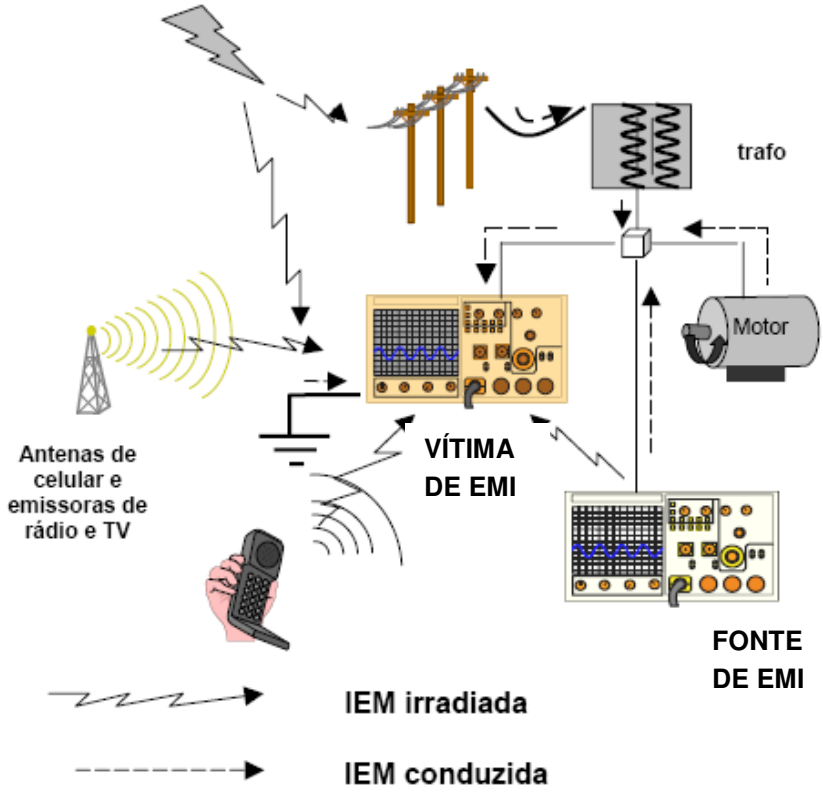


Figura 5.1 – Emissão e imunidade , fontes de EMI (ZEVZIKOVAS, 2004).

Tanto emissão como imunidade são importantes no projeto de um equipamento. No caso de emissões, o projetista deve respeitar os limites impostos pelas normas pertinentes, provendo o equipamento com filtros de linha na entrada da fonte de alimentação, projetando as placas de circuito impresso isolando, blindando e aterrando circuitos digitais que operem com altas freqüências (MHz), ou circuitos analógicos com impedâncias elevadas, além de outras técnicas de montagem do leiaute interno que minimizam as emissões e garantem um nível de emissão dentro dos limites das normas.

Imunidade é uma questão relacionada à interferência externa que pode prejudicar o funcionamento do equipamento. Pode-se empregar o termo

susceptibilidade[2] ao invés de imunidade. Estes dois termos realmente se referem à mesma coisa. Um equipamento estará susceptível acima de certo nível de EMI e imune abaixo desse nível. Por exemplo, quando se testa um equipamento com 2 kV, para ensaio de surto ou rajadas (*burst*), dizemos que ele está imune até esse nível, acima ele pode estar susceptível. Por isso o termo susceptibilidade se torna subjetivo. As normas militares têm usado o termo susceptibilidade por muitos anos e atualmente o termo imunidade tem se tornado mais comum, devido à atividade da União Européia (KIMMEL, 1995).

Quanto à imunidade, uma fonte com alta isolação através de um transformador com blindagem eletrostática e varistores conectados entre fase-fase e fase-terra são recursos que podem surtir bons efeitos.

5.2.2 Acoplamento

Como já visto anteriormente, o caminho para EMI pode ser conduzido, irradiado ou uma combinação dos dois. O meio conduzido pode envolver qualquer cabo de alimentação, entrada de sinal e terminais de terra de proteção.

O modo de acoplamento depende da freqüência e do comprimento de onda. Baixas freqüências propagam-se facilmente por meios condutivos, mas não tão eficientemente por meio irradiado. Altas freqüências se propagam eficientemente pelo ar e são bloqueadas pelas indutâncias da fiação (KIMMEL, 1995).

Os distúrbios podem ser acoplados e conduzidos para dentro de um equipamento por dois modos conhecidos: modo diferencial (entre fases) e modo comum (entre condutores fase e terra), conforme Figura 5.2.

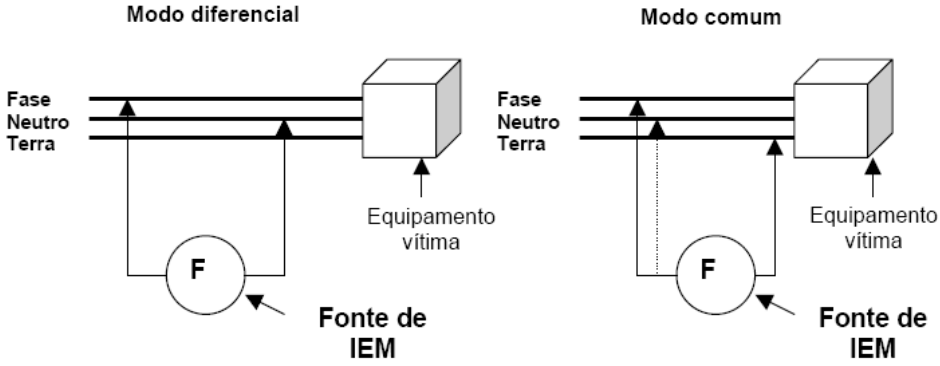


Figura 5.2 – Modos de acoplamento: Diferencial e Comum (ZEVZIKOVAS, 2004).

Além da irradiação não intencional, existe o problema de irradiação intencional de fontes, como estações de rádio, telefones celulares e transmissores em geral. Quando o receptor está próximo à fonte (campo próximo), campos elétricos e magnéticos são

[2] Susceptibilidade é definida como sendo a inabilidade de um dispositivo, equipamento ou sistema desempenhar seu funcionamento sem degradação na presença de um distúrbio eletromagnético (IEC 60050 (161) de 1990, pág. 6).

considerados separadamente. Quando o receptor está longe da fonte (campo distante), a irradiação é considerada como eletromagnética combinada (OTT, 1988).

Campo elétrico: quando dois circuitos estão acoplados por um campo elétrico o acoplamento pode ser representado por um fenômeno capacitivo.

Campo magnético: quando dois circuitos estão acoplados por um campo magnético o acoplamento pode ser representado por um fenômeno indutivo (indutância mútua entre eles).

O acoplamento através de transformadores: dois indutores intencionalmente acoplados, geralmente sobre um núcleo ferro-magnético, formam um transformador. Transformadores são freqüentemente usados para fornecer adequação entre níveis de tensão diferentes e isolação entre dois circuitos. Os transformadores não são ideais e apresentam uma capacitância entre o enrolamento primário e o secundário, o que permite que um ruído de alta freqüência atravesse o transformador (OTT, 1988).

5.3 Efeitos da Interferência Eletromagnética na Eletrônica

A EMI é a energia que causa resposta indesejável a qualquer equipamento e que pode ser gerada por diversas fontes e motivos como alguns exemplos vistos na Tabela 5.1 abaixo.

Classe	Tipo	Origem
Alta Energia	Subtensão (*Voltage Dips)*	• Chaveamento das fontes de alimentação; • Curto circuitos; • Acionamento de motores de alta potência.
Média Freqüência	Harmônicas	• Sistemas com semi-condutores de potência; • Fornos à arco elétrico; • Acionamento de relés; • Reatores eletrônicos; • Fontes chaveadas.
Alta Freqüência	Sobretensão	• Descargas atmosféricas, diretas ou induzidas; • Chaveamento de circuitos de controle; • Atuação de circuitos de proteção
	Descarga Eletrostática (ESD)	• Descargas eletrostáticas entre pessoas e equipamentos.

Tabela 5.1 – Distúrbios Elétricos mais comuns (SOUZA, 2006).

A convivência de equipamentos em diversas tecnologias diferentes somada à inadequação das instalações facilita a emissão de energia eletromagnética e com isto podemos ter problemas de compatibilidade eletromagnética (habilidade de um equipamento funcionar satisfatoriamente sem interferir eletromagneticamente nos equipamentos próximos e ser imune à interferência externa de outros equipamentos e do ambiente), onde o funcionamento de um equipamento pode afetar o outro. Isto é muito comum nas indústrias e fábricas, onde a EMI é muito freqüente em função do maior uso de máquinas e motores e em redes digitais e de computadores próximas a essas áreas.

O maior problema causado pela EMI são as situações esporádicas e que degradam aos poucos os equipamentos e seus componentes. Os mais diversos problemas podem ser gerados pela EMI, por exemplo, em equipamentos eletrônicos, podemos ter falhas na comunicação entre dispositivos de uma rede de equipamentos e/ou computadores, alarmes gerados sem explicação, atuação em relés que não seguem uma lógica e sem haver comando para isto e, queima de componentes e circuitos eletrônicos, etc. É muito comum a presença de ruídos na alimentação pelo mau aterramento e blindagem, ou mesmo erro de projeto (SOUZA, 2006).

A EMI é muito importante principalmente em sistemas digitais e analógicos onde estamos falando de freqüências de 30 a 300MHz, ou seja, superiores a VHF (*very high frequency*). Vale lembrar que estamos falando de pulsos rápidos da ordem de ns e qualquer condutor, como, por exemplo, a trilha de uma placa de circuito impresso passa a ser uma antena, sem contar os efeitos por irradiação de sinais e acoplamentos parasitas.

Em geral, em freqüências elevadas, os condutores se aproximam ainda mais do comportamento de uma antena, o que nos ajuda a entender porque os problemas de emissão de EMI se agravam em redes que operam em altas velocidades. E ainda, qualquer circuito eletrônico é capaz de gerar algum tipo de campo magnético ao seu redor e seu efeito vai depender de sua amplitude e duração.

Um exemplo típico de como a EMI pode afetar o comportamento de um componente eletrônico, é um capacitor que fique sujeito a um pico de tensão maior que sua tensão nominal especificada, com isto pode-se ter a degradação do dielétrico (a espessura do dielétrico é limitada pela tensão de operação do capacitor, que deve produzir um gradiente de potencial inferior à rigidez dielétrica do material), causando um mau funcionamento e em alguns casos a própria queima do capacitor. Ou ainda, podemos ter a alteração de correntes de polarização de transistores levando-os a saturação ou corte, ou dependendo da intensidade a queima de componentes por efeito joule.

Fatores que contribuem para a interferência eletromagnética:

Os principais fatores são:

- Tensão
- Freqüência
- Aterramento

- Os componentes eletrônicos
- Circuitos impressos
- Desacoplamentos

5.4 Interferência Conduzida versus Irradiada

As Emissões eletromagnéticas são interferências causadas por um equipamento, dispositivo ou sistema, as quais são divididas em dois tipos, de acordo com o meio de propagação (SOUZA, 2006):

a) Interferências irradiadas;

b) Interferências conduzidas.

Interferência Conduzida:

Este tipo de interferência ocorre de forma mais usual na rede elétrica de sinal alternado (KRAUSS, 1999). Como esta rede alimenta vários sistemas ruidosos (motores, contactoras, etc.), estes ruídos são conduzidos pela rede elétrica podendo interferir em outros sistemas, equipamentos e circuitos que também são alimentados pela mesma rede.

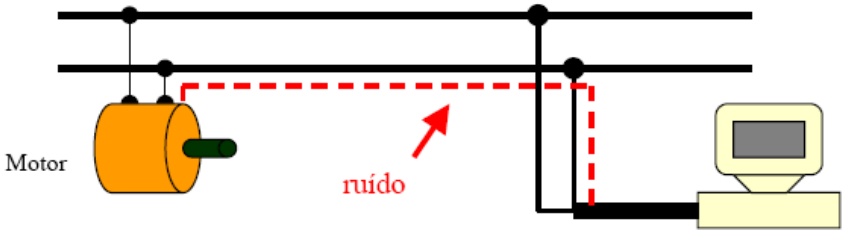


Figura 5.3 - Interferência de modo conduzido (SOUZA, 2006).

A Figura 5.3 mostra um exemplo de interferência eletromagnética ocorrendo pele rede elétrica, onde uma máquina elétrica afeta o funcionamento de um equipamento eletrônico. Este tipo de acoplamento é muito comum em ambientes industriais, comerciais e domésticos.

Interferência Irradiada:

Neste tipo de acoplamento tem-se um elemento irradiando um campo eletromagnético (fonte de interferência), que se propaga na atmosfera e atinge equipamentos eletrônicos e sistemas de telecomunicações (receptores de interferência) (KRAUSS, 1999).

Estas ondas eletromagnéticas são recebidas e produzem ruídos no interior destes equipamentos e sistemas.

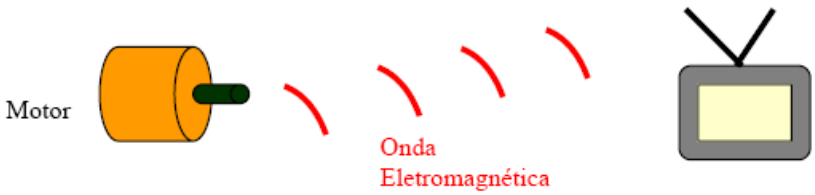


Figura 5.4 – Interferência de modo irradiado (SOUZA, 2006).

É importante lembrar que a teoria elementar de antenas estabelece que campos eletromagnéticos na proximidade da fonte são considerados **CAMPOS DE INDUÇÃO** e campos distantes da fonte são denominados de **CAMPOS DE IRRADIAÇÃO** (SOUZA, 2006).

Elementos de circuitos energizados possuem campos elétrico e magnético próprios, estes campos podem produzir interferência de um sistema em outro (Figura 5.5). As presenças de campos elétricos e magnéticos podem induzir o aparecimento de correntes e tensões elétricas espúrias em circuitos elétricos e eletrônicos (KRAUSS, 1999).

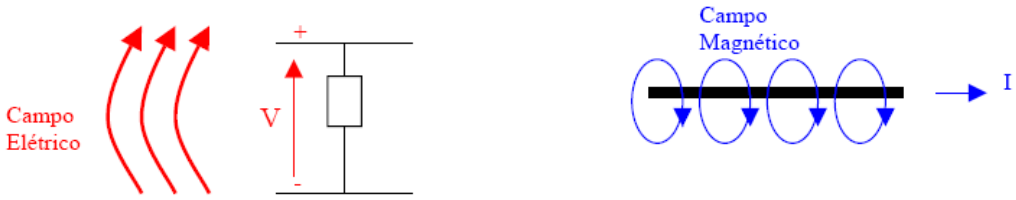


Figura 5.5 – Interferência e modo induzido (SOUZA, 2006).

Um critério para definir se o campo eletromagnético é de indução ou de irradiação é determinado pela comparação entre a distância da fonte e valores de referência (RIBEIRO, 2004):

O valor de referência é dado por:

ro = $\lambda/2\pi$, onde λ é o comprimento de onda do sinal interferente.

Se a distância em relação à fonte da interferência for:

$d \leq ro/100$, temos campos de indução.

Se a distância em relação à fonte de interferência for:

$d \geq ro/100$, temos campo de irradiação.

5.5 Normas IEC 62.132, FCC e CE

5.5.1 Norma CE

É importante inicialmente ressaltar que a marcação CE é apenas um símbolo e não deve ser associada a nenhuma sigla, mesmo que na sua origem francesa signifique Comunidade Européia (MAGNUS, 2001).

A marcação CE estampada em um produto indica que o fabricante declarou publicamente que está atendendo as normas ou diretivas pertinentes. Ao fazer isto, o fabricante está responsabilizando-se a responder judicialmente quando esta declaração não for verdadeira, isto é, quando o seu produto não atender ao que foi declarado. A marcação CE está apresentada na Figura 5.6.

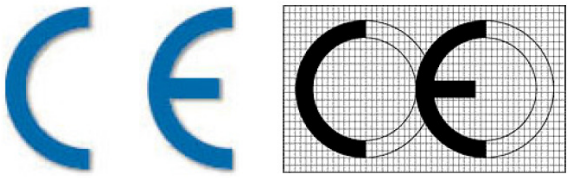


Figura 5.6 – Marcação CE (MAGNUS, 2001).

Quando um produto possui a marcação CE, as autoridades competentes dos países membros da AEE assumem que todos os requisitos essenciais das normas ou diretivas aplicáveis foram atendidos. Caso tenha sido publicada uma diretiva ou norma aplicável ao produto e este não apresentar a marcação CE, sua comercialização é interditada.

Embora a marcação CE permita a entrada de um produto no mercado europeu, ela não garante a qualidade deste produto, mas apenas os requisitos essenciais de segurança em relação ao usuário. A qualidade de um produto só pode ser garantida se este possuir uma marca de conformidade emitida por órgão legalmente capacitado para tal ato. No Brasil o órgão que controla a certificação é o Instituto Nacional de Metrologia, Normalização e Qualidade Industrial (INMETRO), através do Sistema Brasileiro de Certificação (SBC).

5.5.2 Norma FCC

Federal Communications Commission (FCC) é a agência federal dos Estados Unidos da América que regula o setor de telecomunicações. A FCC é a comissão que determina padrões de controle do nível de interferência eletromagnética (radiação ou emissão) de dispositivos eletrônicos de telecomunicações e de equipamentos elétricos dentro dos limites do EMC. Equipamentos de Comunicação ou de Tecnologia da Informação tais como telefones, satélites, radio e tv ou qualquer equipamento que opere no sistema fixo comutado, no espectro radio elétrico ou no espectro eletromagnético devem possuir a marca FCC para ter sua comercialização autorizada nos Estados

Unidos. Na Figura 5.7 mostra a marcação que é incorporada aos equipamentos que seguem as normas da FCC.

Figura 5.7 – Marcação FCC (FCC, 2007).

5.5.3 Norma IEC 62.132

A IEC (*International Electrotechnical Commission*) é uma organização mundial para compreender da estandardização de todos os comitês eletrotécnicos nacionais (comitês nacionais do IEC). O objeto do IEC é promover cooperação internacional em todas as perguntas a respeito da estandardização nos campos elétricos e eletrônicos.

Apresentação da família de normas IEC 62.132 (IEC, 2004) para medições de imunidade eletromagnética irradiada e conduzida em circuitos integrados na faixa de freqüências entre 10kHz e 1GHz.

IEC 62132 é uma família de normas no qual é composta por 5 partes:

- IEC 62.132-1: Condições e definições gerais;
- IEC 62.132-2: Medições de imunidade EM irradiada em CI's na faixa de freqüências de 150kHz a 1GHz;
- IEC 62.132-3: Imunidade de CI's para teste dos distúrbios em bandas estreitas de freqüência causados por injeção de corrente na faixa de freqüências entre 10kHz a 1GHz;
- IEC 62.132-4: Injeção direta de RF, para medição da imunidade de CI's contra distúrbios de RF para freqüências de até 1GHz;
- IEC 62.132-5: Medição da imunidade EM conduzida em CI's na faixa de freqüências entre 150kHz a 1GHz, através do método da Gaiola de Faraday de bancada.

Como estudos de caso, focamos nossos trabalhos nas parte 1,2 e 4 da norma IEC 62.132, onde será melhor explanado nas seções a seguir.

5.5.3.1 IEC 62.132-1

O IEC 62132-1 (IEC, 2004) fornece as informações e definições gerais para a medida da imunidade eletromagnética conduzida e irradiada em circuitos integrados (CI). Para isso a seguir será comentado a terminação dos pinos não utilizados no teste, a construção de uma placa para testes,

a) TERMINAÇÃO DOS PINOS NÃO UTILIZADOS NOS TESTES:

Os pinos do CI em teste devem ter a terminação de acordo com a Tabela 5.2 abaixo para diferentes tipos de pinos, com exceções aos pinos que exigem certa funcionalidade pelo fabricante:

IC Pin Type	Pin Loading
Analogue	
- Supply	as stated by the manufacturer
- Input	10 kΩ to ground (Vss) unless the IC is internally terminated
- Output signal	10 kΩ to ground (Vss) unless the IC is internally terminated
- Output power	Nominal loading as stated by the manufacturer
Digital	
- Supply	as stated by the manufacturer
- Input	Ground (Vss) or 10 kΩ to supply (Vdd) if the input cannot be grounded, unless the IC is internally terminated
- Output	47 pF to ground (Vss)
Control	
- Input	Ground (Vss) or 10 kΩ to supply (Vdd) if the input cannot be grounded, unless the IC is internally terminated
- Output	as stated by the manufacturer
- Bi-directional	47 pF to ground (Vss)
- Analogue	as stated by the manufacturer

Tabela 5.2 – Terminação dos pinos não utilizados no CI sobre teste (IEC, 2004).

Os pinos que não caem em algumas das categorias listadas acima, terão terminações como requerido funcionalmente e indicado no relatório de teste.

Estes são valores padrão recomendados; se outros valores forem mais apropriados para um CI em particular, podem ser substituídos pelos valores mostrados na Tabela 5.2 e posteriormente devem estar indicados no relatório de teste.

b) PLACA DE TESTE:

A placa de circuito impresso (placa de teste) usada para testar a imunidade à RF, pode depender do método específico de medição dos distúrbios, mas no geral, todas as placas de teste usarão um ótimo plano de terra comum ao RF. Sendo assim temos as principais características da PCB de teste, conforme detalhe na Figura 5.8:

- Dimensões: 10cm x 10cm;
- Adicionar furos nos cantos da placa;
- As bordas da placa devem ser estanhadas em 5mm, e ligadas ao terra para contato com a TEM *cell*;
- As vias, devem ter uma distância mínima das bordas de 5mm;

- A placa de teste deve ter apenas o CI em teste no *layer* TOP e o restante dos componentes no *layer* oposto (*layer* 4 ou BOTTON).

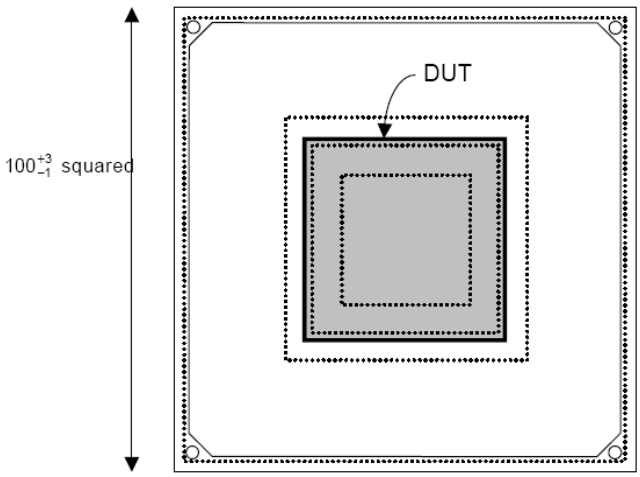


Figura 5.8 – Detalhe do DUT (*Device Under Test*) placa de teste (IEC, 2004).

Na Figura 5.9 pode-se observar que a placa em teste deve possui no mínimo 4 *layers* (camadas). No *layer* TOP (parte de cima da placa) é colocado somente o CI, ou os CI's que estarão sobre teste (DUT – *Device Under Test*). Na *layer* TOP não deverá conter nenhuma trilha de sinal, somente um plano de terra (GND) que cobre toda a placa.

No segundo *layer*, logo abaixo do TOP está contido o plano de alimentação (VCC) que cobre todo o *layer*, não podendo conter nenhum outro sinal.

Nos *layers* seguintes até o *layer* BOTTON (parte de baixo da placa) deverá estar contido os demais sinais utilizados pela placa, podendo também conter sinais de VCC e GND.

No *layer* BOTTON deverão estar contidos os demais componentes da placa bem como qualquer tipo de sinal e/ou alimentação, devendo também conter um plano de terra para reforçar a blindagem.

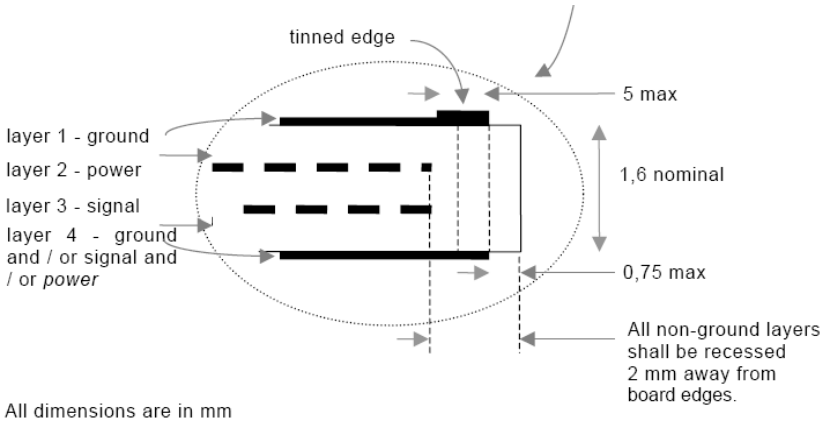


Figura 5.9 – Detalhe dos *layers* e respectivos sinais da placa de teste (IEC, 2004).

Vias (furos de passagem):

Todas as vias na posição **1** (Figura 5.10), que são as contidas ao redor das bordas da placa, terão um diâmetro do furo de 0.8mm. Todas as vias restantes terão um diâmetro de 0.2mm, conforme Tabela 5.3 abaixo:

Posição da VIA	Localização
1	Tudo ao redor das bordas da placa
2	Nas proximidades do DUT
3	Embaixo do emcapsulamento do DUT

Tabela 5.3 - Posicionamento das vias (IEC, 2004).

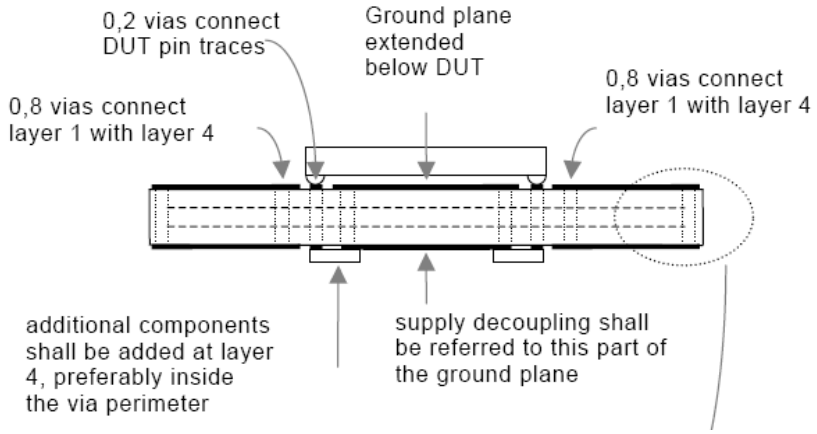


Figura 5.10 – Posicionamento das vias (IEC, 2004).

c) PROCEDIMENTO DE TESTE:

- Verificar a operação da placa de testes sem a aplicação de RF;
- A escala de freqüência destas medidas é geralmente 150kHz a 1GHz. Na prática a escala de freqüência testada depende das freqüências de interrupção do CI, e do método de injeção. Sugerindo uma etapa de intervalos de freqüência a ser aplicado, temos a Tabela 5.4;
- As freqüências críticas tais como freqüências de clock, freqüências do sistema e dos dispositivos auxiliares, por exemplo, devem ser testadas usando intervalos de freqüência menores, de forma a procurar pontos de susceptibilidade;

Faixa de Freqüência (MHz)	0,15 -1	1 -100	100 – 1000
Intervalos Lineares de Freqüência (MHz)	≤ 0,1	≤ 1	≤ 10
Intervalos Logaritmos de Freqüência	Incremento de 5%		

Tabela 5.4 - Intervalos de freqüência aplicados nos testes (IEC, 2004).

- **Modulação em Amplitude**: O sinal de distúrbio usado estará do acordo às exigências do método do teste escolhido, por exemplo, CW (*continuous wave* ou onda contínua), de amplitude 80% e modulada por uma onda senoidal AM (*amplitude modulation* ou modulação em amplitude) de 1kHz ou de um pulso modulado com a taxa da repetição 1kHz;
- **Nível de Potência:** Depende da definição usada do sinal de distúrbio, usada em várias partes desta norma, tais como qualquer pico de sinal RF ou a potência da portadora mantida (comum na maioria dos geradores de RF) pode ser usado nos testes conforme sinal mostrado na Figura 5.11 abaixo:

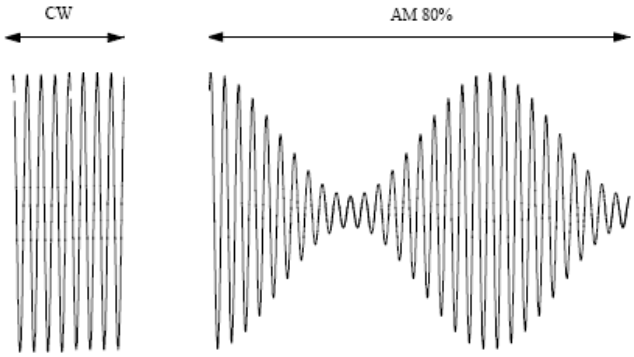


Figura 5.11 – Sinal AM modulado 80% (IEC, 2004).

d) Relatório de Testes:

Os resultados coletados a partir dos testes são classificados da seguinte forma:

- **CLASSE A:** Todas as funções do CI executaram como projetadas durante e após a exposição a um distúrbio;

- **CLASSE B:** Todas as funções do CI executaram como projetadas durante a exposição, entretanto, uma ou o mais delas podem ir além da tolerância especificada. Todas as funções retornam

automaticamente dentro dos limites normais depois que a exposição foi removida;

- **CLASSE C:** Uma função do CI não executa como projetada durante a exposição, mas retorna automaticamente à operação normal depois que a exposição é removida;

- **CLASSE D:** Uma função do CI não executa como projetada durante a exposição e não retorna à operação normal até que a exposição seja removida e o CI seja resetado pela ação do operador;

- **CLASSE E:** Uma ou mais funções do CI não executam como projetadas durante e após a exposição e não pode ser retornado à operação apropriada.

5.5.3.2 IEC 62.132-2

Este procedimento de medida define um método para medir a imunidade aos distúrbios eletromagnéticos irradiados do circuito integrado (CI). O CI que está sendo avaliado é montado em uma placa de teste (PCB) no qual é encaixada a uma porta de acoplamento específica contida na *transverse electromagnetic* (TEM) ou *wideband gigahertz* TEM (GTEM) *cell* (IEC, 2004).

A placa de teste não é inserida dentro da TEM ou GTEM, como no uso convencional, mas transforma-se uma peça da parede de célula. Este método é aplicável a toda célula TEM ou GTEM modificada para incorporar a parta em sua parede parede; entretanto, a resposta medida do CI será afetada por muitos fatores. O fator preliminar que afeta a resposta do CI é o ao afastamento do *septum* até placa de teste.

Este procedimento foi desenvolvido para medir a imunidade EM (eletromagnética) irradiada em CI's na faixa de freqüências de 150kHz to 1GHz (ou limitados pela GTEM) pelos métodos TEM-*cell* e *wideband* TEM-*cell* (GTEM).

TEM-*cell*: A câmara usada para este procedimento de teste possui uma abertura de 10cm x 10cm feito sob medida para acoplar-se com a placa de teste que está o CI a ser testado, conforme mostrado na Figura 5.12.

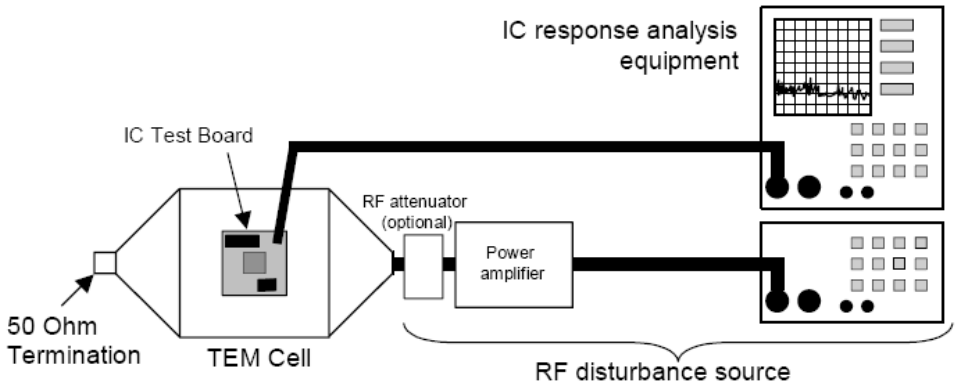


Figura 5.12 – TEM-*Cell* (IEC, 2004).

Wideband TEM/GTEM-*cell*: Semelhante a TEM-*cell* com a diferença na faixa de freqüências que pode passar dos 2GHz indo até a especificação de fabricação da mesma, mostrado na Figura 5.13 abaixo:

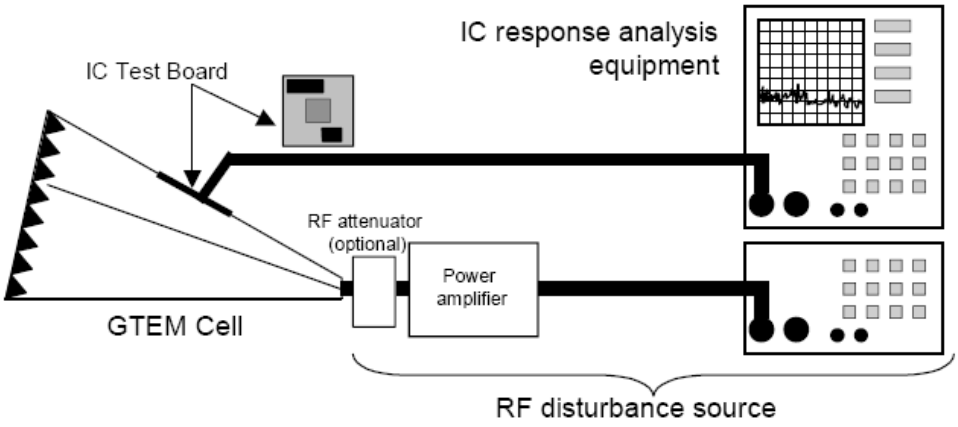


Figura 5.13 - Wideband TEM/GTEM-*cell* (IEC, 2004).

Os níveis de imunidade aceitos pelo CI, se existir, devem ser concordados entre os fabricantes e os usuários dos CI's. Esses níveis aplicam-se as medidas sobre a escala de freqüência de 150 quilohertz a 1 gigahertz nas unidades de Volts por metro [3] (V/m).

Como uma alternativa, os níveis de aceitação para imunidade do CI, podem ser definidos em um nível fixo da potência do campo e podem ser mantidos sobre determinadas escalas de freqüência, conforme as exigências da aplicação do sistema e/ou do CI.

5.5.3.3 IEC 62.132-4

O seguinte procedimento descreve um método para medir a imunidade do CI na presença de distúrbios de RF conduzidos, por exemplo, resultantes dos distúrbios

[3] Um volt por metro (V/m) é a intensidade de um campo elétrico, que aplica uma força de intensidade 1 Newton sobre um corpo eletrizado com quantidade de carga de 1 Coulomb.

irradiados de RF. Este método garante um grau elevado de repetibilidade e a correlação de medidas de imunidade (IEC, 2004).

Esta norma estabelece uma base comum para a avaliação de dispositivos semicondutores usados em equipamentos que trabalham em um ambiente que seja sujeito às ondas eletromagnéticas de rádio freqüência não desejada.

O nível eletromagnético mínimo para a imunidade requerida para um CI, depende do nível máximo permitido de distúrbio que um sistema eletrônico pode ser submetido. O valor do nível de imunidade é dependente dos parâmetros específicos do sistema e da aplicação.

A injeção direta de RF, para medição da imunidade de CI's contra distúrbios de RF, é utilizada uma faixa de freqüências de 150 kHz de até 1 GHz, onde o gerador de RF que varia a freqüência fornece o distúrbio do RF que é amplificado pelo amplificador conectado à saída do gerador conforme observado na Figura 5.14.

O acoplador direcional e os medidores de potência de RF são usados para medir qual a potência real é injetada ao dispositivo sob teste (DUT). Na porta de injeção de RF o mesmo é fornecido à placa de teste e um bloqueio de DC é colocado para evitar de se fornecer algum distúrbio DC na saída do amplificador para a placa.

Para monitorar o comportamento do DUT preferencialmente usa-se um osciloscópio ou o outro dispositivo de monitoração com uma função de passa/falha.

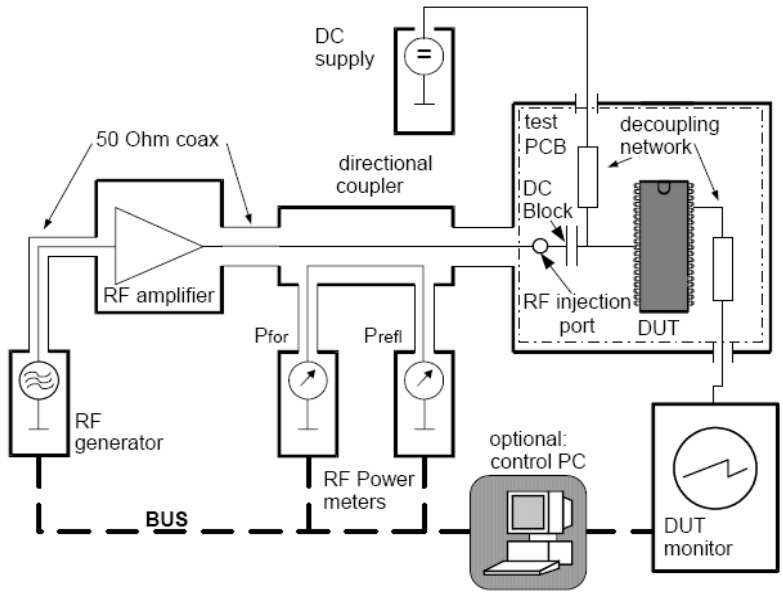


Figura 5.14 – Organização do hardware para testes de imunidade a distúrbios conduzidos de RF (IEC, 2004).

Para a injeção direta de RF em um único pino do CI, temos as seguintes características:

- Para uma seletividade mais elevada do teste, a potência de RF injetada na porta de injeção, é aplicada diretamente a um pino do CI como pode ser observado na Figura 5.15;

- Um capacitor é usado como um bloqueio DC, enquanto o resistor é usado para limitar a corrente ou para simular uma carga real ao sistema. Por definição, o valor típico do capacitor é de 6,8nF enquanto que o resistor pode ser 100 ohms, especificado na norma IEC 61.967.

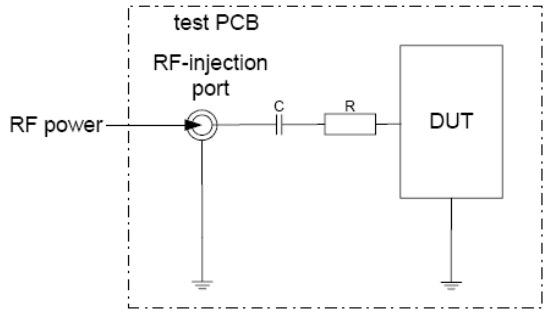


Figura 5.15 - Injeção de RF em um único pino do CI (IEC, 2004).

Para a injeção direta de RF em mais de um pino do CI, temos as seguintes características:

- Se dois ou mais pinos do CI são utilizados para transferir a informação no modo diferencial, na forma digital ou analógica, múltiplos pinos podem ser usados conforme mostrado na Figura 5.16;
- A injeção direta de RF em múltiplos pinos despreza a dependência de fase dos efeitos causados pelo modo diferencial;

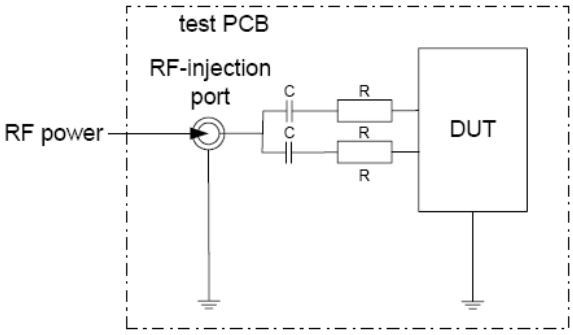


Figura 5.16 - Injeção de RF em dois ou mais pinos do CI (IEC, 2004).

- O nível de potência aplicado no teste, depende do CI e do pino em que se está utilizando para injeção de RF;
- A máxima potência de da onda contínua (CW) aplicado a um pino do CI sem proteção externa é de 5watts ou 37dBm[4];

[4] O decibel (dB) é uma notação destinada a medir níveis de potência de forma relativa. A medida mais comum para expressar a potência de rádio freqüência (RF) é o dBm (dB miliwatt). Zero dBm é definido como 1mW de potência.

- Se o pino do CI é projetado para operar com proteção externa, os níveis de potência podem ser aumentados conforme valores da Tabela 5.5 abaixo;

Potência (Watts)	Proteção Externa	Exemplos de Dispositivos
1....5	Nenhuma ou somente um pequeno capacitor	Grandes chaves(*switches*), circuitos de alimentação, *transceiver* de barramento (LIM, CAM etc..)
0,1....0,5	R-L-C, filtro passa baixa	Dispositivos de condicionamento de sinal, sensores, *drivers* de linhas de comunicação
0,01....0,05	Sem conexão direta	Microntroladores, microprocessadores e memórias

Tabela 5.5 – Valor dos níveis de potência que podem ser utilizados com suas respectivas proteções (IEC, 2004).

- Para o teste do DUT, uma onda contínua e/ou modulação em amplitude (AM) de acordo com o usuário, deve ser usada. Por padrão usa-se um sinal AM com modulação de 1kHz 80% é recomendado.

Equipamentos de teste:

A fonte de RF consiste em um gerador de sinais de RF e um amplificador de potência de RF que deve fornece a potência suficiente mesmo em uma carga mal combinada.

É recomendado que se utilize um amplificador de alta potência (de 10W...50W) para a máxima transferência de potência. A impedância de saída do amplificador deve ser de 50ohms para absorver as ondas refletidas. Se o amplificador não contém essa impedância, um atenuador deve ser colocado entre o amplificador e os cabos de transmissão. A emissão espúrios da fonte de RF, será de pelo menos 20dB abaixo do nível da portadora.

Para a medida de potência durante a modulação, recomenda-se usar um medidor de potência com capacidade de medidas de pico.

Estrutura de testes:

A configuração para a injeção direta de RF, consiste em duas partes. A primeira parte não está na placa do teste e compreende:

- Fonte de RF de potência;
- Cabos coaxiais;
- Conectores de RF;
- Acoplador direcional para medir a potência injetada.

A segunda parte da configuração de injeção direta de RF é colocada diretamente na placa de teste que compreende:

- Uma (ou mais) porta de injeção de RF para conectar os cabos coaxiais e sua respectiva trilha de transmissão até o CI;
- O uso de uma PCB com um plano de terra comum ao RF é fortemente recomendado;
- O DUT deve ser colocado na PCB sem soquetes porque a maioria dos soquetes tem uma indutância significativa que afeta o teste (por exemplo, 10nH em 1GHz fazem XL = 63Ω);
- A conexão da extremidade da linha da transmissão (porta de injeção de RF) até o pino do DUT ocorre através de um resistor para limitar a corrente (de até 100 ohms dependendo da potência do sinal injetado e da impedância da trilha) e um capacitor de 6,8nF para bloquear qualquer espúrio DC vindo do amplificador, que são colocados em série com o pino do DUT conforme visto na Figura 5.17;
- A trilha do conector da porta de injeção de RF ao capacitor de bloqueio de DC, deve ser de 50Ω e a extremidade da trilha de transmissão ao pino do DUT deve ser o mais curta possível.

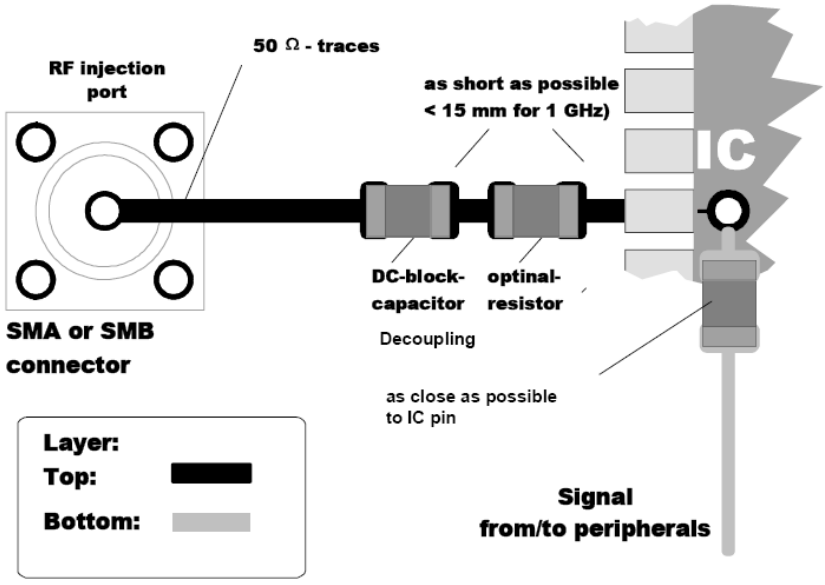


Figura 5.17 - Exemplo de roteamento de uma porta de injeção a um pino do CI (IEC, 2004).

Sugestões para a melhor instalação da estrutura de teste de RF:

A injeção de RF deve ser feita o mais perto possível do dispositivo sob teste sem deixar que a trilha do sistema de transmissão não possua os 50 Ω. Isto pode ser realizado de duas maneiras:

1) Usar uma PCB o menor possível e colocar o conector de transmissão de RF o mais perto do dispositivo possível;

2) Usando uma PCB grande que possua trilhas na placa com impedância de 50 Ω para manter o VSWR (*Voltage Standing Wave Ratio* ou relação de tensão da onda estacionária) total abaixo de 1,2V.

O caso a) pode ser usado para CI's com um número baixo de pinos, enquanto que o caso b) leva vantagem para testar de CI's com um número elevado dos pinos.

a) Neste caso, o conector do cabo coaxial, pode ser colocado na placa conforme Figura 5.18 abaixo:

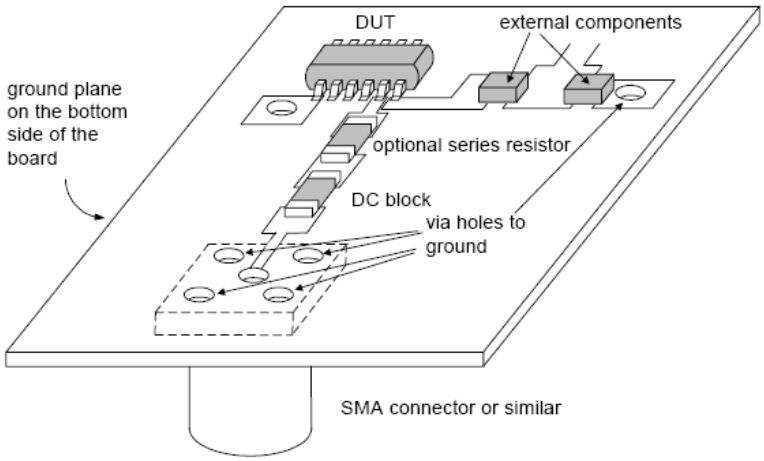


Figura 5.18 - Detalhe da PCB de teste para o caso (a) (IEC, 2004).

b) Neste caso, a placa específica do CI pode ser conectada às trilhas da placa principal grande, pelos pinos de contato do conector. A conexão ao terra é fornecida pelo anel interno dos pinos de contato do conector para evitar *slits* (brechas) significativos entre o plano de terra da placa principal e o plano de terra da placa do DUT conforme Figura 5.19.

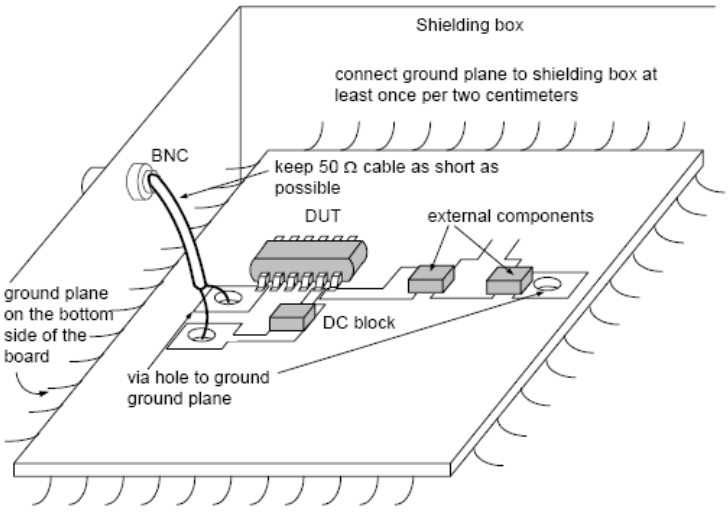


Figura 5.19 - Detalhe da PCB de teste para o caso (b) (IEC, 2004).

6. PLATAFORMAS DE TESTE E INJEÇÃO DE FALHAS PARA SOCS

6.1 Introdução

A avaliação da dependabilidade envolve o estudo de falhas e erros (HSUEH, 1997). A natureza destrutiva de um ruído elétrico e uma latência longa do erro faz difícil de identificar as causas das falhas no ambiente operacional. É particularmente difícil recriar um cenário de falhas para um sistema grande e complexo.

Para identificar e compreender falhas em potencial usa-se uma aproximação baseada em experimentos para estudar a dependabilidade de um sistema. Tal aproximação é aplicada não somente durante as fases da concepção e do projeto, mas também durante o protótipo e as fases operacionais (PRADHAN, 1995) (IYER, 1996).

Para fazer uma análise baseado em experimentos, deve-se compreender primeiramente a arquitetura, a estrutura, e o comportamento do sistema. Especificamente, necessita-se saber sua tolerância para falhas e seus tipos de falhas, incluindo sua capacidade de detecção e seus mecanismos de recuperação (ABRAHAM, 1995), onde necessita-se de instrumentos específicos e de ferramentas para injetar falhas que criam falhas e/ou erros, e monitoram seus efeitos.

6.2 Ambiente de Teste e Injeção de Falhas

A Figura 6.1 mostra um ambiente da injeção da falha genérico, que consiste basicamente em um sistema alvo mais um injetor de falha, biblioteca de falhas, biblioteca de carga de trabalho ou gerador de injeção, controlador, monitor, coletor de dados, e analisador dos dados (HSUEH - 1997).

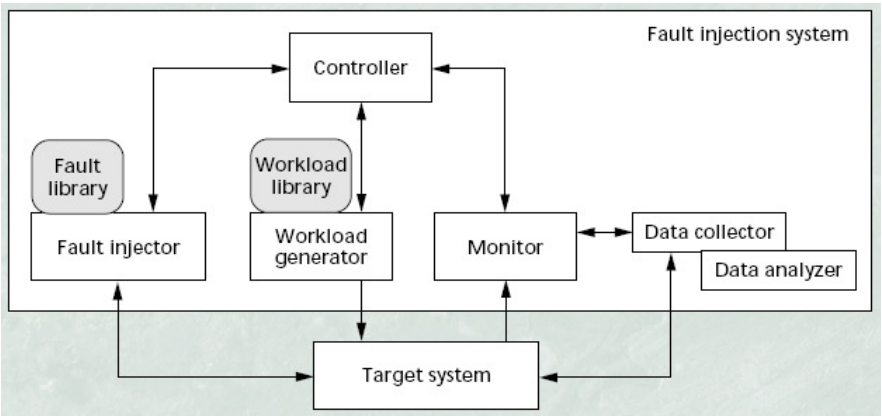


Figura 6.1 – Componentes Básicos de um ambiente de injeção de falhas (HSUEH, 1997).

O injetor de falhas injeta falha no sistema alvo enquanto executa comandos do gerador do injeção (aplicações, *benchmarks*, ou injeções artificiais). O monitor segue a execução dos comandos e inicia o levantamento de dados sempre que necessário. O coletor de dados executa o levantamento de dados *online*, e o analisador dos dados, que pode ser *offline*, executa o processamento e a análise dos dados. O controlador controla o experimento.

Fisicamente, o controlador é um programa que pode funcionar no sistema alvo ou em um computador separado. O injetor de falha pode ser hardware ou software embutido e customizado. O injetor de falha por si próprio pode suportar diferentes tipos de falhas, localizações de falha, tempo da falha, e estruturas apropriadas de configuração de software e de hardware; Valores de que são extraídos de uma biblioteca de falhas. A biblioteca de falha na Figura 6.1 é um componente separado, no qual permite uma flexibilidade e uma portabilidade maior.

O gerador de injeção, o monitor, e outros componentes podem ser executados da mesma maneira.

Escolher entre o método de injeção por hardware ou por software, depende do tipo de falhas que se está interessado a reproduzir. Por o exemplo, deseja-se obter falha de *stuck-at* (falhas que forçam um valor permanente em um ponto do circuito), um injetor de falhas por hardware é preferível porque se pode controlar a posição da falha.

A injeção de falhas permanentes que usam métodos por software causa um elevado *overhead* ou simplesmente são impossíveis de serem implementados, dependendo da falha. Entretanto, se o interesse for à corrupção dos dados, a aproximação por software pode ser suficiente.

Algumas falhas, como *bit-flips* em memória, pode ser injetado por um ou outro método. Em casos onde se deseja esse tipo de falhas, exigências adicionais, tais como o custo, exatidão, intrusão e repetibilidade podem levar a uma escolha por um por outro método. A Tabela 6.1 resume as falhas estudadas e os métodos de injeção.

HARDWARE	SOFTWARE
Open	Corrupção de dados armazenados (como registradores, memórias e disco)
Bridging	
Bit-flip	Corrupção de dados na comunicação (como barramento e rede de comunicação)
Correntes Espúrias	
Súbito de tensão	Manifestação de defeitos de software (como nível de máquina e nível de usuário)
Stuck-at	

Tabela 6.1 – Resumo dos Métodos de injeção (HSUEH, 1997).

6.3 Teste e Injeção de Falhas por Hardware

A implementação de injeção de falhas por hardware usa um hardware adicional para introduzir falha no hardware de sistema alvo (HSUEH - 1997). Dependendo das falhas e suas posições, os métodos de implementação por hardware da injeção de falha, dividem-se em duas categorias:

1. **Injeção de falhas por hardware com contato:** O injetor tem o contato físico direto com o sistema alvo, produzindo tensão ou variações de corrente externamente ao CI alvo. Como exemplos, podemos citar os métodos que usam pontas de prova e soquetes.
2. **Injeção de falhas por hardware sem contato:** O injetor não tem nenhum contato físico direto com o sistema alvo. Em vez disso, uma fonte externa produz algum fenômeno físico, tal como radiação por *heavyion* e/ou interferência eletromagnética, causando correntes espúrias dentro do CI alvo.

6.3.1 Injeção de Falhas por Hardware Com Contato

Injeção da falha por hardware usando o contato direto com os pinos do circuito, chamados frequentemente injeção a nível de pino, é provavelmente o método mais comum de implementação de hardware de injeção de falha (HSUEH - 1997). Há duas técnicas principais para alterarem correntes e tensões elétricas nos pinos:

1. **Pontas de prova ativas (*active probes*):** Esta técnica adiciona a corrente através de pontas de prova junto aos pinos, alterando suas correntes elétricas. O método da ponta de prova é geralmente limitado a falhas de *stuck-at*, embora seja possível realizar falhas de *bridge* colocando uma ponta de prova através de dois ou mais pinos. Um cuidado adicional deve ser tomado ao usarem-se pontas de prova ativas para forçar a corrente adicional no dispositivo alvo, pois uma quantidade exagerada de corrente pode danificar o dispositivo alvo;
2. **Inserção de soquete (*socket insertion*):** Esta técnica introduz um soquete entre o hardware alvo e o circuito injetor. O soquete introduzido injeta *stuck-at*, *open*, ou lógicas mais complexas no hardware alvo, forçando os sinais analógicos que representam valores desejados da lógica nos pinos do hardware alvo. Os pinos de sinais podem ser invertidos, ANDed, ou ORed com sinais de pinos adjacentes mesmo com sinais precedentes no mesmo pino.

Ambos os métodos acima descritos, fornecem uma boa controlabilidade de tempos e de posições da falha com quase nenhuma perturbação ao sistema alvo. Nota-se porque as falhas são modeladas ao nível de pino, não são idênticas aos tradicionais modelos de falha *stuck-at* e *bridge* que geralmente ocorrem dentro do mesmo CI. Apesar de tudo, pode-se conseguir muitos dos mesmos efeitos, como a utilização de circuitos de detecção de falhas, usando estes métodos de injeção. Pontas de prova ativas acopladas ao hardware da fonte de alimentação, injeta falhas de distúrbio da fonte de alimentação. Entretanto, isto pode danificar o dispositivo injetado ou aumentar o risco da injeção destrutiva.

6.3.2 Injeção de Falhas por Hardware Sem Contato

Estas falhas são injetadas criando radiações *heavyion* (HSUEH - 1997). Um íon passa através da região de depleção do dispositivo alvo gerando uma corrente transitória. Colocando o hardware alvo dentro ou próximo de um campo eletromagnético também injeta falhas. Os engenheiros em geral gostam destes métodos porque imitam fenômenos físicos naturais. Entretanto, é difícil provocar exatamente o tempo e a posição de uma injeção de falha usando esta técnica porque não se pode precisamente controlar o exato momento da emissão do *heavyion* ou do campo eletromagnético.

6.3.3 Plataformas de Teste e Injeção de Falhas por Hardware Consagradas na Literatura

6.3.3.1 MESSALINE

O Messaline (ARLAT, 1989) foi criado no LAAS-CNRS, em Toulouse, França, usa ambas as técnicas de pontas de prova e soquetes ativos para conduzir a injeção da falha a nível de pino. Na Figura ilustra a arquitetura geral do ambiente Messaline de testes e injeção de falhas. A plataforma Messaline pode injetar falha de *stuck-at*, *open*, *bridge* e outras lógicas mais complexas entre outras. Sinais coletados do sistema alvo podem fornecer uma realimentação para o injetor. Além disso, um dispositivo é associado a cada ponto de injeção para detectar quando cada falha é ativada para produzir um erro. Com essa ferramenta é possível também de se injetar até 32 pontos de injeção simultaneamente. Esta ferramenta foi usada nas experiências de um sistema integrando, empregado em um sistema computadorizado de controle de uma via férrea e em um sistema distribuído para o projeto Esprit Delta-4 (BARRETT, 1993).

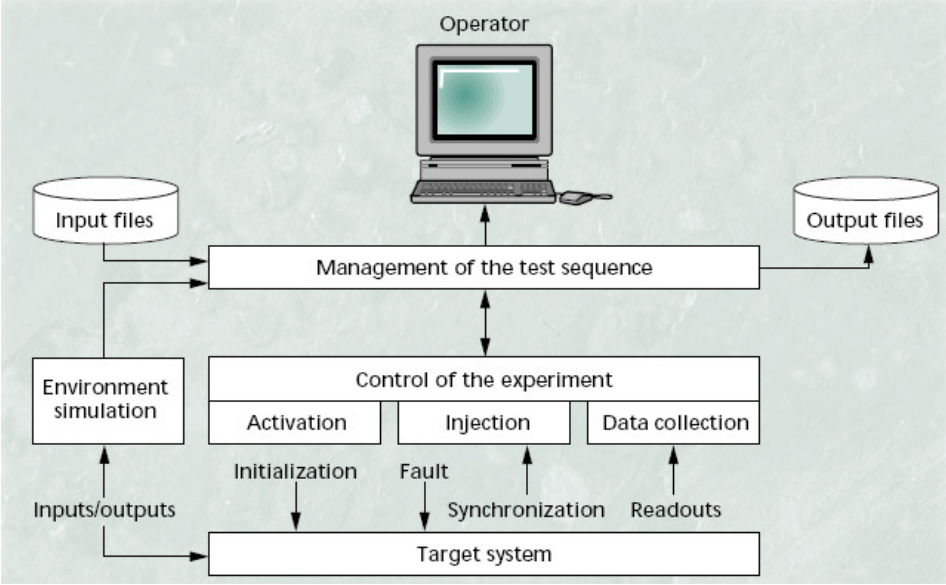


Figura 6.2 – Ambiente de testes Messaline (HSUEH - 1997).

6.3.3.2 FIST

O FIST (*Fault Injection System for Study of Transient Fault Effect*) (GUNNETLO, 1989) foi criado na Universidade de Tecnologia de Chalmers na Suécia, e empregam ambos os métodos de injeção, com contato e sem contato para criarem falhas transientes dentro do sistema alvo. Esta ferramenta usa a radiações *heavyion* para criar falhas transientes em posições aleatórias dentro do CI quando o mesmo é exposto à radiação e causam um ou múltiplos *bit-flips*.

A fonte de radiação é montada dentro de uma câmara a vácuo juntamente com um sistema computadorizado bi-processado. O computador é posicionado de modo que um do processador é exposto diretamente sob a radiação. O outro processador é usado como uma referência detectando se a radiação resulta em alguns *bit-flips*. A Figura 6.3 ilustra o ambiente de testes FIST.

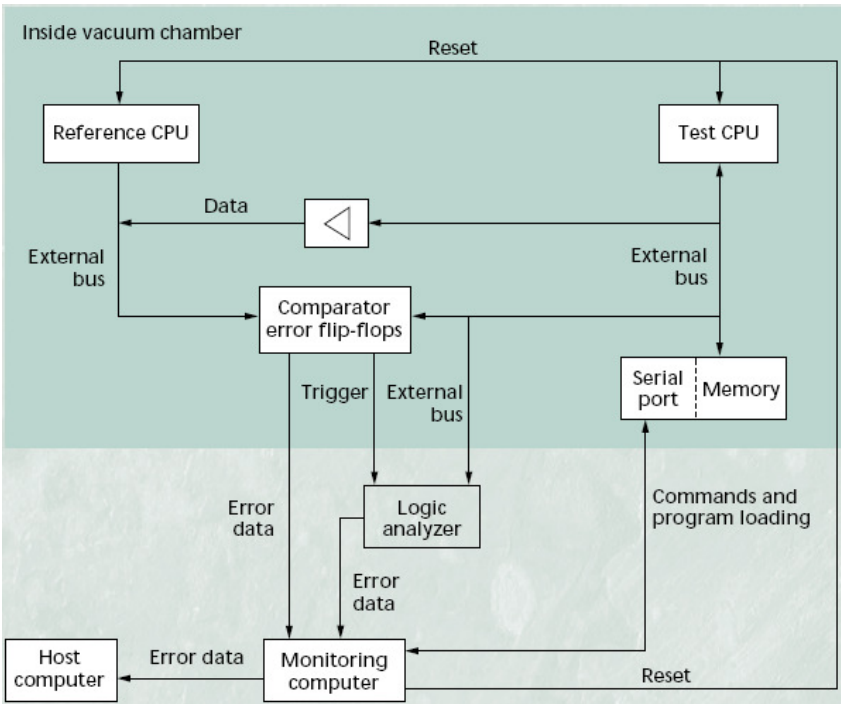


Figura 6.3 – Ambiente de testes FIST (HSUEH - 1997).

O FIST pode injetar falha diretamente dentro de um CI naqueles em que não se possa injetar diretamente no pino do CI. Podem-se produzir falhas transientes em posições aleatórias igualmente pelo CI, que leva a uma grande variação nos erros vistos na saída dos pinos. Além da radiação o FIST permite a injeção de falhas e distúrbios na alimentação do sistema alvo.

Isto é feito colocando um transistor MOS entre a fonte de alimentação e o pino de Vcc do CI processador para controlar a amplitude da queda de tensão (*voltage drop*).

Os distúrbios na fonte de alimentação geralmente afetam posições múltiplas dentro de um CI e podem causar falhas de propagação de atraso nos *gates* dos transistores. Os resultados experimentais mostram que os erros resultantes de ambos os métodos, causam efeitos similares que causam erros de dados e de fluxo de programa. Entretanto, as radiações *heavyion* podem causar na maior parte erros no barramento de endereço, em quanto os distúrbios na fonte de alimentação afetam na maior parte os sinais de controle.

6.3.3.3 MARS

O MARS (*Maintainable Real-Time System*) (KARLSSON, 1995) é uma arquitetura tolerante a falhas distribuída, foi criada na *Technical University of Vienna* na Áustria. Além de usar radiações *heavyion* como é usado no FIST, o MARS usa campos eletromagnéticos usos para conduzir a injeção da falha sem contato: Uma placa de circuito colocada entre duas placas de metal carregadas ou um CI colocado perto de uma ponta de prova, causa uma injeção de falha.

Fios que agem como antenas e colocados nos pinos individuais do CI, acentuam o efeito do campo eletromagnético naqueles pinos. Pesquisadores compararam estes três métodos (radiações *heavyion*, injeção a nível de pino, e interferência eletromagnética), em termos de sua potencialidade para exercitar os mecanismos de detecção de erro do MARS. Os resultados mostraram que os três métodos são complementares e geram diferentes tipos de erros. As injeções a nível de pino fazem com que os mecanismos de detecção de erro fora do processador central sejam exercitados mais eficazmente do que as radiações *heavyion* ou a interferência eletromagnética. Os últimos dois métodos eram melhores para testar software e aplicação como mecanismos da detecção de erro.

6.4 Teste e Injeção de Falhas por Software

Recentemente, pesquisadores tem tido maior interesse em desenvolver ferramentas de testes e injeção de falhas por software. Este tipo de técnica é um atrativo, porque não requer hardware com altos custos (HSUEH - 1997). Além disso, podem ser usados em aplicações e sistemas operacionais, o que é difícil de se fazer por injeção de falhas por hardware.

Se o alvo for uma aplicação, o injetor de falhas está introduzido na própria aplicação ou inserido entre a aplicação e o sistema operacional. Se o alvo for o sistema operacional, o injetor de falhas deve ser introduzido dentro dele, porque é muito difícil adicionar uma camada entre a máquina e o sistema operacional.

Embora a aproximação por software seja flexível, tem suas desvantagens, tais como:

- Não se pode injetar falhas nas posições que são inacessíveis ao software;
- O software de instrumentação pode perturbar a carga de trabalho que funciona no sistema além de mudar a estrutura do software original. Um cuidado no projeto do ambiente de injeção pode minimizar a carga de trabalho do software;

- A baixa resolução de tempo desta abordagem pode causar problemas de exatidão. Para falhas de latência prolongada, tais como falhas de memória, a baixa resolução de tempo não deve ser um problema. Para falhas de latência mais baixas, tais como falhas de barramento e de processado, esta abordagem pode falhar e não capturar determinados comportamentos de erro, como propagação, por exemplo. Pode-se resolver este problema fazendo uma aproximação híbrida, que combina a versatilidade da injeção de falha por software e a exatidão da monitoração por hardware.

Podemos dividir os métodos da injeção por software sobre quando as falhas serão injetadas: em tempo de compilação (*compilation time*) ou em tempo a execução (*run time*).

6.4.1 Injeção de Falhas em Tempo de Compilação

Para conseguir injetar falha em tempo de compilação, as instruções do programa devem ser modificadas antes da imagem ser carregada e executada (HSUEH - 1997). Melhor que injetar falha no hardware do sistema alvo, este método injeta erros no código fonte ou no código *assembler* do programa alvo para emular o efeito de hardware, software e de falhas transientes. O código modificado altera as instruções do programa alvo, causando a injeção. A injeção gera uma imagem errônea do software, e quando o sistema executa a imagem da falha, ativa a falha.

Este método requer a modificação do programa que avaliará o efeito da falha, e não requer qualquer software adicional durante a execução. Além disso, ele não causa nenhuma perturbação ao sistema alvo durante execução. Porque o efeito da falha é *hard-coded* (codificada em hardware). Pode-se usá-lo para emularem falhas permanentes. A implementação deste método é muito simples, mas não permite a injeção das falhas enquanto o programa funciona.

6.4.2 Injeção de Falhas em Tempo de Execução

Durante a execução, um mecanismo é necessário para provocar a injeção da falha. Comumente utilizam-se os seguintes mecanismos (HSUEH - 1997):

- ***Time-out*:** Nesta técnica, uma das mais simples, um temporizador expira em um tempo predeterminado, provocando a injeção. Especificamente, o evento do

intervalo de parada (*time-out*) gera uma interrupção para invocar a função que injeção da falha. O temporizador pode ser por hardware ou por software. Este método não requer nenhuma modificação no programa da aplicação ou a carga de trabalho (*workload*). Um temporizador por hardware deve ser ligado ao vetor de interrupção do sistema (*interrupt handler vector*). Por causa da injeção de falhas na base do tempo, melhor que eventos específicos ou o estado do sistema, este produz efeitos de falha e comportamentos do programa imprevisíveis. É, entretanto, apropriado para emularem falhas transientes e falhas periódicas de hardware.

- ***Exception/trap*:** Neste caso, em uma exceção de hardware ou uma falha por software, transfere-se o controle ao injetor de falhas. Ao contrário do *time-out*, e *exception/trap* podem injetar a falha sempre que determinados eventos e/ou circunstâncias ocorrem. Por exemplo, em uma falha injetada por software, uma instrução é introduzida no programa alvo que invocará a injeção da falha antes que o programa execute uma instrução particular. Quando é executada a injeção, uma interrupção é gerada que transfere o controle ao tratamento de interrupção (*interrupt handler*). Uma exceção de hardware invoca a injeção quando um evento observado pelo hardware (*hardware-observed*) ocorre (quando uma posição de memória em específico é acessada, por exemplo). Ambos os mecanismos devem ser ligados ao vetor de interrupção.
- ***Code insertion:*** Nesta técnica, as instruções são adicionadas ao programa alvo para permitirem que a injeção da falha ocorra antes das instruções particulares, melhor que o método de modificação de código. Ao contrário da modificação de código, a inserção do código executa a injeção da falha durante a execução e adiciona instruções, sendo assim melhor que modificar as instruções originais. Ao contrário do método *exception/trap* em que o injetor de falhas pode existir como parte do programa alvo e funcionar na modalidade usuário (*user*) melhor que na modalidade sistema (*root*).

6.4.3 Plataformas de Teste e Injeção de Falhas por Software Consagradas na Literatura

6.4.3.1 FERRARI

Ferrari (*Fault and Error Automatic Real-Time Injection*) (KANAWATI, 1992) desenvolvido pela Universidade do Texas em Austin nos Estados Unidos, usa injeções

por software para injetar falhas no processador, na memória e falhas no barramento. A plataforma Ferrari consiste em quatro componentes: o inicializador/ativador, a informação do usuário, o injetor de erros/falhas, e o coletor/analisador de dados.

O injetor de erros e falhas usa a injeção por rotinas de software. As injeções por software são provocadas ou pelo *program counter* quando desejado determinadas posições do programa ou por um temporizador (*timer*). Quando as injeções são provocadas, as rotinas de tratamento de injeções de falhas injetam as falhas em lugares específicos, tipicamente mudando o conteúdo dos registradores ou posições de memória selecionadas para emular corrupções reais dos dados. As falhas injetadas podem ser falhas permanentes ou transientes que resultam em um erro nas linhas de endereço ou dados e um erro de condição (fluxo de controle).

6.4.3.2 FTAPE

O FTAPE (The *Fault Tolerance and Performance Evaluator*) (TSAI, 1996), desenvolvido pela Universidade de Illinois nos Estados Unidos, consiste nos componentes mostrados na Figura 6.4. Podem-se injetar falhas em registradores acessíveis do usuário nos módulos processador, nas posições de memória, e no subsistema de armazenamento (disco).

As falhas são injetadas como *bit-flips* para emular erros que em conseqüência resultam falhas. As falhas de disco são injetadas executando uma rotina no código do *driver* que emula erros do I/O (erros de barramento e erros de temporização, por exemplo). Nenhum hardware ou modificação adicional do código da aplicação são necessários para este método.

Um gerador artificial de carga de trabalho cria as tarefas que contem quantias especifica do processador, da memória, e atividade de I/O, e as falhas que são injetadas com uma estratégia que considera as características das tarefas de trabalho no tempo da injeção.

O FTAPE foi usado em diversos computadores e servidores Tandem tolerante a falhas como a base de um *benchmark* para tolerância a falha, que mede a ocorrência de falhas de sistema e o quanto a degradação do desempenho sob condições de falha.

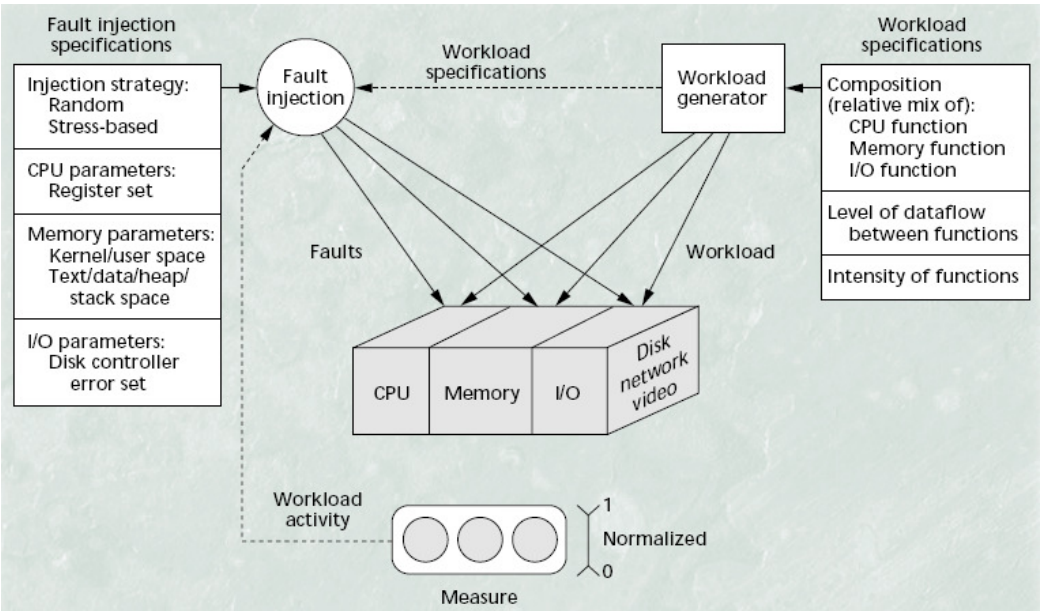


Figura 6.4 – Plataforma de Testes FTAPE (HSUEH - 1997).

6.4.3.3 DOCTOR

O DOCTOR (*Integrated Software Fault Injection Environment*) (HAN, 1995), desenvolvido na Universidade do Michigan nos Estados Unidos, permite a injeção de falhas no processador, na memória, e falhas de comunicação em rede. São usados três métodos de funcionamento, *time-out*, injeção direta, e a modificação de código para disparar a injeção da falha. Uma vez que o *time-out* ocorre, a injeção da falha ocorre e sobrescreve o conteúdo da memória, para emular a ocorrência de uma falha de memória. Para as falhas transitórias do processador, injeções por software provocam a injeção da falha. Para falhas permanentes do processador, a injeção da falha é feita mudando instruções do programa durante a compilação para emular a corrupção de instruções e de dados devido às falhas.

O DOCTOR foi usado no Harts (*Highly Available Real Time System*) um sistema *real-time* distribuído, para investigar o efeito de perdas intermitentes de mensagens entre dois nós adjacentes e o efeito do roteamento usando dados de falha. Os pesquisadores usaram resultados experimentais para validar um modelo da entrega de mensagens e para avaliar diferentes métodos de entrega destas mensagens.

6.4.3.4 XCEPTION

O XCEPTION (*Software Fault Injection and Monitoring in Processor Functional Units*) (CARREIRA, 1995) foi desenvolvido pela Universidade de Coimbra

em Portugal, tem a vantagem de ter características avançadas de depuração (*debugging*) e uma ótimo desempenho de monitoração, presente em muitos processadores modernos atuais, para injetar falhas mais realistas. Não requer nenhuma modificação no software da aplicação e nenhuma inserção de *traps* de software. Xception usa integrado ao processador um hardware disparador que provocar a injeção da falha. O injetor de falhas é implementado como um processo que trata a exceção no qual requer a modificação dos vetores de interrupção. As falhas geradas pelo Xception são provocadas pelo acesso à endereços específicos, assim as experiências podem ser reproduzidas novamente. Os seguintes eventos podem provocar a injeção da falha:

- Busca de *opcode* de um endereço especificado;
- Carga do operando de um endereço especificado;
- Armazenamento do operando a um endereço especificado;
- Um atraso específico desde o boot do sistema;
- Ou uma combinação dos eventos acima.

Cada falha tem uma máscara de falhas específica: uma série de bits determina qual e onde os bits correspondentes serão injetados no sistema alvo. Os bits da máscara de falha que estão em nível lógico 1 podem ser usados em diversas operações a nível de bit, tais como: *stuck-at-zero*, *stuck-at-one*, *bit-flip*, e *bridge*.

6.5 Injeção de Falhas por Hardware versus Software

A Tabela 6.2 classifica os métodos de injeções de falhas por hardware e por software:

	HARDWARE		SOFTWARE	
	Com Contato	**Sem contato**	**Compilação**	**Execução**
Custo	Alto	Alto	Baixo	Baixo
Perturbação	-	-	Baixo	Alto
Risco de danificação	Alto	Baixo	-	-
Tempo de monitoração	Alto	Alto	Alto	Baixo
Acessibilidade dos pontos de injeção	Pino do chip	Interno ao Chip	Registrador de memória por software	Registrador de memória por controlador de I/O
Controlabilidade	Alto	Baixo	Alto	Alto
Controle de disparo	Sim	Não	Sim	Sim
Repetibilidade	Alto	Baixo	Alto	Alto

Tabela 6.2 – Diferenças entre injeção de falhas por hardware e por software (HSUEH, 1997).

Os contrastes entre os métodos de injeção por hardware e por software encontram-se principalmente nos pontos de injeção de falha que se pode acessar, custo, e o nível de perturbação. Os métodos de injeção por hardware podem injetar falhas nos pinos do CI e em componentes internos, tais como os circuitos combinacionais e registradores que não são endereçáveis por software. Por outro lado, os métodos de injeção por software são convenientes para produzir mudanças diretamente no estado do software (memórias e registradores, por exemplo).

Dessa forma, usam-se métodos de hardware para avaliar a detecção de erros de baixo nível, já pelos métodos em software usa-se para a detecção de erros em um nível mais elevado do teste. Os métodos por software têm os custos menores, mas incorrem também em perturbações de *overhead* mais elevadas porque executam o software no próprio sistema alvo.

6.6 Plataformas Comerciais de Teste e Projetos de SoCs

A seguir serão apresentadas algumas das plataformas mais utilizadas em projetos de SoCs no mercado acadêmico e industrial.

6.6.1 Plataformas da Altera

Kit de desenvolvimento Nios-II Cyclone II:

Este kit de desenvolvimento de baixo custo contém uma FPGA Cyclone® II que é ideal para avaliar o alto desempenho e o baixo consumo de dispositivos Altera que possuem tecnologia 90 nanômetros que é mostrado na Figura 6.5, além de avaliar a família de processadores embarcados da Altera, o Nios-II.

O kit desenvolvimento Cyclone II FPGA possui as seguintes características:

- FPGA Cyclone II EP2C20F484C7N de 20k elementos lógicos;
- USB-Blaster™ cabo de download e debug download cable ;
- Memória Serial EPCS4 para configuração;
- Memória SDRAM de 8-Mbytes;
- Memória SRAM de 512-Kbytes;
- Memória flash de 1 a 4-Mbytes
- Codec de Audio 24-bits coder/decoder;
- Botões e push buttons diversos;
- Diaplays de 7-segmentos;
- LEDs para indicações;
- Comunicação serial padrão RS-232;
- Portas de controle de vídeo VGA (Video Graphics Array) e PS/2 (*Programming System* 2);
- Conectores de expansão;
- Socket para memórias SD (Secure Digital)/MMC(Multi Media Card).

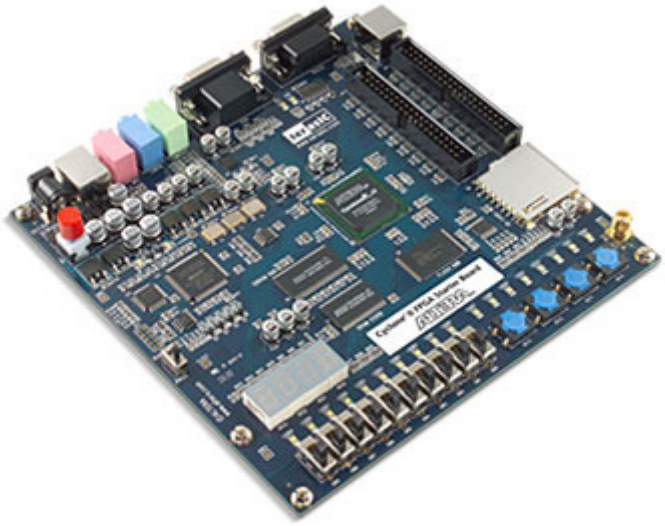


Figura 6.5 - Cyclone II FPGA Starter Development Kit.

Kit de desenvolvimento Nios II, edição Stratix:

O kit de desenvolvimento Nios® II da Altera, edição Stratix® mostrado na Figura 6.6 fornece tudo o que é necessário para o desenvolvimento em *system-on-a-programmable-chip* (SOPC). Baseado na família Nios II de processadores embarcados da Altera e no dispositivo Stratix EP1S10, este kit de desenvolvimento fornece um ambiente ideal para desenvolvimento e prototipação em uma larga escala de aplicações embarcadas.

O kit desenvolvimento *Nios II Development Kit, Stratix Edition* possui as seguintes características:

- Um ano de licença para software de desenvolvimento Quartus® II incluindo o SOPC *Builder* e ambiente de desenvolvimento para o Nios-II;
- FPGA Stratix EP1S10F780 de 10k elementos lógicos;
- MAX® EPM7128AE CPLD para controle de configuração lógica;
- Memória SRAM (1 Mbyte em dois bancos de 512 Kbytes e largura de 16-bits)
- Memória SDR SDRAM (16 Mbytes, 32-bit de largura)
- Memória Flash (8 Mbytes)
- Conector para cartões de memória CompactFlash Tipo I ;
- 10/100 Ethernet (PHY/MAC);
- 2 conectores comunicação serial padrão RS-232;
- Conectores de expansão;
- Botões e push buttons diversos;
- Diaplays de 7-segmentos;
- Display de LCD;
- LEDs para indicações;
- USB-BlasterTM cabo de download.

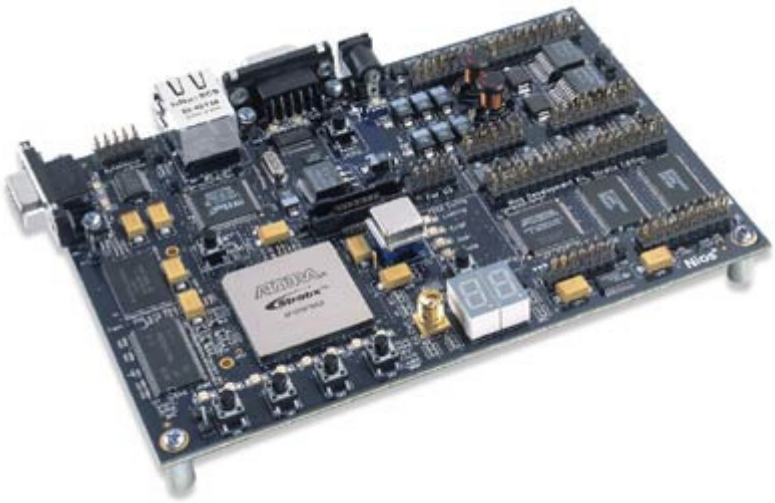


Figura 6.6 - Nios II Development Kit, Stratix Edition.

6.6.2 Plataformas da Xilinx/Digilent

Kit de Desenvolvimento XUPV2P-PRO:

O sistema de desenvolvimento Virtex-II PRO (XUPV2P – *Xilinx University Program Virtex II PRO*) mostrado na Figura 6.7 pode ser usado virtualmente em todos os níveis de projeto bem com para ensino nos currículos de engenharia, cursos técnicos e de projetos de pesquisa avançada. Baseado na FPGA Virtex-II Pro, a placa pode ser usada para treinamento em projetos para sistemas digitais, desenvolvimento de microprocessadores embarcados, tais como MicroBlaze e PowerPC 405. É poderoso o bastante suportar projetos de pesquisa avançados, porém de custo baixo o bastante para ser colocado em cada estação de trabalho. Os conectores da expansão podem acomodar circuitos e sistemas com propósito específico.

O Kit de desenvolvimento XUPV2P-PRO possui as seguintes características:

- FPGA Virtex-2 Pro XC2VP30 de 30,816 células lógicas e 1,5 milhões de *gates*, 136 multiplicadores internos de 18-bits, 2,448Kb de *block* RAM interna, e 2 processadores PowerPC 405;
- *Slot* para memória DDR (*Double Data Rate*) SDRAM DIMM (*dual in-line memory module*) que pode aceitar até 2Gbytes de memória RAM;
- Possui uma porta 10/100 Ethernet;
- Possui uma porta USB2.0;

- Conector para cartão de memória Compact Flash;
- Conector VGA (*Video Graphics Array*) para vídeo;
- Codec de áudio;
- 3 conectores para comunicação SATA (*Serial Advanced Technology Attachment*), 1 conector PS/2 e um conector para comunicação serial padrão RS-232;
- Conectores de expansão.

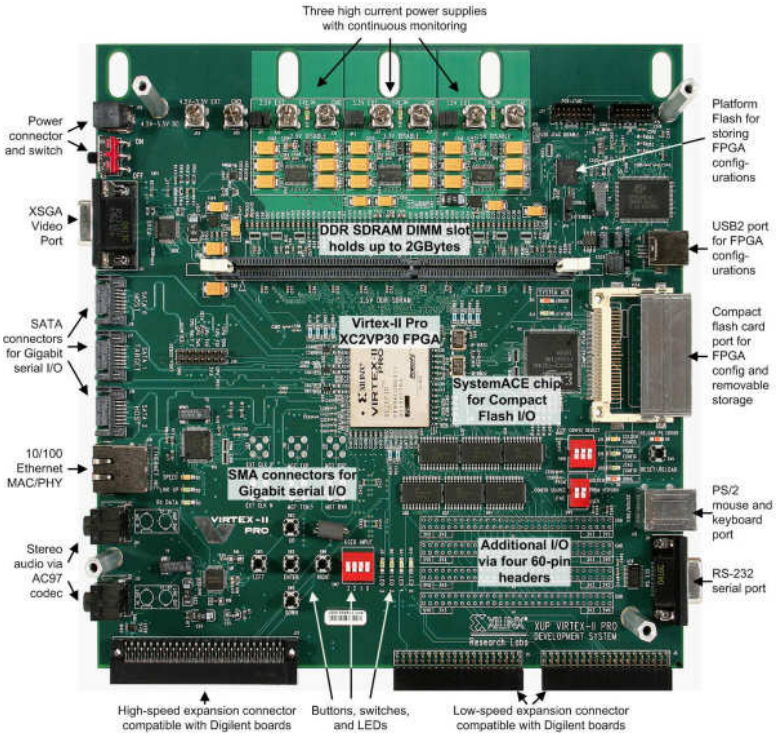


Figura 6.7 – Kit de desenvolvimento XUPV2P-PRO.

Kit de Desenvolvimento Spartan 3 Starter Kit:

Este kit fornece uma poderosa plataforma para desenvolvimento de projetos que utilizam FPGAs da família Spartan-3 da Xilinx conforme mostrado na Figura 6.8. Possui 200k *gates*, dispositivos de I/O *on-board* e 1MByte de memória SRAM de rápido acesso, fazendo desta, uma plataforma perfeita e de baixo custo para prototipação de novos projetos , desde circuitos com lógica simples até processadores embarcados, como MicroBlaze por exemplo.

O kit de desenvolvimento Spartan-3 Starter Kit possui as seguintes características:

- FPGA Spartan-3 XC3S200 com 12 multiplicadores internos de 18-bits, 216Kbits de *block* RAM interna, e velocidades de clocks internos de até 500MHz;
- Possui uma memória flash serial de 2Mbit (XCF02S);
- 8 chaves, 4 botões, 9 LEDs, e 4-digitos de displays de 7 segmentos;
- Comunicação serial padrão RS-232;
- Conector VGA para vídeo;
- Conectores PS/2 para mouse/teclado;
- Conectores de expansão;
- Possui memória SRAM de 1Mbyte e 10ns de velocidade de acesso.

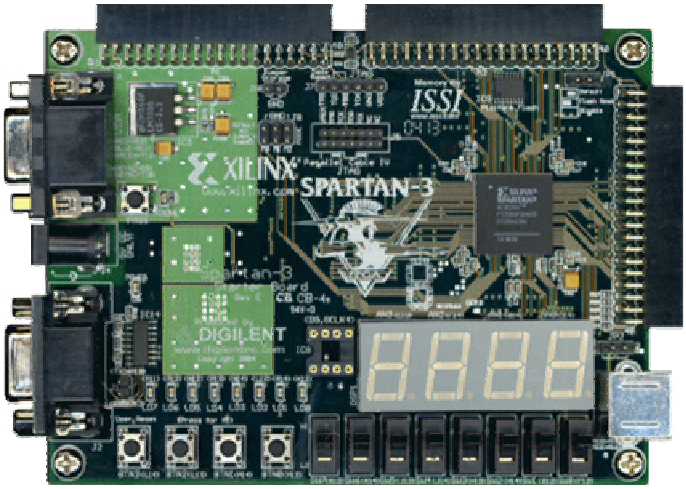


Figura 6.8 – Kit de desenvolvimento Spartan-3 Starter Kit.

PARTE II. METODOLOGIA

7. PLATAFORMA PARA DESENVOLVIMENTO DE SoCs PARA EMC

7.1 Estudos e Ensaios Preliminares Utilizando como um Estudo de Caso a Plataforma Xilinx/Digilent Spartan-3 Starter Kit

Após algumas pesquisas e avaliações referentes a plataformas de desenvolvimento, optamos pela a aquisição do kit ou plataforma de desenvolvimento *Spartan-3 Starter Kit* devido aos recursos financeiros disponíveis na época e porque esta plataforma atenderia inicialmente nossos propósitos do grupo SiSC (Laboratório de Sistemas, Sinais e Computação) que era o desenvolvimento de projetos de sistemas embarcados utilizando SoCs e de prototipação em hardware para fins de pesquisa. Com isso, para os primeiros testes e análises foi utilizada esta plataforma onde nos possibilitou a aprender e entender o funcionamento das ferramentas da Xilinx bem como seu processador MicroBlaze onde podemos avaliar o funcionamento de aplicações embarcadas para o mesmo. Após a etapa de aprendizado, iniciamos a implementação de técnicas de detecção de falhas compiladas para o processador MicroBlaze. Para os primeiros testes, foram implementadas três técnicas de detecção de falhas ilustradas no diagrama da arquitetura do hardware de testes na Figura 7.1 e explanadas a seguir:

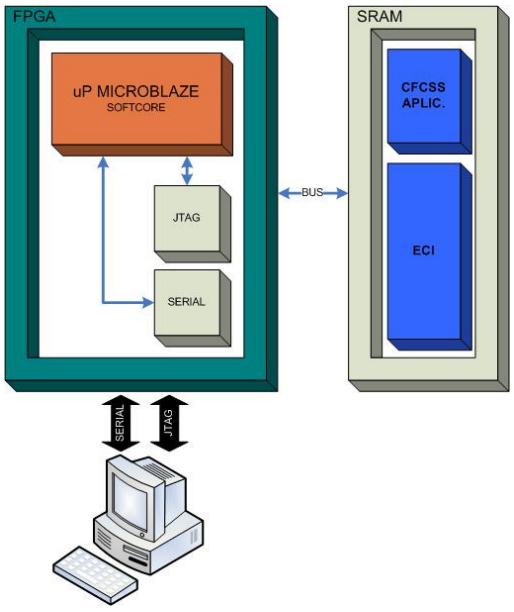


Figura 7.1 – Diagrama do hardware utilizado para os testes.

A três técnicas utilizadas nestes testes, sendo as duas primeiras de detecção de falhas e a terceira de detecção e recuperação, são apresentadas abaixo:

A técnica proposta por McCluskey (MCCLUSKEY, 2002), de CFCSS (*Control-Flow Checking by Software Signatures*) embutida em nossas aplicações usadas para testes (*benchmarks*) tais como multiplicação de matrizes, filtro FIR (*Finite Impulse Response*), *bubble sort* (ordenação de vetores) e algoritmo gerador de números primos (Crivo de Erastóstenes). Este algoritmo é baseado na identificação de blocos básicos no código da aplicação e na adição de dois tipos de assinaturas a estes blocos: O primeiro é adicionado em tempo de compilação, em quanto que o segundo é calculado em tempo de execução (*runtime*). O estudo consiste basicamente em executar instruções da comparação entre as duas assinaturas cada vez que o processador entra em um bloco básico. Sendo assim, as falhas que afetam o fluxo de controle são detectadas fazendo-se a comparação das duas assinaturas: a armazenada (tempo de compilação) e a calculada (tempo de execução) no qual tem que ser iguais, se não houve erro no fluxo de controle do sistema:

1) A técnica de detecção de falhas proposta por Miremadi (MIREMADI, 1995) usada foi a ECI (*Error Capturing Instructions*). Foi utilizado para cobrir o restante da memória de programa onde não havia código algum. Caso ocorra falha, e a técnica de CFCSS não consiga detectar (pois ela abrange apenas áreas de memória onde exista código), a técnica de ECI o faz, pois caso o PC (*program counter*) do microprocessador salte para um endereço onde não exista código essa técnica o fará reportando ao programa monitor externo que o PC do microprocessador caiu em um endereço de memória não válido;

2) A técnica proposta por Bezerra (BEZERRA, 2001) *Signature Analysis-Driven Refresh* (SADR), emprega um *Linear Feedback Shift Reg*ister (LFSR) que opera em duas modalidades diferentes: como um *Pseudo-Random Pattern Generator* (PRPG) para determinar o tempo médio entre duas operações de *readback* do *bitstream* e da configuração do FPGA e como um *Parallel Signature Generator* (PSG) com a finalidade de compactar os bits de configuração do FPGA sob monitoração. O método da análise da assinatura é usado para determinar se o *bitstream* da configuração do FPGA necessita *refresh*. Cada um dos FPGA's executa dois processos:

a) PRPG/PSG: Esse processo opera de duas maneiras diferentes: i) no modo *pseudo-random pattern generator* quando o LFSR alcança um dado valor, esse processo sinaliza o momento em que deve-se iniciar a operação de *readback*; ii) no modo *parallel signature generator*, para compactar o *bitstream* do FPGA monitorado em um única assinatura.

b) RDB&R: O processo de *readback* e *refresh* é responsável pela execução do *readback* e do *refresh* do *bitstream* do FPGA.

Conforme o diagrama da Figura 7.1, a descrição completa dos blocos segue abaixo:

Bloco FPGA:

Neste bloco está contido o processador *SoftCore* MicroBlaze da Xilinx e uma porta JTAG para configuração e leitura (*readback*) do *bitstream* de configuração do dispositivo. Possui também uma porta de comunicação serial padrão RS232 para comunicação com o microcomputador externo que nesta configuração irá fazer o papel da FPGA2 como proposta na técnica SADR descrita anteriormente. Este computador possui um programa de monitoramento através da serial onde monitora periodicamente o processador MicroBlaze e a respectiva leitura do *bitstream* bem com sua configuração e reconfiguração caso necessário.

Bloco SRAM:

Na memória SRAM está contido o aplicativo a ser rodado pelo microprocessador (protegido pela técnica CFCSS). Já no restante de memória que não é utilizado, é coberto pela técnica de ECI. Esta técnica insere uma instrução de comunicação com o computador que faz uma comunicação via interface serial RS232 e avisa ao programa monitor residente no computador gerenciador que o PC (*program counter*) saltou para um endereço onde não há código a ser processado, ou seja, não válido.

Para validar e avaliar tais técnicas de detecção de falhas e avaliar o comportamento do dispositivo FPGA perante um ambiente de interferência eletromagnética, utilizou-se o laboratório do *Instituto Nacional de Tecnologia Industrial* (INTI), situado em Buenos Aires na Argentina para fazermos testes de interferência eletromagnética (EMI) irradiada e conduzida, onde submetemos nosso hardware juntamente com nossas técnicas de detecção de falhas à prova em um ambiente de EMI em uma Célula *GigaHertz Transverse Electromagnetic* (GTEM cell) Figura 7.2.

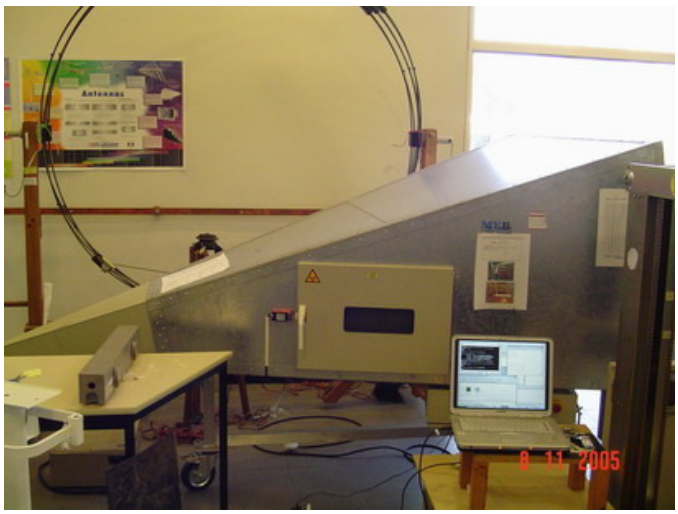


Figura 7.2 – Célula GTEM do laboratório INTI (Buenos Aires) utilizada nos testes de injeção de falhas.

Nossa estrutura de destes, mostrada na Figura 7.3, foi baseada na norma internacional IEC 62.132, já descrita anteriormente, com regras de incidência de campo EM (eletromagnético) igual a 10V/m (volts/metro) em uma faixa de freqüência de 150kHz a 1GHz com 80% de modulação de amplitude, no qual a freqüência foi incrementada (em cada passo do teste) de 5%. Em seguida o campo EM foi aumentado para 60V/m para submeter o circuito a um teste de stress e verificar seu comportamento.

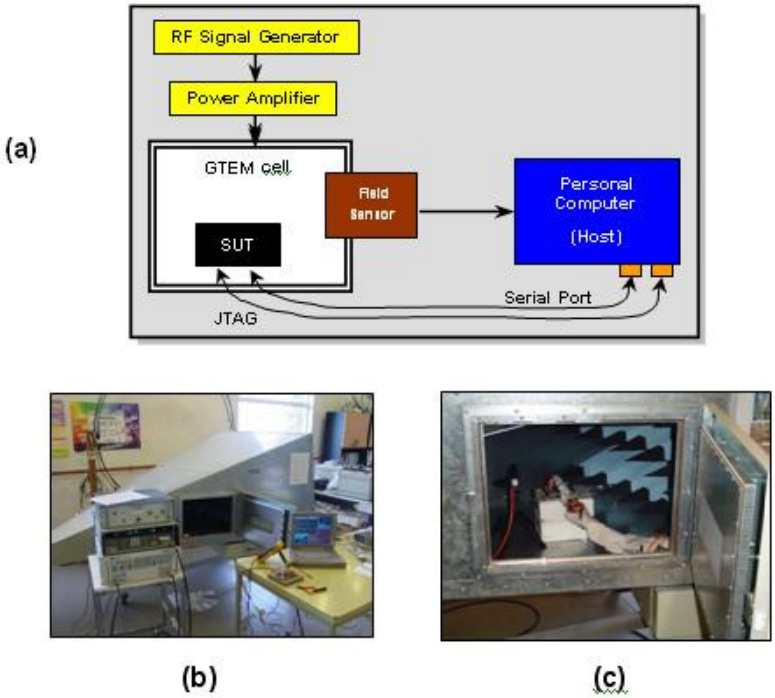


Figura 7.3 - Estrutura de teste de EMI. (a) e (b): Esquema geral e equipamentos; (c): Placa de circuito a ser testado dentro da GTEM cell (VARGAS, 2006).

Para isso realizamos dois tipos de teste com o kit de desenvolvimento da Xilinx:

a) Uma bateria de testes com a placa toda coberta com um filme de cobre e posteriormente aterrado, exceto o dispositivo FPGA exposto ao campo eletromagnético onde é ilustrado na Figura 7.4a e chamada de "*Open Window*";
b) Toda a placa do sistema ao campo eletromagnético, exceto o dispositivo FPGA que permaneceu sob uma camada de isolante de filme de cobre, que é ilustrado na Figura 7.4b e chamado de "*Cloded Window*".

(a)  (b) 

Figura 7.4 – Placas de teste. (a) Placa *Open Window*; (b) Placa *Closed Window*.

Nos testes utilizando a placa "*Open Window*" espera-se que a maioria das falhas injetadas ocorra apenas no FPGA; já nos testes com a placa "*Closed Window*", espera-se que a maioria das injeções de falhas seja conduzida para o FPGA através das linhas de alimentação, barramento de memória e pinos de i/o da placa.

De posse de todos os resultados dos testes efetuados, chegamos às seguintes conclusões:

Observou-se que em se tratando de falhas de configuração, ou seja, no *bitstream* do FPGA, ocorreu com maior freqüência na placa "*Closed Window*" e da forma conduzida como mostra a Tabela 7.1. A Tabela 7.1 mostra que as falhas de lógica (falhas de fluxo de controle) mostraram-se aproximadamente constante entre 35 e 38%.

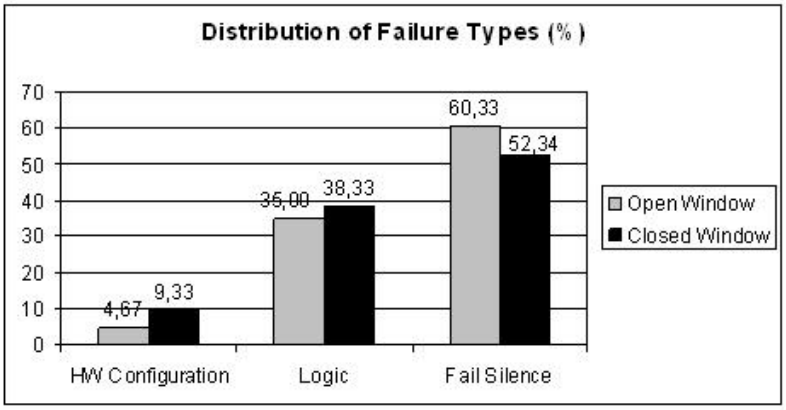


Tabela 7.1 - Distribuição de Falhas onde afetaram a FPGA (VARGAS, 2006).

A Tabela 7.2 resume a capacidade das três técnicas de detecção de falhas utilizadas até o momento quando embarcadas com um núcleo de IP (MicroBlaze) em um FPGA em funcionando em um ambiente de EMI : gerador de números primos (PNG), multiplicação de matrizes (MM) e filtro digital FIR (DF).

Application Code	Open Window			Closed Window		
	Configuration Errors (%)	Logic Errors (%)		Configuration Errors (%)	Logic Errors (%)	
	Detected by SADR	Detected by CFCSS	Detected by ECI	Detected by SADR	Detected by CFCSS	Detected by ECI
PNG	100	33.33	100	100	28.86	NA*
MM	100	12.50	NA*	100	33.33	NA*
DF	100	NA*	NA*	100	33.33	NA*
Average (%)	100	22.92	100	100	31.84	0.00

Tabela 7.2 - Sumário dos testes para as placas *Open* e *Closed Window* (VARGAS, 2006).

*** NA**: Não foram observadas falhas que afetassem o fluxo de controle da CPU.

A tabela acima foi construída a partir da extração de 642 medidas (214 medidas para cada um das aplicações utilizadas onde 107 utilizadas na a placa "*Closed Window*" e 107 medidas na placa "*Open Window*". Cada medida corresponde aproximadamente a 30 segundos com a CPU exposta a EMI.

Fazendo a interpretação da Tabela 7.2, temos as seguintes conclusões:

Na coluna "*Configuration Errors*", indica aquelas falhas que afetam o FPGA que é detectado pela técnica de *Signature Analysis-Driven Refresh* (SADR) (algumas destas falhas são de erros de fluxo de controle da CPU; outros não mudaram o fluxo de controle, mas resultaram de qualquer maneira em um erro de processamento de dados). Na maioria (~75%) das falhas são produzidas múltiplas inversões de bits (*bit-flips*) do *bitstream* de configuração do FPGA no qual resultaram numa paralisação do sistema (*system crash*) (isto é, nenhuma resposta ao estimulo externo, exceto ao sinal de comunicação e de restauração através do JTAG). As falhas restantes (25%) eram falhas únicas ou falhas múltiplas no *bitstream* de configuração que não induziram um ruído elétrico ao sistema. De qualquer maneira, o tipo de falha, pedido uma vez pelo microcomputador, a interface de JTAG podia sempre executar a operação do *readback* do FPGA ao microcomputador, onde o *bitstream* é comprimido e comparado à referência armazenada.

Na coluna "*Detected by* CFCSS" é relacionado aquelas falhas que resultaram em um erro de fluxo de controle do processador em qual parou de executar no espaço de memória alocado para o código da aplicação, pois o restante da memória não é coberto por essa técnica (estas falhas corrompem a lógica da configuração do FPGA, a lógica funcional, ou uma combinação destes elementos). Quando a técnica de CFCSS não pode assegurar a detecção das falhas que modificam a execução/controle de fluxo do processador e o *bitstream* de configuração do FPGA, a potencialidade da detecção desta técnica é afetada. As técnicas complementares, tais como SADR e ECI são necessárias para aumentar a detecção de falhas induzidas por EMI.

Na coluna "*Detected by* ECI" é relacionado aquelas falhas que resultaram em um erro de fluxo de controle do processador em qual parou de executar o código da

aplicação no espaço de memória físico não usado para o código ou os dados da aplicação. Mesmo que todas estas falhas fossem detectadas (100%), representaram somente 1,28% do número total das falhas ocorridas. Isto é explicado devido ao fato de que somente o espaço de memória não utilizado para o código e os dados da aplicação (1Mbyte disponível na placa) foi protegido com instruções de "ECI".

7.1.1 Conclusões dos Ensaios Preliminares

Este trabalho nos proporcionou fazer um levantamento sobre o comportamento de FPGAs do tipo SRAM assim com a plataforma de desenvolvimento de SoCs comercial da Xilinx operando em ambientes de exposição a EMI. Os resultados obtidos mostram que elementos da memória usados pela lógica funcional do FPGA são de quatro a cinco vezes mais sensíveis à EMI do que aqueles usados para a lógica da configuração do hardware.

Nas duas configurações, "*Closed Window*" e "*Open Window*" tiveram como principal mecanismo de injeção de falha o seguinte:

1. Na configuração "*Closed Window*", as falhas eram geradas devido ao efeito do acoplamento entre o campo EM e as trilhas e conectores da placa que eram então conduzidos para os pinos de entrada do FPGA. Estas falhas induzem principalmente erros no *bitstream* da configuração do FPGA (ou seja, falhas de hardware);

2. Na configuração "*Open Window*", as falhas induzidas pelo campo EM sobre o encapsulamento do FPGA era o mecanismo principal da injeção da falha para este componente. Estas falhas induzem principalmente erros na lógica da aplicação do FPGA (ou seja, falhas de software).

Finalmente, chegou-se à conclusão de que para poder avaliar com mais precisão o tipo de falha e onde exatamente ela ocorre, tem de se ter uma placa normalizada, ou seja, seguindo regras de projeto de alguma norma para podermos avaliar com maior precisão os efeitos da interferência eletromagnética tanto induzida quanto irradiada. Com isso tenta-se separar os efeitos da EMI sobre o CI daqueles sobre os demais componentes da placa, assim como da própria placa (como seus conectores, pinos, *switches*, trilhas etc.). Assim o conjunto de normas IEC 62.132 para a medição da imunidade de circuitos integrados a interferências eletromagnéticas tanto induzidas, quanto irradiadas, foi escolhido como base para a proposta deste trabalho onde será explanado nos próximos capítulos.

7.2 Apresentação da Plataforma

Após a análise feita nos testes e ensaios preliminares mostrados no capítulo anterior que avaliam o comportamento de um FPGA do tipo SRAM e de uma plataforma de teste e desenvolvimento de SoCs comercial, que não segue norma alguma para testes específicos de EMI, apresentamos a seguinte plataforma para testes de EMI:

A plataforma desenvolvida é baseada em duas placas específicas e complementares. A primeira é dedicada para teste de imunidade a ruídos irradiados em uma *Gigahertz Transverse Electromagnetic Cell* (GTEM *Cell*) ilustrado na Figura 7.5 e na Figura 7.6 de acordo com a norma IEC 62.132-2 (IEC, 2004). A segunda placa é dedicada ao teste conduzido de ruídos de RF e é de acordo com a norma IEC 62.132-4 e IEC 62.132-2 (IEC, 2004), no qual pode ser utilizada na GTEM Cell (Figura 7.5) também para testes de imunidade a interferência irradiada.

Ambas as placas contêm FPGAs da Xilinx. A PCB para testes de imunidade a RF irradiado, contém uma Spartan3 XC3S200 e 1Mbyte de SRAM, além da lógica necessária para comunicação e configuração. A PCB para testes de imunidade a RF conduzido contém duas FGPAs Spartan3E XC3S500E, um microcontrolador MSC1211 da Texas Instruments (*core* 8051), 16MBytes de SDRAM, e 8MBytes de memória Flash serial, entre outras lógicas requeridas para uma comunicação com o computador gerenciador que é o monitor dos testes. Com esta infra-estrutura, múltiplos microprocessadores embutidos podem ser prototipados, tais como MicroBlaze[5] da Xilinx (XILINX, 2005) e PowerPC 405 onde podem rodar os sistemas operacionais uCLinux e/ou uCOS-II[6]. Adicionalmente as partes de hardware, diversas implementações descritas em VHDL (***V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage*) descrevendo *cores* de propriedade intelectual (IP) e de programas em código C que podem também ter sua resposta à imunidade a ruídos de RF irradiado e/ou conduzido a fim de aumentar a dependabilidade para o projeto do SoC.

Como exemplo de utilização desta plataforma, é possível prototipar sistemas de redundância de hardware, sistemas de processamento paralelo e distribuído, dentre outras aplicações.

[5] MicroBlaze™ é um processador RISC de 32bits otimizado para o uso em arquiteturas das FPGAs da Xilinx. A interface de memória do processador é de acordo com a especificação do IBM *CoreConnect* para a o *On-Chip Peripheral Bus* (OPB).

[6] MicroC/OS-II foi certificado para RTCA DO-178B nível A, para o uso em sistemas aniônicos onde a ocorrência de uma falha poderia resultar na perda catastrófica do avião, e aprovado para o uso em dispositivos médicos FDA de classe III, onde uma falha poderia resultar na perda de vida do paciente.

Figura 7.5 – GTEM Cell (LOPES, 2005).

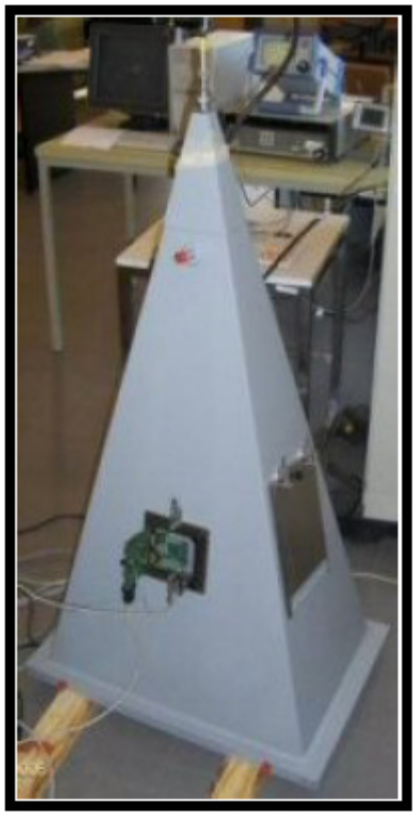


Figura 7.6 – GTEM-Cell utilizada para PCBs de teste com 10x10cm (PICCOLI, 2006).

7.3 Placa para Ensaios Irradiados

A placa para ensaios irradiados é *full compliant* (totalmente de acordo) com a norma IEC 62.312-2 mostrada na Figura 7.7 e na Figura 7.8 (o esquemático desta placa está incluso no ANEXO II).

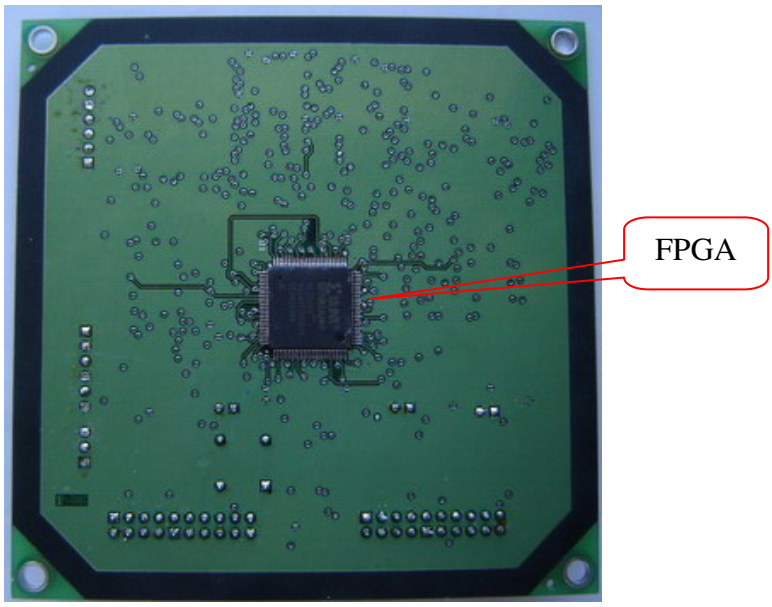


Figura 7.7 – Vista de cima (TOP) da placa de testes para ensaios irradiados.

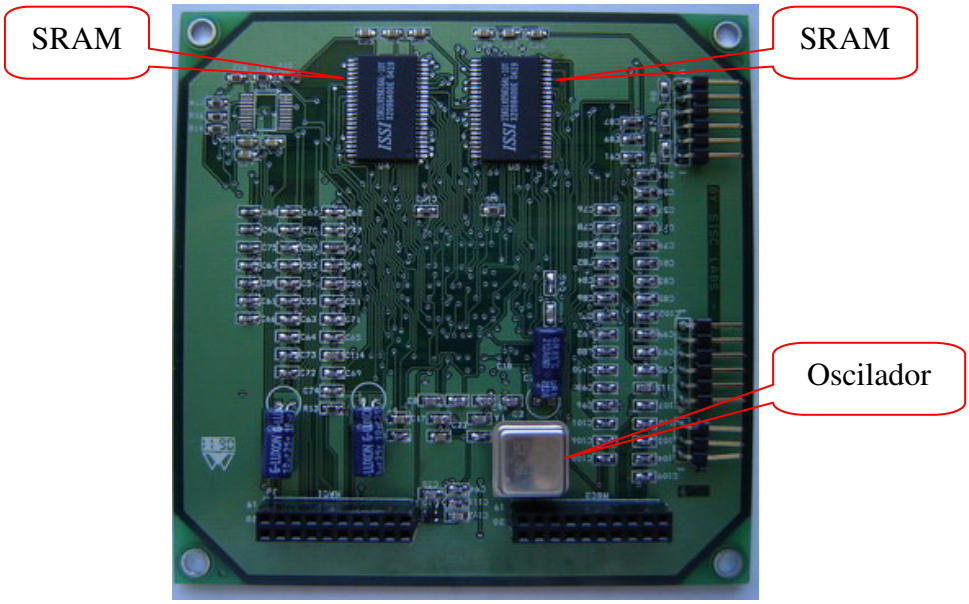


Figura 7.8 – Vista de baixo (BOTTON) placa de testes para ensaios irradiados.

Seguindo as regras de projeto descritas na norma IEC 62.132-1 (definições gerais) a placa possui as seguintes características:

1. A placa possui quatro *layers* (camadas) ilustrados na Figura 7.9 e distribuídos da seguinte forma:
    - *Layer* 1 (TOP) – Possui somente os componentes que serão irradiados com os ruídos de RF. Neste *layer* da placa somente haverá o plano de terra (GND) que cobre toda a área da placa, fazendo assim com que os outros sinais sejam colocados nos outros *layers* seguintes. Na borda da placa deve haver um anel

de cobre ou estanho ligado ao terra para contato com a porta da GTEM Cell que também está em contato com o terra da célula e fazer a união dos mesmos, para não haver diferenças de potencial entre a placa e a célula conforme ilustrado na Figura 7.6;

- *Layer* 2 (INNER 1) – Possui somente um plano de alimentação (VCC) ligado a alimentação principal da placa (3V3) e que cobre toda a área do *layer*;
- *Layer* 3 (INNER 2) - Possui todo o restante dos sinais da placa e do usuário que são utilizados para o funcionamento da plataforma;
- *Layer* 4 (BOTTON) – Possui todos os outros componentes da plataforma, tais como memórias, lógicas de comunicação, configuração e alimentação da placa. Neste *layer* é permitido o roteamento de todos os tipos de sinais que a placa utiliza, porém sendo necessário também adicionar um plano de terra.

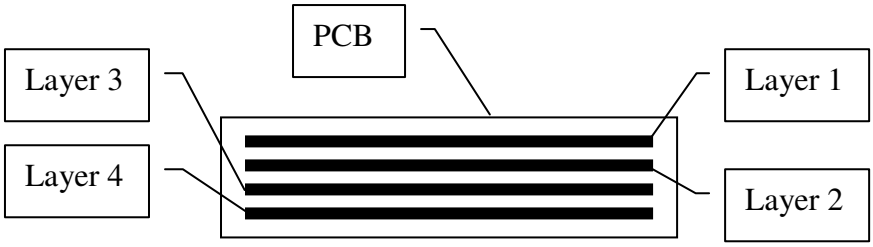


Figura 7.9 - Distribuição dos *layers* na placa.

2. A placa possui o tamanho de 10 x 10 centímetros, para o encaixe a uma porta de acoplamento específica contida na GTEM *Cell,* ilustrado na Figura 7.6 para união dos terras da célula com o da placa;

3. A placa possui os furos de passagem (vias) a uma distância das bordas de 5 milímetros para não correr o risco de algum sinal entrar em curto com a porta da célula;

4. Possui uma FPGA XC3S200 de 200k *gates* e de 144 pinos;

5. A placa contém duas memórias SRAM (*Static Random Access Memory*) IS61LV25616AL-10T do fabricante ISSI na configuração de 256x16 que formam um banco de memória de 1Mbyte;

6. Possui uma comunicação serial padrão RS232, para comunicação o microcomputador;

7. Possui uma comunicação padrão JTAG (*Joint Test Action Group*) para configuração e monitoração da FPGA;

8. Um oscilador de 49,152 MHz;

9. Possui 2 conectores para expansão.

7.4 Placa para Ensaios Conduzidos

De acordo com a norma IEC 62132-1, IEC 62132-2 e IEC 62132-4 para testes de suscetibilidade a EMI irradiado e conduzido em circuitos integrados foi projetado uma plataforma de testes que atenda tanto ensaios irradiados quanto conduzidos (conforme diagrama geral mostrado na Figura 7.11) , logo, seguindo as regras de projeto descritas nas normas a placa possui as seguintes características:

1. A placa possui quatro *layers* (camadas) ilustrados na Figura 7.10 e distribuídos da seguinte forma:
    - *Layer* 1 (TOP) – Possui somente os componentes que serão irradiados com os ruídos de RF. Neste *layer* da placa somente haverá o plano de terra (GND) que cobre toda a área da placa, fazendo assim com que os outros sinais sejam colocados nos outros layers seguintes;
    - *Layer* 2 (INNER 1) – Possui somente um plano de alimentação (VCC) ligado a alimentação principal da placa e que cobre toda a área do *layer*;
    - *Layer* 3 (INNER 2) - Possui todo o restante dos sinais da placa e do usuário que são utilizados para o funcionamento da plataforma;
    - *Layer* 4 (INNER 3) - Possui todo o restante dos sinais da placa e do usuário que são utilizados para o funcionamento da plataforma;
    - *Layer* 5 (INNER 4) - Possui todo o restante dos sinais da placa e do usuário que são utilizados para o funcionamento da plataforma;
    - *Layer* 6 (BOTTON) – Possui todos os outros componentes da plataforma, tais como memórias, lógicas de comunicação, configuração e alimentação da placa. Neste *layer* é permitido o roteamento de todos os tipos de sinais que a placa utiliza, porém sendo necessário também adicionar um plano de terra.

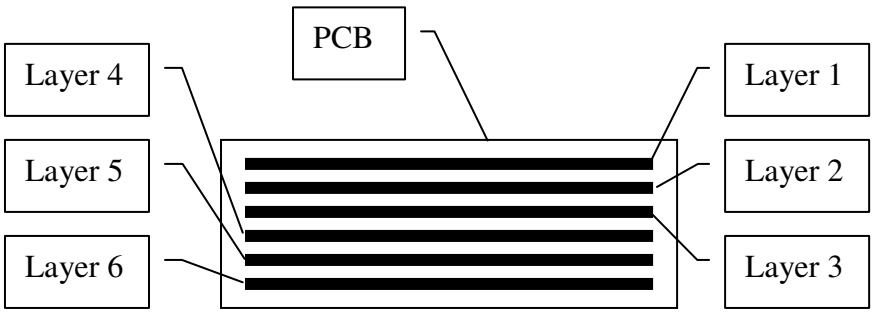


Figura 7.10 - Distribuição dos *layers* na placa.

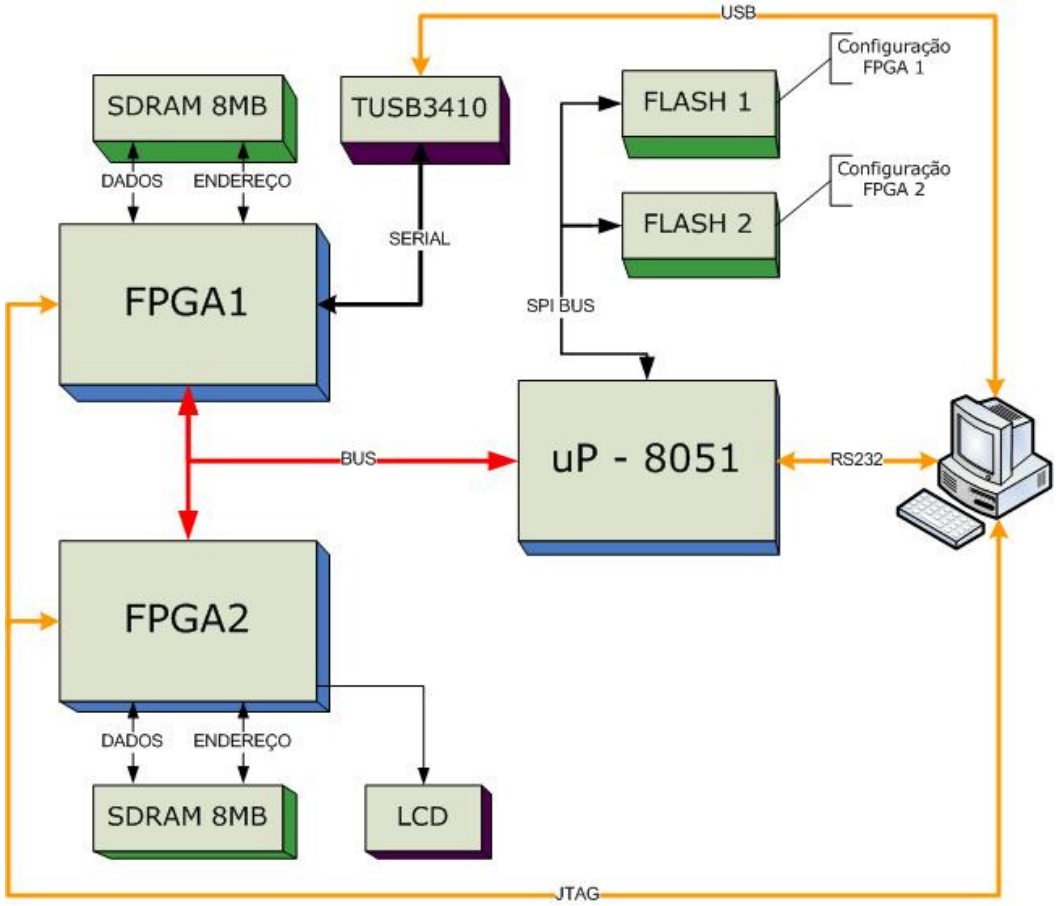


Figura 7.11 – Diagrama geral da placa para ensaios conduzidos.

2. A placa possui 2 FPGA's Xilinx XC3S500E (500k gates) 256 pinos;
3. Possui 1 microcontrolador MSC1211Y5 (*core* 8051) do fabricante Texas Instruments;
4. Possui 2 memórias SDRAM (*Synchronous Dynamic Random Access Memory*) MT48LC4M16A2 – 1 Meg x 16 x 4 bancos, formando um total de 8Mbytes por memória do fabricante Micron;
5. Possui 4 memórias Flash serias MP25P20 de 2Mbits cada, formando 2 bancos de 4Mbits do fabricante ST;
6. Possui comunicação serial RS232;

7. Possui comunicação USB;
8. Possui comunicação JTAG;
9. Possui uma conexão para um display LCD 16x2;
10. Possui 2 osciladores de 49.152MHz, um para cada FPGA;
11. Possui 16 conectores SMA para injeção direta de RF;
12. Possui 2 conectores de expansão.

Esta nova plataforma foi elaborada com o propósito de atender as injeções de falhas tanto irradiadas (IEC 62132-2) quanto conduzidas (IEC 62132-4), para minimizar os custos de fabricação já que são elevados, por se tratar de uma placa multicamada (*multilayer*) de 6 camadas e com componentes em ambos os lados, TOP e BOTTON, conforme mostrado na Figura 7.12 e Figura 7.13 (o esquemático desta placa está incluso no ANEXO III).

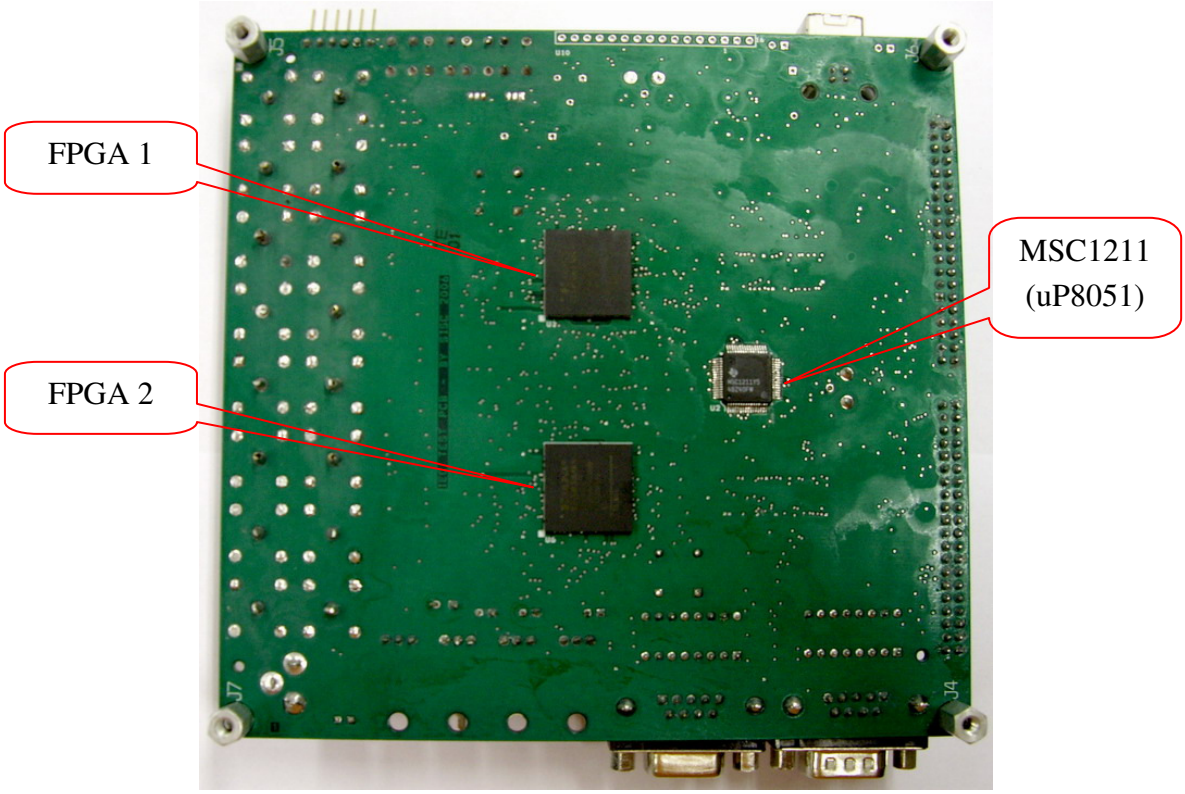


Figura 7.12 - Plataforma de testes de EMI vista do *layer TOP*.

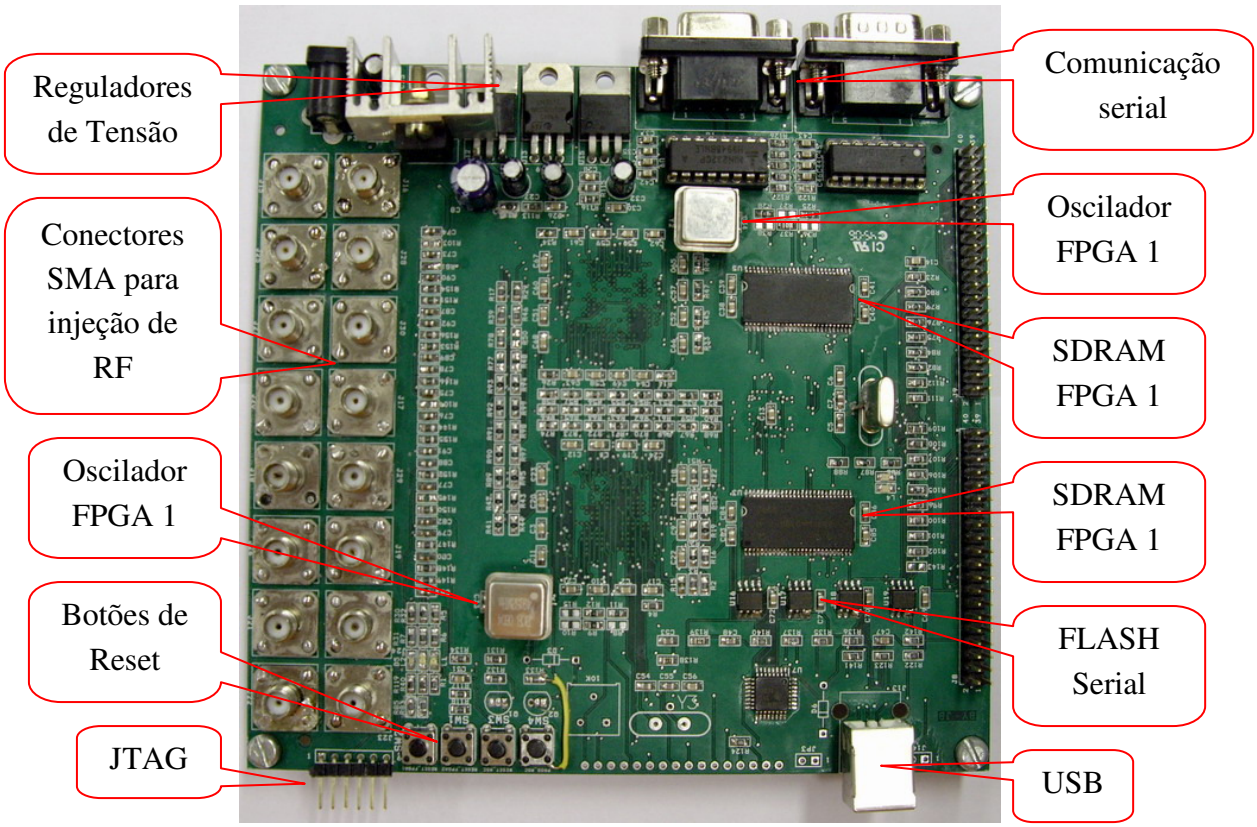


Figura 7.13 - Plataforma de testes de EMI vista do *layer BOTTON*.

PARTE III. RESULTADOS E CONCLUSÕES

8. RESULTADOS PRÁTICOS

8.1 Objetivos e Estudos de Caso

Este capítulo tem dois principais objetivos:

1. Demonstrar a aplicabilidade para validar SoCs em desenvolvimento;
2. Verificar a robustez da plataforma em função de ensaios de EMI de acordo com a norma IEC 62.132.

Baseado nos objetivos acima propostos, temos como estudo de caso para validação da plataforma de testes e desenvolvimento de SoCs as seguintes arquiteturas de SoCs para os ensaios de EMI, que são divididas em duas etapas:

1) Nesta primeira etapa foi utilizada a **plataforma de testes para ensaios irradiados** (VARGAS, 2006) conforme apresentado anteriormente na seção 7.3. Nesta primeira etapa de testes, foi utilizado, para validar a plataforma um SoC utilizando o processador MicroBlaze sob o comando do sistema operacional uC/OS-II juntamente com o a implementação de um *Watchdog* (WDP-IP) em hardware proposto e validado nesta plataforma por Leonardo Piccoli (PICCOLI, 2006) em sua dissertação de mestrado.

Este SoC tem a seguinte abordagem geral do sistema conforme ilustrado na Figura 8.1:

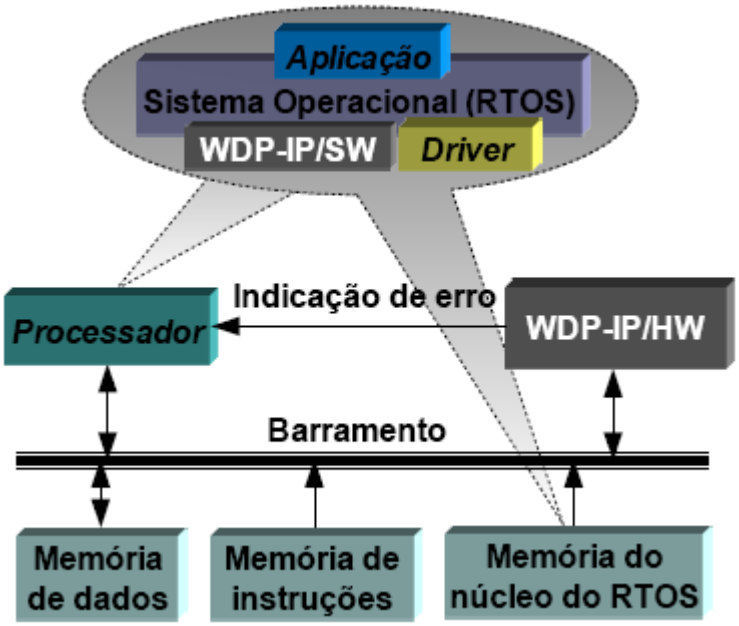


Figura 8.1 - Abordagem geral da arquitetura do WDP-IP proposta (PICCOLI, 2006).

A Figura 8.1 apresenta de modo geral a arquitetura e implementação proposta para o WDT-IP (PICCOLI, 2006), assumindo que o sistema alvo está executando, sob o

controle do RTOS uC/OS-II, em ambiente multitarefa. O WDP-IP possui duas implementações diferentes, a saber:

- WDP-IP/HW - implementado em ***hardware*** e descrito na linguagem VHDL;
- WDP-IP/SW - implementado em ***software*** e descrito na linguagem C ANSI (*American National Standards Institute*).

O WDP-IP apresenta detecção de erro de fluxo de controle (provenientes de perturbações externas como citado no capítulo 3) em tarefas na aplicação do usuário e no código do RTOS (*Real-Time Operating System*), através do controle do tempo de execução de cada um. Compreende-se que as falhas no sistema alvo que afetam o fluxo de controle são aquelas que obrigatoriamente mudam o tempo (acrescentando ou reduzindo) das tarefas monitoradas durante a execução do programa.

Deste modo, a fim de detectar a ocorrência de erros do fluxo de controle, a função do WDP-IP é monitorar o número de ciclos de *clock* que a aplicação utiliza para executar as tarefas do aplicativo.

A Figura 8.2 apresenta a estrutura em diagrama de blocos do SoC em hardware implementado. A descrição de cada bloco é abordada na sequência.

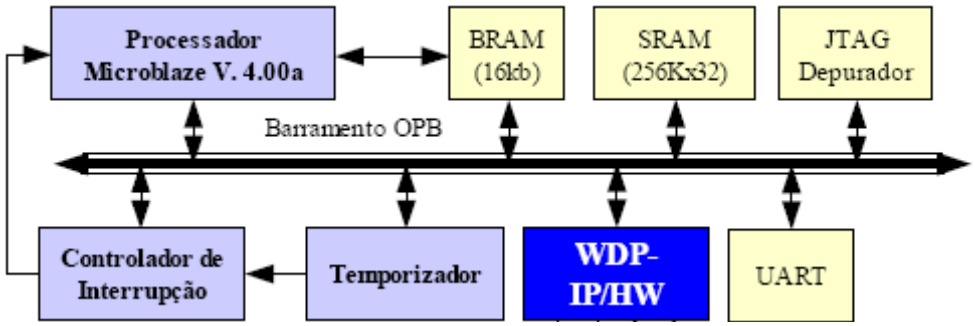


Figura 8.2 - Diagrama em blocos do SoC, salientando-se as conexões do processador, WDP-IP/HW e memórias através do barramento OPB (PICCOLI, 2006).

Bloco BRAM:

A plataforma contém blocos de memória interna (*Block* RAM - BRAM) 216-Kbit. A BRAM tem duas portas totalmente independente de acesso (dual port), entretanto, pode-se ler e escrever na mesma posição que será fornecido os dados velhos ou recentemente escritos (PICCOLI, 2006).

Os dados e instruções são armazenados normalmente na BRAM utilizando o controlador de BRAM para o barramento OPB (*On-chip Peripheral Bus*), ao se carregar o *bitstream* com o programa compilado. Porém, como este espaço de memória não é suficiente para carregar a aplicação utilizada nestes testes, esta área não foi utilizada.

Bloco JTAG:

O bloco JTAG é utilizado para conectar o aplicativo depurador (*debugger*) XMD/GDB do pacote de ferramentas de desenvolvimento da Xilinx. Pode servir como uma ferramenta para carregar na memória SRAM, o programa de aplicação e na elaboração das aplicações desenvolvidas pelo programador de software (PICCOLI, 2006).

Bloco Temporizador:

O bloco Temporizador consiste em um contador que decrementa ou incrementa na velocidade do *clock* do processador. O processador pode ler o contador e o Temporizador pode gerar um pulso de interrupção de saída quando o contador passar por zero.

Todos os RTOS (*Real-Time Operating System*) requerem uma interrupção periódica do sistema ou um "*tick tack*" (*clock tick*) para executar seus serviços. O temporizador para os experimentos com o WDP-IP/HW, foi utilizado em conjunto com o bloco de interrupção para gerar uma interrupção periódica no processador. Este temporizador foi ajustado para gerar uma interrupção a cada 10 ms (PICCOLI, 2006).

Bloco UART:

O bloco UART é usado para conectar o microcomputador via porta serial com a placa de desenvolvimento onde encontra-se o processador Microblaze e as técnicas embarcadas na FPGA, permitindo assim a visualização dos dados enviados da aplicação, sob o controle do RTOS, pela serial e mostrado na tela do microcomputador (PICCOLI, 2006).

Bloco Controlador de Interrupção:

É um controlador de interrupção simples e parametrizável que está conectado no barramento OPB. O controlador de interrupção esta conectado na saída da interrupção do Temporizador com a entrada da interrupção do processador (PICCOLI, 2006).

Bloco WDP-IP/HW:

O WDP-IP/HW é a implementação do WDP-IP em hardware cuja função é a de monitorar a aplicação a partir dos tempos de execução das tarefas do usuário ou do núcleo do uC/OS-II no modo supervisor. Indicar qual tarefa não cumpriu, seus prazos de tempo requerido caso o tempo de execução ultrapassar um valor determinado ou quando este tempo não foi atingido, ou se durante algumas execuções do sistema a tarefa não foi executada (PICCOLI, 2006).

Maiores detalhes e informações sobre a implementação do WDP-IP poderão ser encontradas na dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia da Pontifícia Universidade Católica do Rio Grande do Sul, a PUCRS e apresentada em agosto de 2006 e pode ser acessada diretamente pelo site http://verum.pucrs.br/ppgee .

2) Na segunda etapa foi utilizada a **plataforma de testes para ensaios conduzidos** (VARGAS, 2007) (VARGAS, 2007a) apresentada anteriormente na seção

7.4. O SoC implementado nesta placa para validação da plataforma de testes e desenvolvimento tem a arquitetura ilustrada na Figura 8.3.

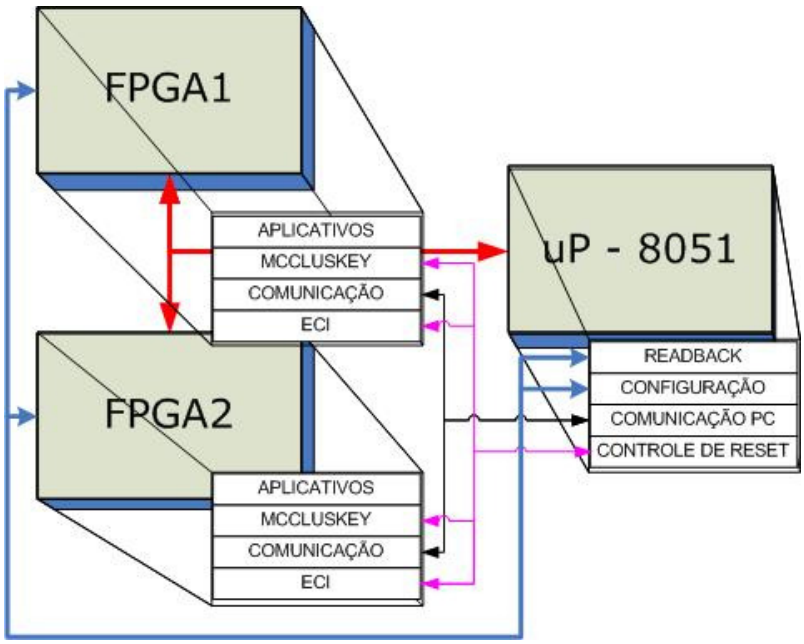


Figura 8.3 – Diagrama de funcionamento da placa de testes conduzidos.

Descrição dos blocos da arquitetura para ensaios proposta:

Bloco FPGA 1:

Dispositivo sobre teste 1 (DUT – *Device Under Test*). O FPGA 1, como mostra a Figura 8.4, contém:

- O microprocessador (Microblaze Xilinx) que é a CPU principal e que executa as aplicações do usuário;
- As funções de comunicação serial (Uart Lite Xilinx), para comunicação com o microcomputador, onde a saída (resultados) das aplicações e os avisos de algum erro encontrado são transmitidos para esse microcomputador gerenciador que armazena em um log para posterior verificação;
- O Watchdog IP em VHDL proposto por Piccoli (PICCOLI, 2006), que monitora o tempo de execução das aplicações. O WDP-IP de antemão sabe o tempo que cada aplicação leva para executar em suas condições normais de operação, então quando uma aplicação demora mais ou menos tempo para executar por algum distúrbio provocado, o WDP-IP detecta e sinaliza o usuário através de um LED (*Light Emitting Diode***)**;
- O JTAG é a interface de monitoração e de configuração do FPGA que é feito pelo microcomputador;
- A *Block* RAM é a memória interna do FPGA, que está dividida em duas partes: **(i)** usado como memória de programa onde estão contidas as

aplicações do usuário e protegidas com a técnica de CFCSS proposta por McCluskey (MCCLUSKEY, 2002) da forma simplificada (VARGAS, 2007b), ou seja, com alguns blocos básicos protegidos pela técnica escolhidos através de estatísticas do caminho crítico da aplicação (caminho mais executado pelo processador); **(ii)** o restante da memória onde não havia código algum, foi coberto com a técnica de ECI (MIREMADI, 1995), pois caso o *program counter* do microprocessador salte para um endereço onde não exista código, essa técnica o fará reportando via comunicação serial ao programa monitor externo, que reside no microcomputador. Esta mensagem avisa que o *program counter* do microprocessador caiu em um endereço de memória não válido;

- A *flag* de sinalização de erro do FPGA 2 serve para monitorar se houve erro detectado por CFCSS por parte do FPGA 2 que é feita através de um pino de i/o diretamente ligado entre as duas. Quando o FPGA 2 detecta um erro de fluxo de controle avisa o FPGA 1 colocando este pino em nível lógico "1". Com isso o FPGA 1 avisa o microcontrolador MSC1211 (uP8051) que faça a operação de *reset* no dispositivo com defeito e posterior operação de *readback* do *bitstream*;
- A *flag* de sinalização de erro para o 8051 é quando a técnica de CFCSS da FPGA 1 detecta algum erro de fluxo de controle da aplicação, ela assinala através de um pino de i/o ligado diretamente ao uP8051 colocando em nível lógico "1" onde é resetado pelo microcontrolador e tem seu *bitstream* verificado pelo microcomputador através do *readback*.

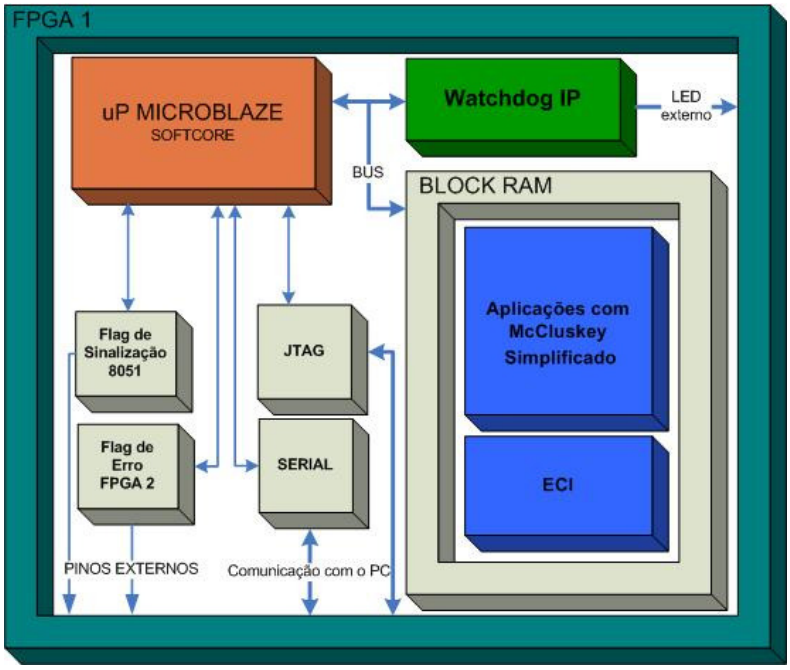


Figura 8.4 – Arquitetura do SoC contido no FPGA 1.

Bloco FPGA 2:

Dispositivo sobre teste 1 (DUT – *Device Under Test*). O FPGA 2, como mostra a Figura 8.5, contém:

- O microprocessador (Microblaze Xilinx) que é a CPU principal e que executa as aplicações do usuário;
- As funções de comunicação serial (Uart Lite Xilinx), para comunicação com o microcomputador, onde a saída (resultados) das aplicações e os avisos de algum erro encontrado são transmitidos para o microcomputador gerenciador que armazena em um log para posterior verificação;
- O Watchdog IP em VHDL proposto por Piccoli (PICCOLI, 2006), que monitora o tempo de execução das aplicações. O WDP-IP de antemão sabe o tempo que cada aplicação leva para executar em suas condições normais de operação, então quando uma aplicação demora mais ou menos tempo para executar por algum distúrbio provocado, o WDP-IP detecta e sinaliza o usuário através de um LED (*Light Emitting Diode***)**;
- O JTAG é a interface de monitoração e de configuração do FPGA que é feito pelo microcomputador;
- A *Block* RAM é a memória interna do FPGA, que está dividida em duas partes: **(i)** usado como memória de programa onde estão contidas as aplicações do usuário e protegidas com a técnica de CFCSS proposta por McCluskey (MCCLUSKEY, 2002) da forma completa (VARGAS, 2007b), ou seja, com todos os blocos básicos protegidos pela técnica; **(ii)** o restante da memória onde não havia código algum, foi coberto com a técnica de ECI (MIREMADI, 1995), pois caso o *program counter* do microprocessador salte para um endereço onde não exista código, essa técnica o fará reportando via comunicação serial ao programa monitor externo, que reside no microcomputador. Esta mensagem avisa que o *program counter* do microprocessador caiu em um endereço de memória não válido;
- A *flag* de sinalização de erro do FPGA 1 serve para monitorar se houve erro detectado por CFCSS por parte do FPGA 1 que é feita através de um pino de i/o diretamente ligado entre as duas. Quando o FPGA 1 detecta um erro de fluxo de controle avisa o FPGA 2 colocando este pino em nível lógico "1". Com isso o FPGA 2 avisa o microcontrolador MSC1211 (uP8051) que faça a operação de *reset* no dispositivo com defeito e posterior operação de *readback* do *bitstream*;
- A *flag* de sinalização de erro para o 8051 é quando a técnica de CFCSS do FPGA 2 detecta algum erro de fluxo de controle da aplicação, ela assinala através de um pino de i/o ligado diretamente ao 8051 colocando em nível lógico "1" onde é resetado pelo microcontrolador e

tem seu *bitstream* verificado pelo microcomputador através do *readback*.

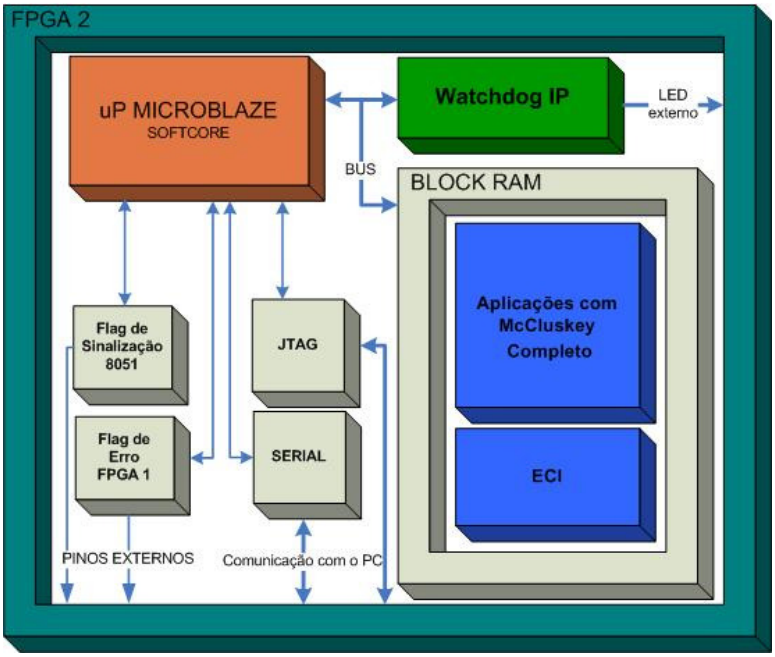


Figura 8.5 - Arquitetura do SoC contido no FPGA 2.

Bloco MSC1211:

Dispositivo sobre teste 3 (DUT – *Device Under Test*) conforme mostrado na Figura 8.3, contém:

- Os aplicativos de monitoração dos FPGAs. Cada FPGA tem um pino de i/o dedicado e ligado diretamente ao microcontrolador com a função de indicação de erro de fluxo de controle por parte do FPGA. Quando ocorre algum erro de fluxo de controle o uP8051 informa o usuário através da serial enviando uma mensagem sinalizando qual o FPGA deverá ser feito *readback* para verificação do hardware;
- O microcontrolador também tem a função de fazer a comunicação com o computador enviando os *logs* do sistema informando ao usuário o número de vezes em que os FPGAs foram reconfigurados, número de falhas ocorridas, números de *readbacks*, relatório de falhas de fluxo de controle detectados pelo CFCSS e demais técnicas de software já descritas anteriormente;
- Controle de *Reset*: O uC8051 tem a responsabilidade de controlar o *reset* dos processadores MicroBlaze, contidos nos FPGAs caso necessitem após algum erro detectado, seja ele por software (CFCSS) ou por hardware (*readback*).

Procedimento de Readback e Reconfiguração:

O procedimento de *readback* é responsável pela leitura do *bitstream* do FPGA e tem a finalidade de verificar se existe algum erro de configuração do FPGA, ou seja, algum erro de hardware.

O *readback* é feito de forma periódica em intervalos de tempos aleatórios feitos por uma função residente no microcomputador gerenciador. Então quando o tempo estoura, este programa residente chama o programa Impact da Xilinx que faz o *readback* completo dos FPGAs e caso encontre alguma diferença entre o *bitstream* original e sem erros armazenado no microcomputador como referência e o lido do FPGA, então é iniciado a reconfiguração completa do(s) FPGA(s) com problema(s). Esta operação pode ser feita também quando requerida a qualquer momento pelo uP8051, mostrado na tela por uma mensagem enviada pela serial que detectou algum erro em algum dos FPGAs.

8.2 Infra-estrutura e Procedimentos de Teste

Os equipamentos e procedimentos utilizados para injetarem falhas no DUT são resumidos e divididos em dois ambientes de testes, ensaios irradiados mostrado na Figura 8.6 e ensaios conduzidos mostrado na Figura 8.7.

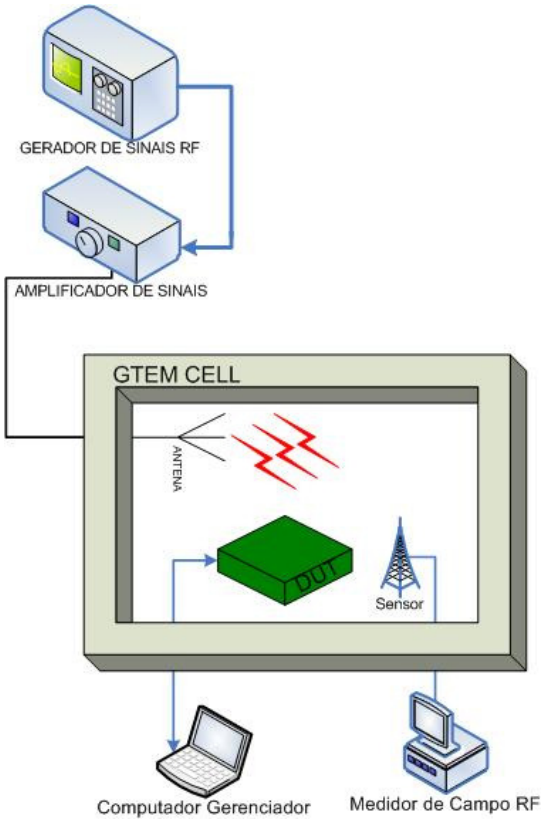


Figura 8.6 – Ambiente de testes para ensaios irradiados.

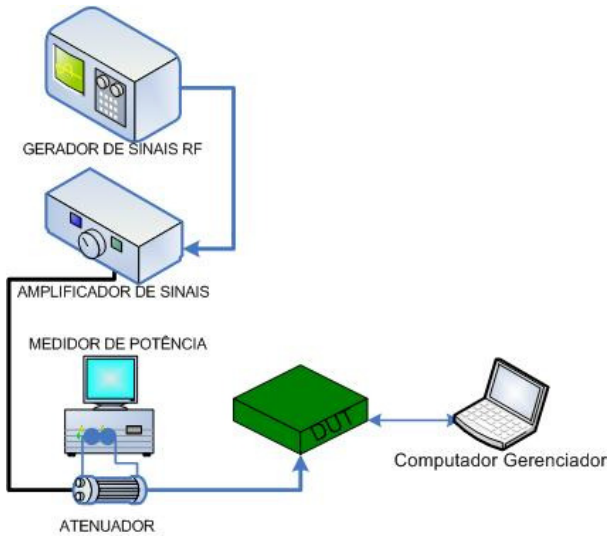


Figura 8.7 – Ambiente de testes para ensaios conduzidos.

Conforme ilustrado na Figura 8.6 que mostra o gerador de sinal RF que é utilizado para selecionar a freqüência do sinal portador, a freqüência e amplitude do sinal modulador. Este gerador de sinais tem a capacidade de gerar de sinais de 1Hz até 3GHz para testes de EMC. No próximo passo, o sinal portador já alterado pela modulante é amplificado pelo Amplificador para então alimentar a Célula GTEM. Este amplificador tem a capacidade de amplificar sinais de 1Hz até 3GHz e com uma potência de até 300 watts.

O campo eletromagnético desejado, no qual o DUT é exposto dentro da célula GTEM, é gerado combinando as seguintes variáveis: a freqüência do sinal portador, a freqüência, a amplitude e a potência do sinal de modulação. Um sensor de campo (Medidor de potência), como visto na Figura 8.6, é utilizado para medir o campo eletromagnético próximo ao DUT. Assim, é possível monitorar em tempo real, através do equipamento medidor de campo de RF, o campo ao qual o DUT está exposto naquele momento.

O ambiente de testes para ensaios irradiados descrito acima é utilizado para certificar a compatibilidade eletromagnética (EMC) de dispositivos eletrônicos. Esses testes são realizados seguindo normas internacionais, como por exemplo, as ditadas pela IEC, onde escolhemos a IEC 62.132 para os ensaios.

O seguinte procedimento de testes foi utilizado para os ensaios irradiados:

Conforme mostrado na Figura 8.8, a seqüência dos testes são da seguinte forma:

- O DUT é ligado e tem o *bitstream* de configuração do FPGA carregado através JTAG pelo programa Impact da Xilinx executado no microcomputador gerenciador;

- A aplicação é escolhida e carregada na memória RAM ou *Block* RAM (neste caso foi utilizado a *Block* RAM), através do JTAG pelo programa EDK da Xilinx executado no microcomputador gerenciador;
- Acionamento do gerador de sinais;
- Acionamento do amplificador de sinais;
- Ajuste da freqüência e da potência desejada para o teste;
- *Reset* do processador através do JTAG pelo programa EDK da Xilinx Xilinx executado no microcomputador gerenciador;
- Houve erro na saída da aplicação? (conforme enviado pela serial e mostrado na tela do computador gerenciador através do programa *HyperTerminal* do *Windows* executado neste mesmo microcomputador gerenciador). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- Houve erro de controle de fluxo conforme? (conforme enviado pela serial e mostrado na tela do computador gerenciador através do programa *HyperTerminal* do *Windows* executado neste mesmo microcomputador gerenciador). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- Houve erro de tempo de execução da aplicação? (monitorado pelo WDP-IP (PICCOLI, 2006) sinalizado visualmente através de um LED). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- O *readback* do FPGA é realizado para verificar se houve modificações no *bitstream* do hardware do FPGA. Caso apresente diferença na comparação entre o *bitstream* armazenado no FPGA e o *bitstream* original, é conduzido para o procedimento de reconfiguração do *bitstream* que é carregado novamente e reinício total do sistema. E se não houve erro de configuração, o sistema é apenas resetado e registrado na tabela;
- Fim do teste. É conduzido para o próximo intervalo de freqüência, registrado na tabela e *reset* do processador para reinício dos testes.

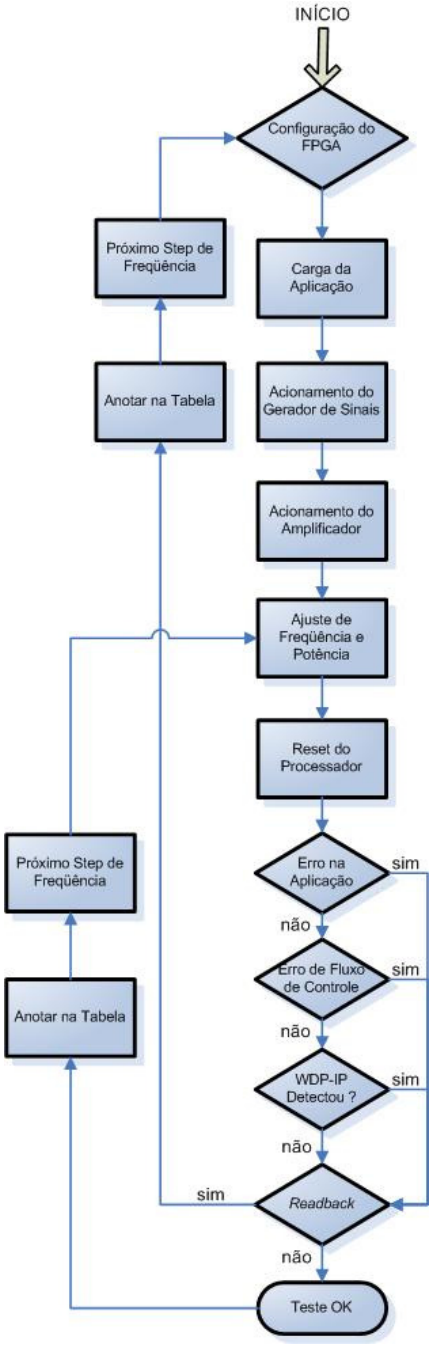


Figura 8.8 – Fluxograma do procedimento de testes para ensaios irradiados.

O ambiente para ensaios conduzidos mostrado na Figura 8.7 dispõe também de um gerador de sinais e um amplificador. A diferença é que a injeção do ruído de RF é feita de forma direta, feita por um cabo coaxial do amplificador diretamente ao DUT conectado através de um conector SMA. Entre o DUT e o amplificador há um atenuador e medidor de potência para medir a potência real injetada no circuito sobre teste assim como sua potência refletida de RF, pois o circuito sob teste pode não absorver toda a potência injetada refletindo parte dela de volta para o amplificador.

O seguinte procedimento de testes foi utilizado para os ensaios conduzidos:

Conforme mostrado na Figura 8.9, a seqüência dos testes são da seguinte forma:

- O DUT é ligado e tem o *bitstream* de configuração do FPGA carregado através do JTAG pelo programa Impact da Xilinx executado no microcomputador gerenciador;
- A aplicação é escolhida e carregada na memória RAM ou *Block* RAM (neste caso foi usado a *Block* RAM), através do JTAG pelo programa EDK da Xilinx Xilinx executado no microcomputador gerenciador;
- É escolhido o ponto de teste a ser injetado dentre os 16 pontos existentes na placa. Então é conectado através de um conector SMA por um cabo coaxial do amplificador até o DUT;
- Acionamento do gerador de sinais;
- Acionamento do amplificador de sinais;
- Ajuste da freqüência e da potência desejada para o teste;
- *Reset* do processador através do JTAG pelo programa EDK da Xilinx Xilinx executado no microcomputador gerenciador;
- Houve erro na saída da aplicação? (conforme enviado pela serial e mostrado na tela do computador gerenciador através do programa *HyperTerminal* do *Windows* executado neste mesmo microcomputador gerenciador). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- Houve erro de controle de fluxo? (conforme enviado pela serial e mostrado na tela do computador gerenciador através do programa *HyperTerminal* do *Windows* executado neste mesmo microcomputador gerenciador). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- Houve erro de tempo de execução da aplicação? (monitorado pelo WDP-IP (PICCOLI, 2006) sinalizado visualmente através de um LED). Se sim, é conduzido para o procedimento de *readback* e é registrado na tabela, se não há erros, vai para o próximo passo;
- O *readback* do FPGA é realizado para verificar se houve modificações no *bitstream* do hardware do FPGA. Caso apresente diferença na comparação entre o *bitstream* armazenado no FPGA e o *bitstream* original, é conduzido para o procedimento de reconfiguração do *bitstream* que é carregado novamente e reinício total do sistema. E se não houve erro de configuração, o sistema é apenas resetado e registrado na tabela;
- Fim do teste. É conduzido para o próximo intervalo de freqüência, registrado na tabela e reset do processador para reinício dos testes.

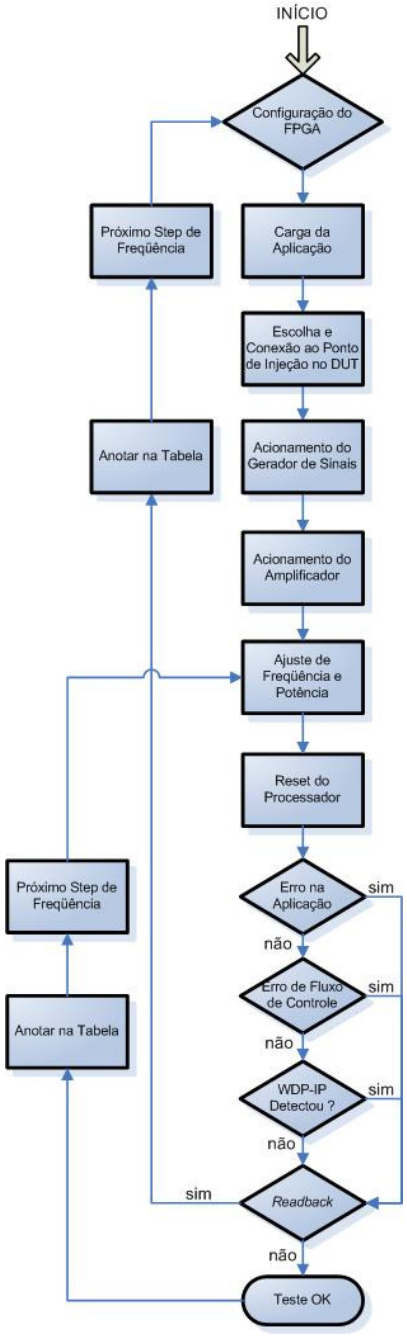


Figura 8.9 - Fluxograma do procedimento de testes para ensaios conduzidos.

8.3 Ensaio de Imunidade para EMI Irradiada

Os ensaios de imunidade à EMI irradiada foram divididos em três partes, ou em três ensaios diferentes conforme explanado a seguir:

8.3.1 Parte I

A primeira etapa de ensaios para testes de EMI irradiada foram realizadas na plataforma de testes para ensaios irradiados, mostrado na

Figura 7.7 e na Figura 7.8 pelo aluno e colega de mestrado, Leonardo Piccoli (PICCOLI, 2006) em julho de 2006 nos laboratórios do INTI em Buenos Aires como base de testes para sua dissertação de mestrado onde maiores informações podem ser encontradas diretamente pelo site http://verum.pucrs.br/ppgee. A arquitetura de testes e resumo dos resultados serão apresentados abaixo:

Os testes se dividiram basicamente em duas etapas, na primeira etapa (Figura 8.10) o DUT foi colocado seguindo os padrões convencionados para a Célula GTEM, onde somente o FPGA estaria exposto às irradiações. Nos testes realizados nessa etapa não foi detectado nenhuma falha ativada no sistema, para freqüências que variaram de 1MHz a 3GHz e campo elétrico que variou de 10V/m a 210V/m.

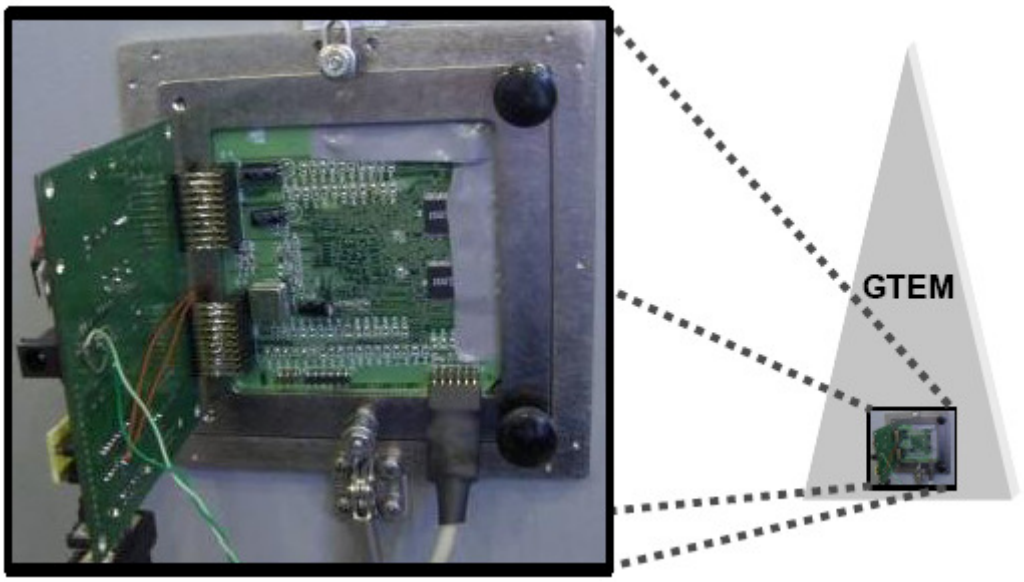


Figura 8.10 - DUT seguindo padrões de teste, mostrando o detalhe da placa de teste encaixada na porta de abertura da GTEM CELL (PICCOLI, 2006).

Na segunda etapa, como podemos ver na Figura 8.11, o DUT foi inserido dentro da Célula GTEM e os cabos foram isolados e colocados tubos de ferrite para não haver interferência na comunicação com o computador.

Sendo assim, a injeção de falhas pode ser realizada com sucesso para os testes realizados na segunda etapa.

Figura 8.11 – DUT, mostrando o detalhe da placa inserida dentro da Célula (PICCOLI, 2006).

Três aplicações foram utilizadas para os testes no qual serão apresentados os resumos dos resultados obtidos que são classificados quanto aos tipos de erros gerados que são ilustrados na Figura 8.13, Figura 8.16 e na Figura 8.19:

a) Erro na configuração (*bitstream*) do FPGA, onde são consideradas falhas "permanentes";
b) Erro na área de memória utilizada para armazenar o código, onde são consideradas falhas "transientes";
c) Outras falhas são aquelas que provocam a inversão de um ou mais bits (ex.: PC, ponteiro de pilha, instruções de registradores), ou na interferência na comunicação entre o FPGA e o computador gerenciador, entre outras..

O estado da saída da aplicação é enviado para a porta serial e exibida na tela do computador. Essas saídas foram marcadas na tabela quando se observou erros na execução do programa e classificou-se entre erros visuais que apenas alteram as informações coletadas ou que além de alterar essas informações, causaram uma parada forçada no sistema conforme será mostrado na Figura 8.14, na Figura 8.17 e na Figura 8.20.

Teste Utilizando Ordenador de Matrizes:

A Figura 8.12 apresenta um resumo da capacidade de detecção de erros das técnicas utilizadas durante o funcionamento do sistema no interior da célula GTEM, para teste de imunidade irradiada. Nota-se uma pequena taxa de detecção de erro na utilização de semáforos pelo RTOS uC/OS-II e uma significante diferença entre o WDP-IP/HW frente as demais técnicas. O WDP-IP+ implementado em software (o qual combina as técnicas YACCA e WDP-IP/SW) apresentou pouco acréscimo na taxa de detecção frente à versão do WDP-IP/SW. Já a técnica de detecção YACCA apresentou

menor taxa de detecção em relação às demais técnicas, salvo a detecção por violação dos semáforos, nativos do próprio RTOS uC/OS-II.

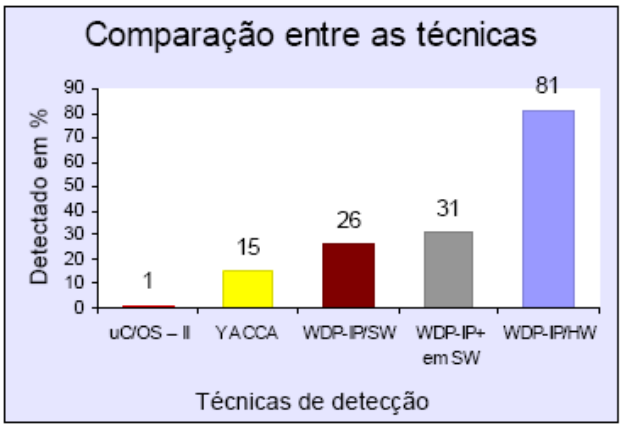


Figura 8.12 - Comparação entre as técnicas de detecção utilizando o algoritmo ordenador de matrizes (PICCOLI, 2006).

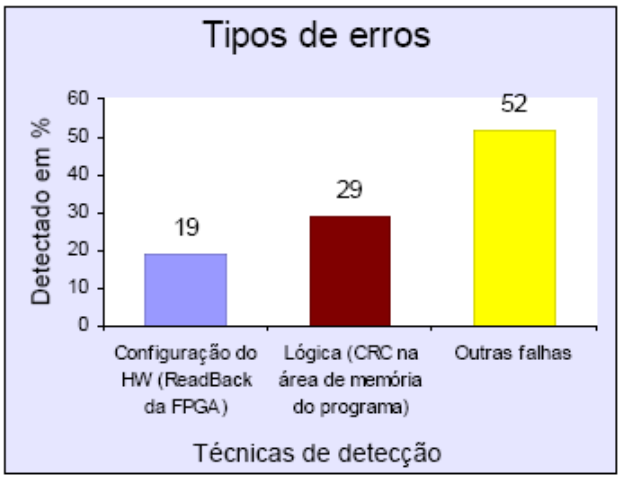


Figura 8.13 - Tipos de erros detectados utilizando o algoritmo ordenador de matrizes (PICCOLI, 2006).

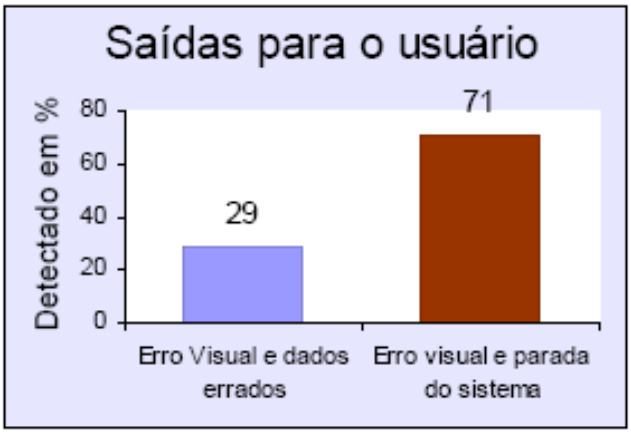


Figura 8.14 - Saídas do programa em execução utilizando o algoritmo ordenador de matrizes (PICCOLI, 2006).

Teste Utilizando Gerador de Números Primos:

A Figura 8.15 apresenta um resumo da capacidade de detecção de erros das técnicas utilizadas durante o funcionamento do sistema no interior da célula GTEM, para teste de imunidade irradiada. Nota-se uma pequena taxa de detecção de erro na utilização de semáforos pelo RTOS uC/OS-II e uma significante diferença entre o WDP-IP/HW frente as demais técnicas. O WDP-IP+ implementado em software (o qual combina as técnicas YACCA e WDP-IP/SW) apresentou pouco acréscimo na taxa de detecção frente a versão do WDP-IP/SW. Já a técnica de detecção YACCA ficou aquém na taxa de detecção em relação as demais técnicas, salvo a detecção por violação dos semáforos, nativos do próprio RTOS uC/OS-II.

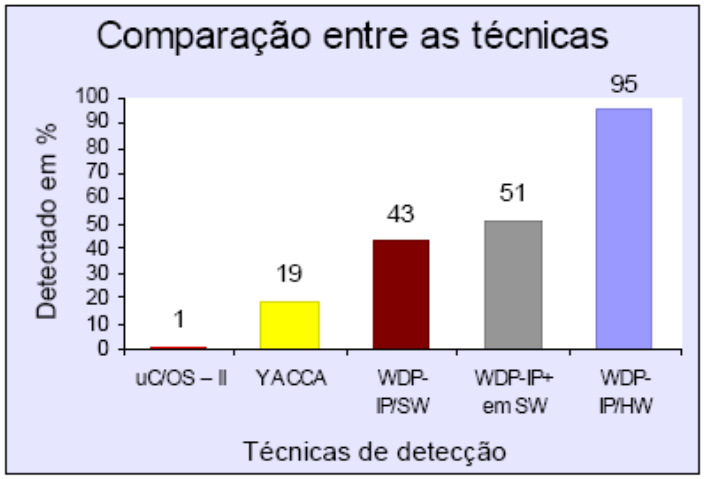


Figura 8.15 - Comparação entre as técnicas de detecção utilizando o algoritmo gerador de números primos (PICCOLI, 2006).

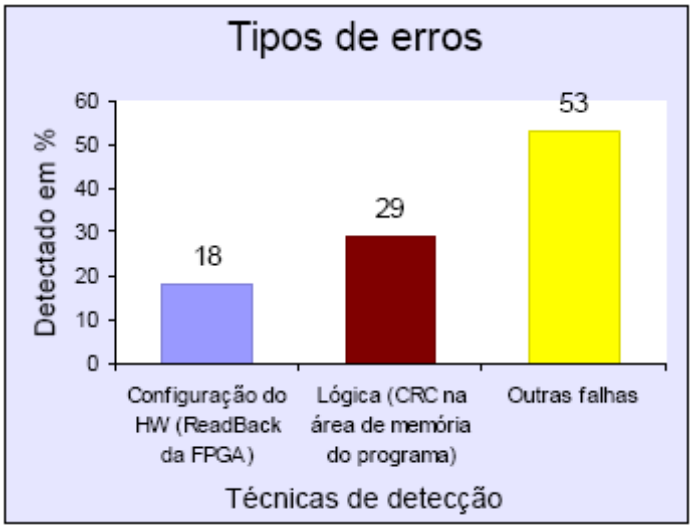


Figura 8.16 - Tipos de erros detectados utilizando o algoritmo gerador de números primos (PICCOLI, 2006).

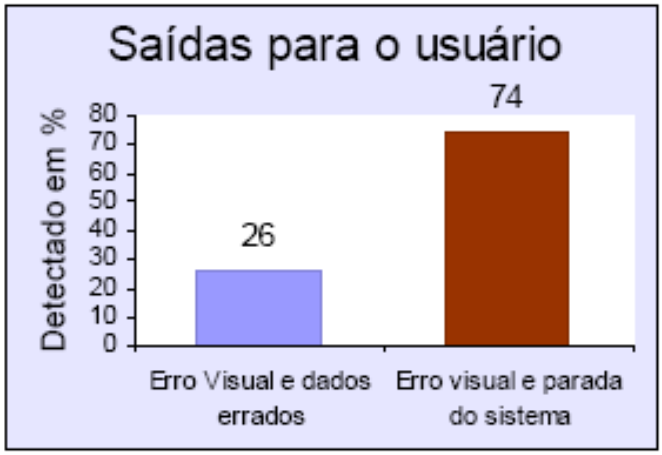


Figura 8.17 - Saídas do programa em execução utilizando o algoritmo gerador de números primos (PICCOLI, 2006).

Teste Utilizando Filtro IIR:

A Figura 8.18 apresenta um resumo da capacidade de detecção de erros das técnicas utilizadas durante o funcionamento do sistema no interior da célula GTEM, para teste de imunidade irradiada. Nota-se uma pequena taxa de detecção de erro na utilização de semáforos pelo RTOS uC/OS-II e uma significante diferença entre o WDP-IP/HW frente as demais técnicas. O WDP-IP+ implementado em software (o qual combina as técnicas YACCA e WDP-IP/SW) apresentou pouco acréscimo na taxa de detecção frente a versão do WDP-IP/SW. Já a técnica de detecção YACCA apresentou menor taxa de detecção em relação as demais técnicas, salvo a detecção por violação dos semáforos, nativos do próprio RTOS uC/OS-II.

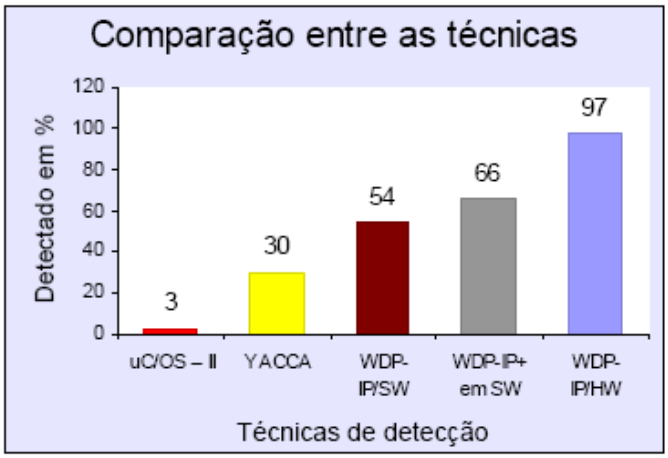


Figura 8.18 - Comparação entre as técnicas de detecção utilizando o algoritmo filtro IIR (PICCOLI, 2006).

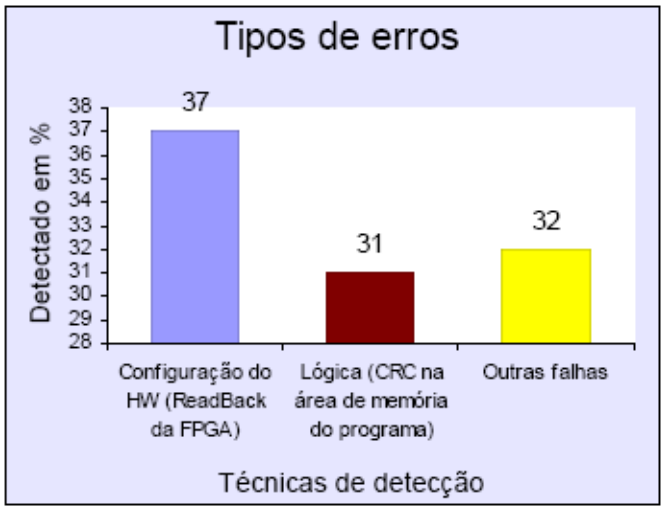


Figura 8.19 - Tipos de erros detectados utilizando o algoritmo filtro IIR (PICCOLI, 2006).

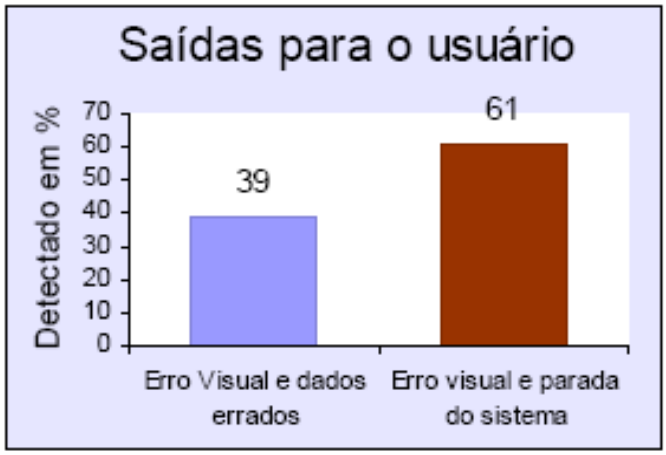


Figura 8.20 - Saídas do programa em execução utilizando o algoritmo filtro IIR (PICCOLI, 2006).

8.3.2 Parte II

A segunda etapa de ensaios para testes de EMI irradiada foi realizada na plataforma de testes para ensaios conduzidos (IEC 62.132-4), que também atende a norma para ensaios irradiados (IEC 62.132-2) conforme já mostrado na Figura 7.12 e na Figura 7.13 em janeiro de 2007 nos laboratórios do INTI em Buenos Aires na Argentina.

A estrutura de testes consiste na utilização dos seguintes equipamentos (igualmente aos testes da Parte II) conforme mostrado na Figura 8.24 e na Figura 8.25:

- Gerador de sinais de 1Hz a 3GHz;
- Amplificador de sinais de 1Hz a 3GHz e potência de 300 watts;
- GTEM Cell com capacidade de freqüências de até 18GHz;
- Sensor de campo para medição da potência irradiada.

Este teste foi realizado com a placa colocada dentro de uma caixa metálica de 12 cm de altura x 24 cm de comprimento e 17 cm de largura hermeticamente fechada mostrado na Figura 8.21. Esta caixa tem apenas uma abertura exatamente do tamanho da placa de testes, ou seja, 14 cm x 15 cm com suportes para a fixação da mesma por quatro parafusos e uma abertura de 3 cm x 1,5 cm para saída dos cabos ligados a placa.

A caixa tem a finalidade de se comportar como uma Gaiola de Faraday[7] aterrada junto a placa e ao terra da Célula GTEM, para terem a mesma referência de terra e assim proteger todos os componentes e trilhas de sinais que estão do lado BOTTON (baixo) da placa, ou seja, as partes que não se deseja expor a interferência eletromagnética. Assim, desta maneira, temos uma semelhança com a GTEM Cell (Figura 7.6) para testes de circuitos integrados que aceita somente placas de teste no tamanho de 10cm x 10cm, e desta maneira podemos fazê-la de qualquer tamanho, apenas elaborando uma caixa adequada ao tamanho da placa a ser testada. Assim, somente os CIs a serem testados pela exposição a interferência ficam no lado TOP (cima) da placa e o restante não.

Após a fixação da placa de teste na caixa metálica então o conjunto (caixa + placa) é colocado dentro da Célula GTEM para injeção de falhas conforme mostrado na Figura 8.22.

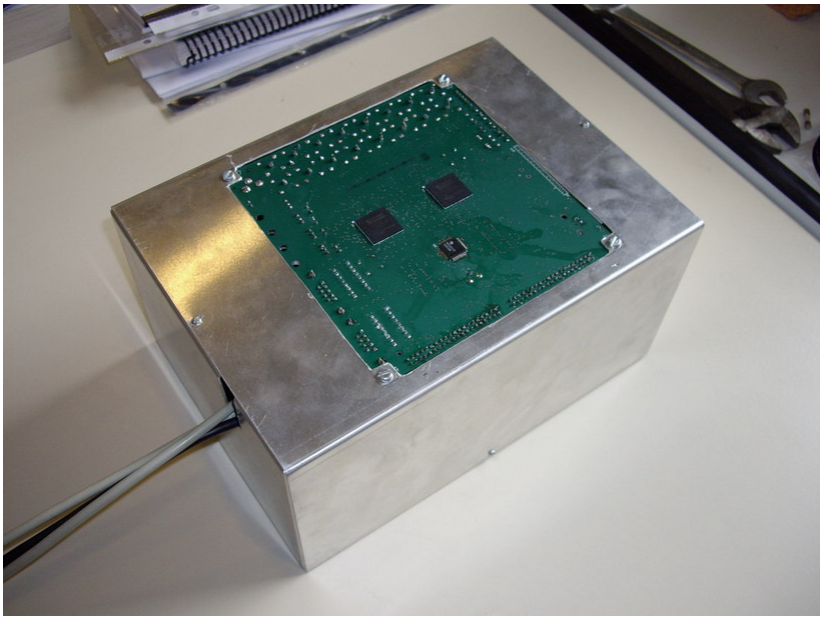


Figura 8.21 – Detalha da caixa metálica juntamente com a placa de testes devidamente encaixada e aparafusada.

[7] Uma Gaiola de Faraday é uma blindagem elétrica, ou seja, uma superfície condutora que envolve uma dada região do espaço e que pode, em certas situações, impedir a entrada de perturbações produzidas por campos elétricos e ou eletromagnéticos externos.

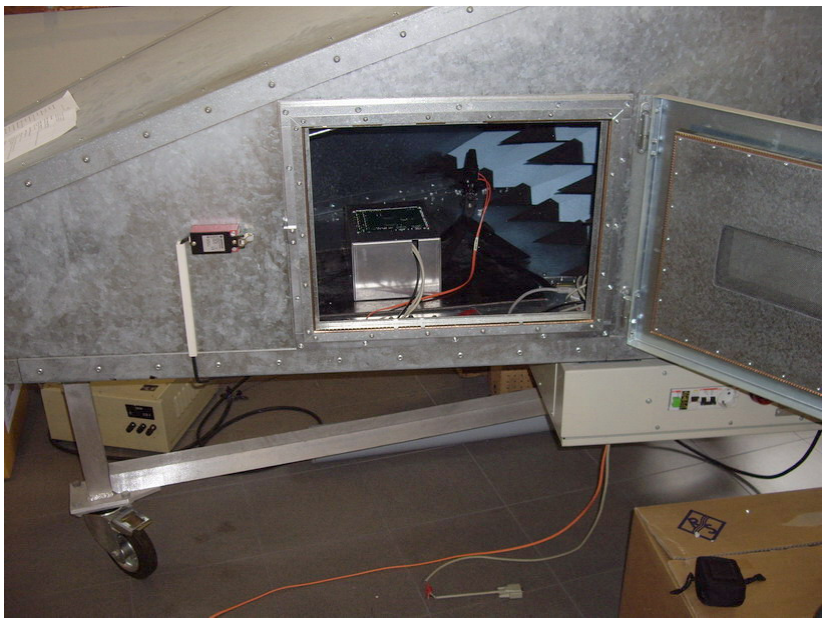


Figura 8.22 – Detalhe do conjunto caixa + placa dentro da GTEM Cell para os testes de ensaios irradiados.

Nesta etapa de testes, foram utilizados freqüências de 150kHz a 3GHz e uma potência de campo elétrico de 33V/m a 186V/m, conforme o gráfico mostrados na Figura 8.28.

Conforme o gráfico mostrado na Figura 8.23, as freqüências utilizadas foram de 150kHz a 3GHz divididos nos seguintes intervalos conforme mostrado na Tabela 8.1:

FAIXA DE FREQUÊNCIA	INTERVALO	EXEMPLO DE VARIAÇÃO
150kHz a 1MHz	100kHz	250, 350, 450kHz...1MHz
1MHz a 50MHz	1MHz	2,3,4...50MHz
49.100MHz a 49.200MHz	2kHz	49.102, 49.104...49.200MHz
50MHz a 100MHz	10MHz	60, 70, 80.. 100MHz
100MHz a 500MHz	20MHz	120, 140...500MHz
500MHz a 3GHz	50MHz	550, 600, 650...3GHz

Tabela 8.1 – Tabela dos intervalos de freqüência utilizados nos testes para ensaios irradiados com a placa utilizando a caixa de blindagem.

Como se observa na Tabela 8.1 nas freqüências próximas a freqüência do *clock*, que é de 49.152MHz, utilizamos um intervalo mais fino de freqüência com o objetivo de procurar uma maior susceptibilidade do sistema por se tratar de uma freqüência crítica para este sistema.

A potência do amplificador neste caso, estava sempre configurado para o máximo, então nota-se que conforme a freqüência varia, a potência varia também. Isso

se dá devido às características de reverberação da Célula GTEM que tem um comportamento não linear conforme a variação da freqüência.

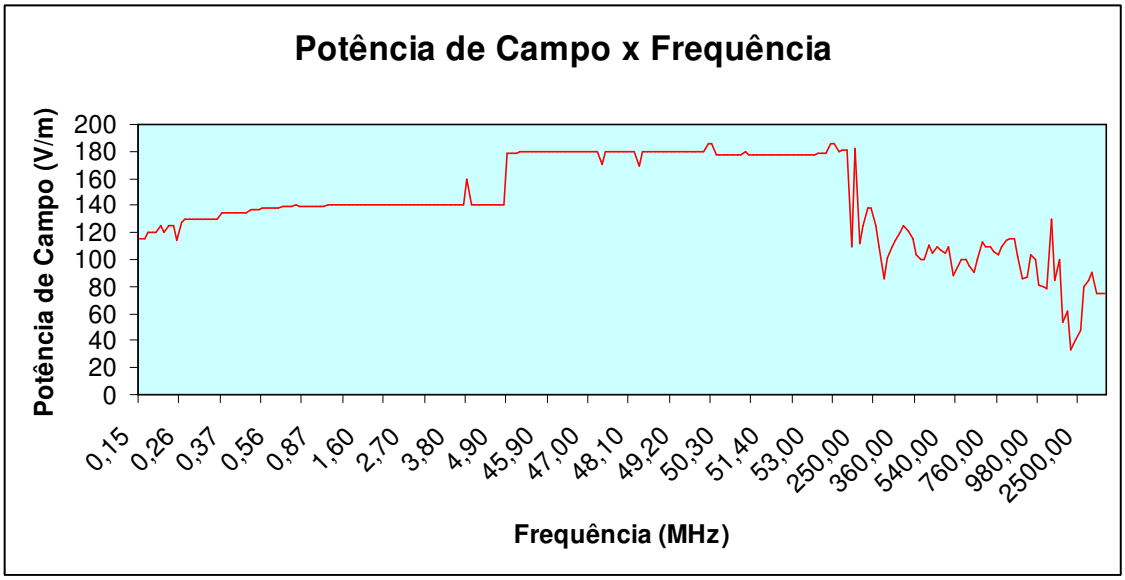


Figura 8.23 - Gráfico da potência de campo versus freqüência aplicada nos testes para ensaios irradiados com a placa com a caixa de blindagem.

Para esta etapa dos testes, foram utilizadas as seguintes técnicas de detecção de falhas:

- Aplicações protegidas pela técnica de CFCSS (completa (MCCLUSKEY, 2002) e simplificada (VARGAS, 2007b)) que tem a finalidade de detectar erros de fluxo de controle do processador. Neste caso aplicação foi carregada na *Block* RAM interna ao FPGA;
- WDP-IP em hardware (descrito em VHDL e instanciado no barramento do processador Microblaze da Xilinx) para monitoração do tempo de execução das aplicações (PICCOLI, 2006);
- ECI (MIREMADI, 1995) cobrindo o restante das *Block* RAMs, que tem a finalidade de detectar erros de fluxo de controle do processador da parte da memória não coberta pela técnica de CFCSS, ou seja onde não existe código algum;
- READBACK (BEZERRA, 2001) que é a verificação do *bitstream* de configuração do FPGA com a finalidade de detectar erros no hardware do FPGA.

No que diz respeito às aplicações, foram utilizadas três para os testes efetuados nesta etapa, e cada uma das aplicações tiveram duas versões:

Algoritmo multiplicador de matrizes:

- Protegida com a técnica CFCSS da forma completa proposta originalmente por McCluskey (MCCLUSKEY, 2002), que consiste em atribuir assinaturas e testes em todos os blocos básicos do aplicativo;
- Protegido com a técnica CFCSS da forma simplificada proposta por Vargas (VARGAS, 2007b), que consiste em atribuir assinaturas e testes em um número reduzido de blocos básicos, baseados em dados estatísticos determinados pelo caminho crítico da aplicação, ou seja o caminho mais executado da mesma.

Algoritmo utilizando um Filtro FIR:

- Protegida com a técnica CFCSS da forma completa proposta originalmente por McCluskey (MCCLUSKEY, 2002), que consiste em atribuir assinaturas e testes em todos os blocos básicos do aplicativo;
- Protegido com a técnica CFCSS da forma simplificada proposta por Vargas (VARGAS, 2007b), que consiste em atribuir assinaturas e testes em um número reduzido de blocos básicos, baseados em dados estatísticos determinados pelo caminho crítico da aplicação, ou seja, o caminho mais executado da mesma.

Algoritmo gerador de números primos (Crivo de Erastóstenes):

- Protegida com a técnica CFCSS da forma completa proposta originalmente por McCluskey (MCCLUSKEY, 2002), que consiste em atribuir assinaturas e testes em todos os blocos básicos do aplicativo;
- Protegido com a técnica CFCSS da forma simplificada proposta por Vargas (VARGAS, 2007b), que consiste em atribuir assinaturas e testes em um número reduzido de blocos básicos, baseados em dados estatísticos determinados pelo caminho crítico da aplicação, ou seja, o caminho mais executado da mesma.

As aplicações eram sempre executadas aos pares, por exemplo, na FPGA1 era executado, o Filtro FIR com CFCSS da forma completa e na FPGA2 era executado o Filtro FIR com CFCSS da forma simplificada. Esta configuração foi escolhida para comparar em tempo real a diferença da taxa de detecção entre as duas técnicas propostas. E assim sucessivamente para as outras aplicações.

Os seguintes resultados foram obtidos com esta etapa de testes:

Para os ensaios a imunidade irradiada com a placa utilizando a caixa de blindagem, foi utilizado a potência máxima e freqüência máxima fornecidas pelos equipamentos dos laboratórios do INTI, que nestes ensaios chegaram a 186V/m para a potência de campo dentro da Célula GTEM e de 150kHz até 3GHz para freqüência, que

foram aplicados ao DUT e não observou-se nenhuma falha. Ou seja, para a faixa de freqüências de 150kHz a 3GHz e para uma potência de campo elétrico de 33V/m a 186V/m, não se observou nenhuma ocorrência de falhas.

8.3.3 Parte III

A terceira etapa de ensaios para testes de EMI irradiada foi realizada na plataforma de testes para ensaios conduzidos (IEC 62.132-4), que também atende a norma para ensaios irradiados (IEC 62.132-2) conforme já mostrado na Figura 7.12 e na Figura 7.13 em janeiro de 2007 nos laboratórios do INTI em Buenos Aires na Argentina.

A estrutura de testes consiste na utilização dos seguintes equipamentos conforme mostrado na Figura 8.24 e na Figura 8.25:

- Gerador de sinais de 1Hz a 3GHz;
- Amplificador de sinais de 1Hz a 3GHz e potência de 300 watts;
- GTEM Cell com capacidade de freqüências de até 18GHz;
- Sensor de campo para medição da potência irradiada.

Este teste foi realizado com a placa colocada inteiramente dentro da Célula GTEM sem a caixa de blindagem para injeção de falhas conforme mostrado na Figura 8.26 e na Figura 8.27.

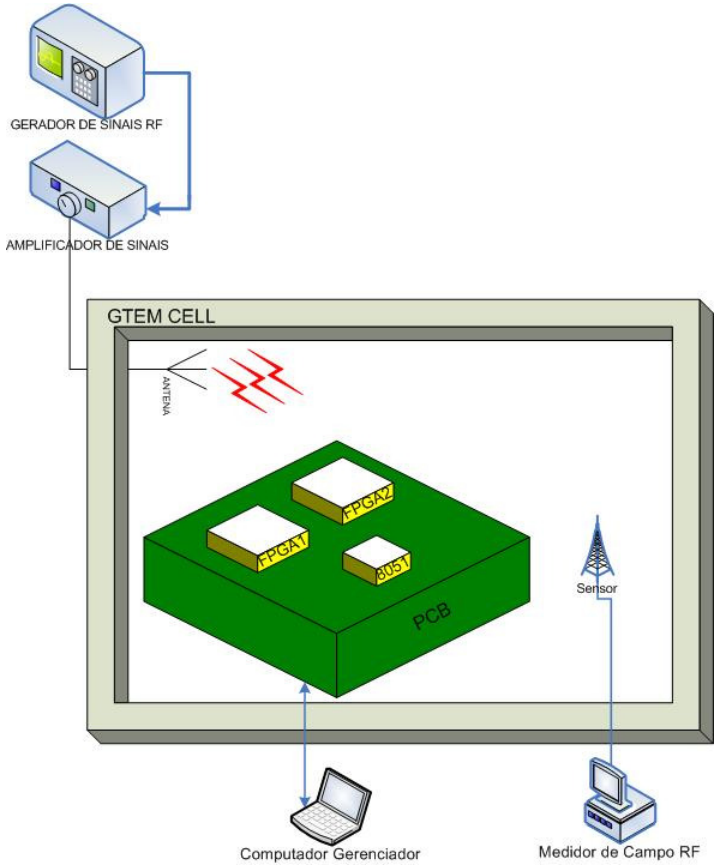


Figura 8.24 – Ambiente de teste de injeção de RF irradiado.

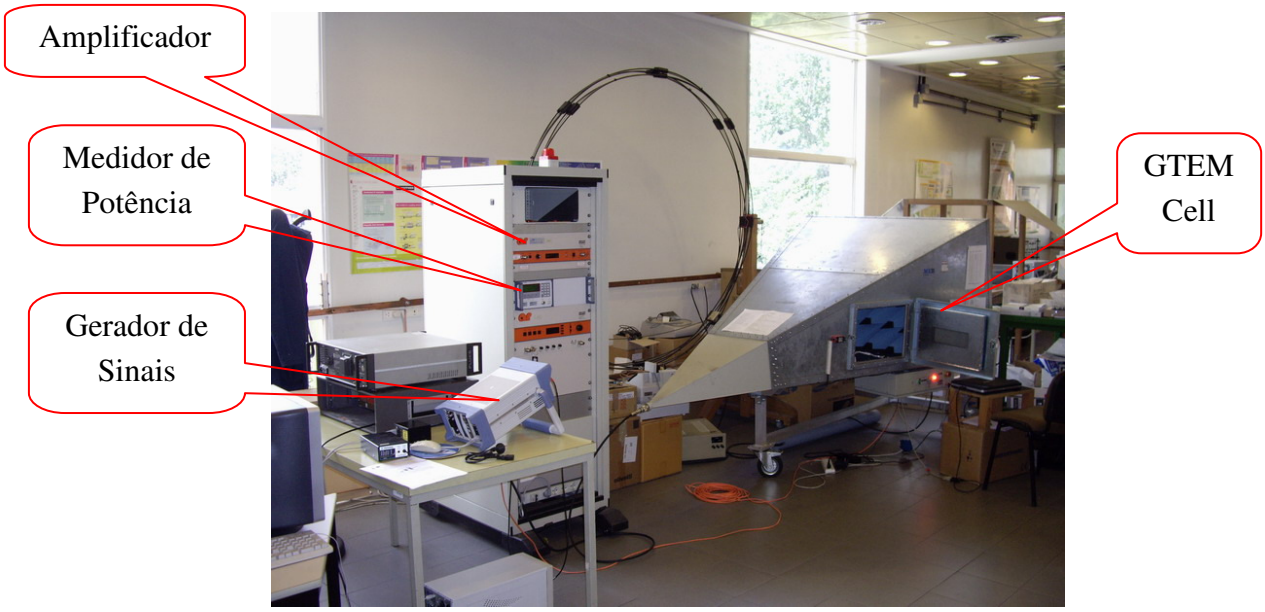


Figura 8.25 – Foto do ambiente de testes para ensaios irradiados nos laboratórios do INTI.

A placa de testes foi inteiramente colocada dentro da Célula GTEM a fim de se conseguir uma maior probabilidade de injeção de falhas e de se avaliar a capacidade de detecção de falhas das técnicas propostas.

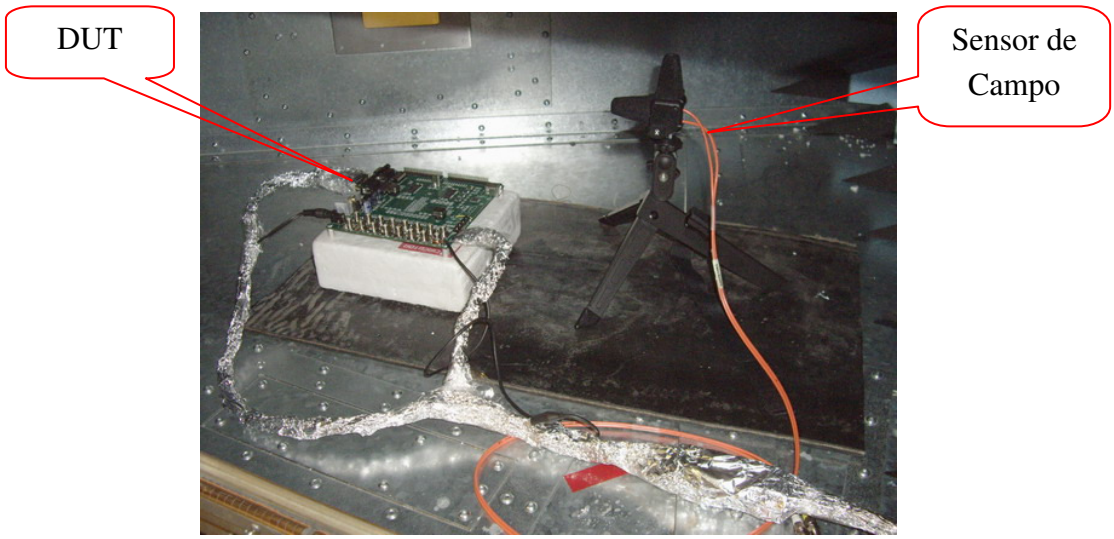


Figura 8.26 – Detalhe do DUT e do sensor de campo dentro da GTEM Cell para os ensaios irradiados.

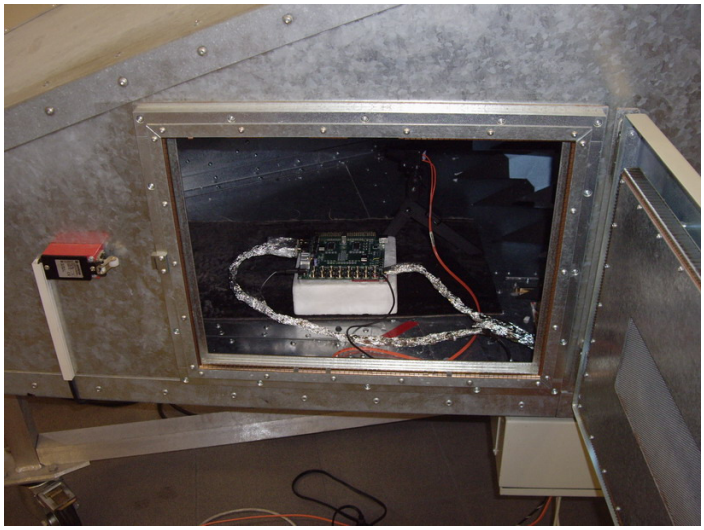


Figura 8.27 - Detalhe da GTEM Cell e do DUT posicionado dentro da célula para os ensaios irradiados.

Nesta etapa de testes, foram utilizados freqüências de 150kHz a 3GHz e uma potência de campo elétrico de 20V/m a 190V/m, conforme o gráfico mostrados na Figura 8.28.

Conforme o gráfico mostrado na Figura 8.28, as freqüências utilizadas foram de 150kHz a 3GHz divididos nos seguintes intervalos conforme mostrado na Tabela 8.2:

FAIXA DE FREQUÊNCIA	INTERVALO	EXEMPLO DE VARIAÇÃO
150kHz a 1MHz	100kHz	250, 350, 450kHz...1MHz
1MHz a 50MHz	1MHz	2,3,4...50MHz
49.100MHz a 49.200MHz	2kHz	49.102, 49.104...49.200MHz
50MHz a 100MHz	10MHz	60, 70, 80.. 100MHz
100MHz a 500MHz	20MHz	120, 140...500MHz
500MHz a 3GHz	50MHz	550, 600, 650...3GHz

Tabela 8.2 - Tabela dos intervalos de freqüência utilizados nos testes para ensaios irradiados com a placa utilizando a caixa de blindagem.

Como se observa na Tabela 8.2 nas freqüências próximas a freqüência do *clock*, que é de 49.152MHz, utilizamos um intervalo mais fino de freqüência com o objetivo de procurar uma maior susceptibilidade do sistema por se tratar de uma freqüência crítica para este sistema.

A potência do amplificador neste caso, estava sempre configurado para o máximo, então nota-se que conforme a freqüência varia, a potência varia também. Isso

se dá devido às características de reverberação da Célula GTEM que tem um comportamento não linear conforme a variação da freqüência.

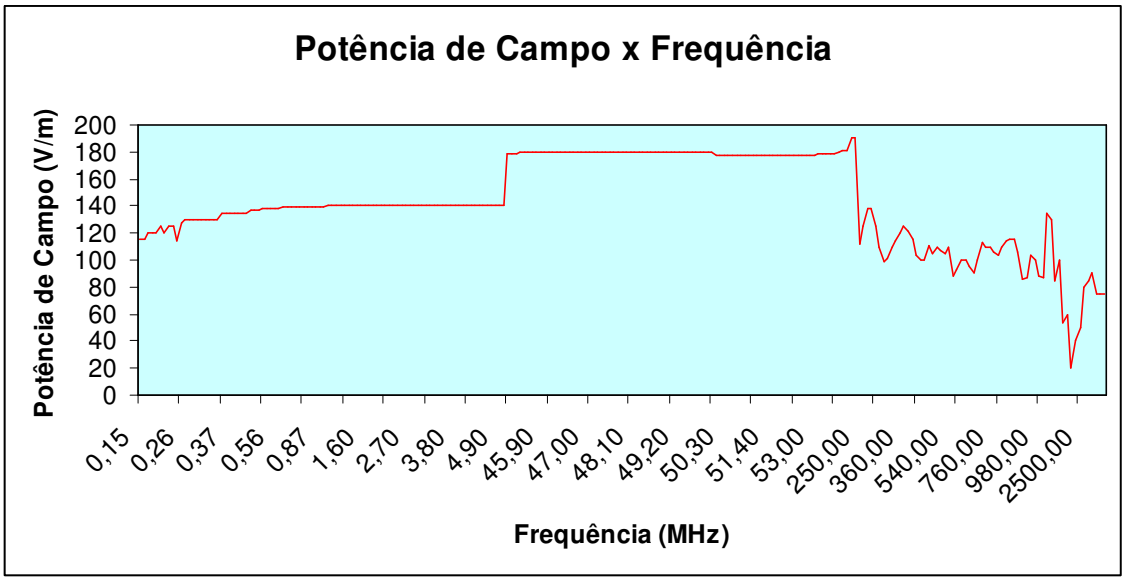


Figura 8.28 – Gráfico da potência de campo versus freqüência aplicada nos testes para ensaios irradiados com a placa sem a caixa de blindagem.

Para esta etapa dos testes, foram utilizadas as mesmas técnicas de detecção de falhas utilizadas pelos ensaios da Parte II, assim como as mesmas configurações das aplicações utilizadas para os testes.

Os seguintes resultados foram obtidos com esta etapa de testes:

Para os ensaios a imunidade irradiada com a placa totalmente descoberta (sem a caixa de blindagem) foi utilizado a potência máxima e freqüência máxima fornecidas pelos equipamentos dos laboratórios do INTI, que nestes ensaios chegaram a 190V/m para a potência de campo dentro da Célula GTEM e de 150kHz até 3GHz para freqüência, que foram aplicados ao DUT e não observou-se nenhuma falha. Ou seja, para a faixa de freqüências de 150kHz a 3GHz e para uma potência de campo elétrico de 20V/m a 190V/m, não se observou nenhuma ocorrência de falhas.

Tal como visto na Parte II dos ensaios, onde o ensaio de EMI irradiado sobre os três componentes da placa demonstrou a imunidade desta para a norma IEC 63.132. Este fato demonstra que a plataforma também atende a mesma norma IEC sendo imune a todo e qualquer tipo de interferência eletromagnética no intervalo de freqüências e potências requeridas pela norma.

8.4 Ensaio de Imunidade para EMI Conduzida

Os testes de ensaios de interferência conduzida, obedecendo a norma IEC 62.132-4 foram realizados com a placa para ensaios conduzidos conforme mostrado na Figura 7.12 e na Figura 7.13.

Para estes testes de ensaios de imunidade a interferência conduzida, foi previsto 16 conectores do tipo SMA (*Subminiature* A) ligados a 16 pontos diferentes da plataforma de forma a abranger o maior número de pontos críticos dos dispositivos em teste, onde conecta-se diretamente um cabo coaxial do amplificador de sinais diretamente a um ou mais conectores previstos na placa conforme ilustrado no diagrama da Figura 8.29 a fim de injetar a interferência controlada conforme as regras da norma. Foram escolhidos os seguintes pontos de injeção:

NA ALIMENTAÇÃO:

- A tensão de **1V2**, que é utilizada para alimentar o *core* (lógica interna) da FPGA;
- A tensão de **2V5**, que é utilizada para alimentar dentre outras funções a função responsável pelo chaveamento dos pinos de i/o;
- A tensão de **3V3**, que é utilizada para alimentar as funções dos bancos de i/o das FPGAs e o restante dos componentes existentes da placa, sendo a tensão de alimentação principal da placa;
- O terra (GND) da placa.

NA FPGA 1:

- D0 – Pino do barramento de dados menos significativo;
- D15 – Pino do barramento de dados mais significativo;
- A0 – Pino do barramento de endereço menos significativo;
- A11 – Pino do barramento de endereço mais significativo;
- TCK – Pino de *clock* da interface de JTAG está ligado à máquina de estados de programação e debug interno ao dispositivo, e tem prioridade máxima de execução.
- TMS – Pino de TMS da interface de JTAG está ligado à máquina de estados de programação e debug interno ao dispositivo, e tem prioridade máxima de execução.

NA FPGA 2:

- D0 – Pino do barramento de dados menos significativo;
- D15 – Pino do barramento de dados mais significativo;
- A0 – Pino do barramento de endereço menos significativo;
- A11 – Pino do barramento de endereço mais significativo;

- TCK – Pino de *clock* da interface de JTAG está ligado à máquina de estados de programação e debug interno ao dispositivo, e tem prioridade máxima de execução.
- TMS – Pino de TMS da interface de JTAG está ligado à máquina de estados de programação e debug interno ao dispositivo, e tem prioridade máxima de execução.

NO MSC1211 (uP8051):

- P3.6 – Pino de I/O ligado diretamente a interface de acesso a memória externa;
- P3.7 – Pino de I/O ligado diretamente a interface de acesso a memória externa.

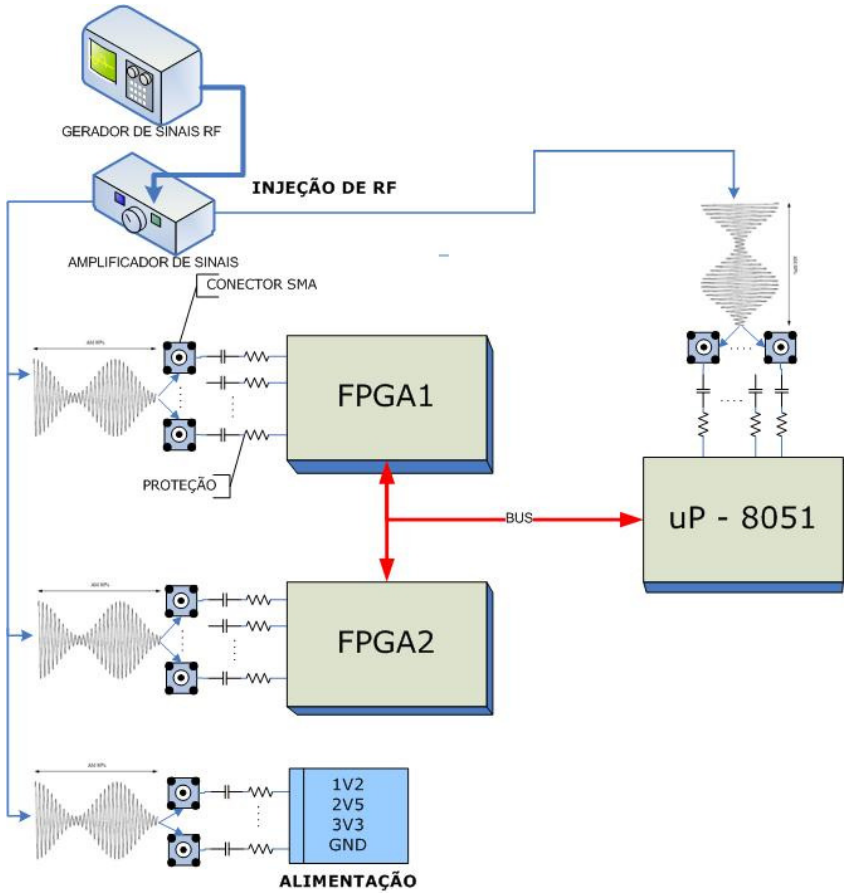


Figura 8.29 - Diagrama de Injeção direta de RF.

Como mostrado no diagrama da Figura 8.29 acima, a injeção de RF ocorre diretamente no pino desejado, como mostrado na Figura 8.31 passando por uma proteção de um resistor em série de 50 ohms para limitar a corrente juntamente com um capacitor de 6,8nF para bloquear qualquer espúrio de tensão DC.

A etapa de ensaios de exposição à EMI conduzido foi realizada conforme a seguinte estrutura de testes, que consiste na utilização dos seguintes equipamentos conforme mostrado na Figura 8.30:

- Gerador de sinais de 1Hz a 3GHz;
- Amplificador de sinais de 1Hz a 3GHz e potência de 300 watts;
- Cabo coaxial com conector SMA para conexão com a placa;
- Osciloscópio para visualização do sinal injetado;
- Medidor de potência injetada.

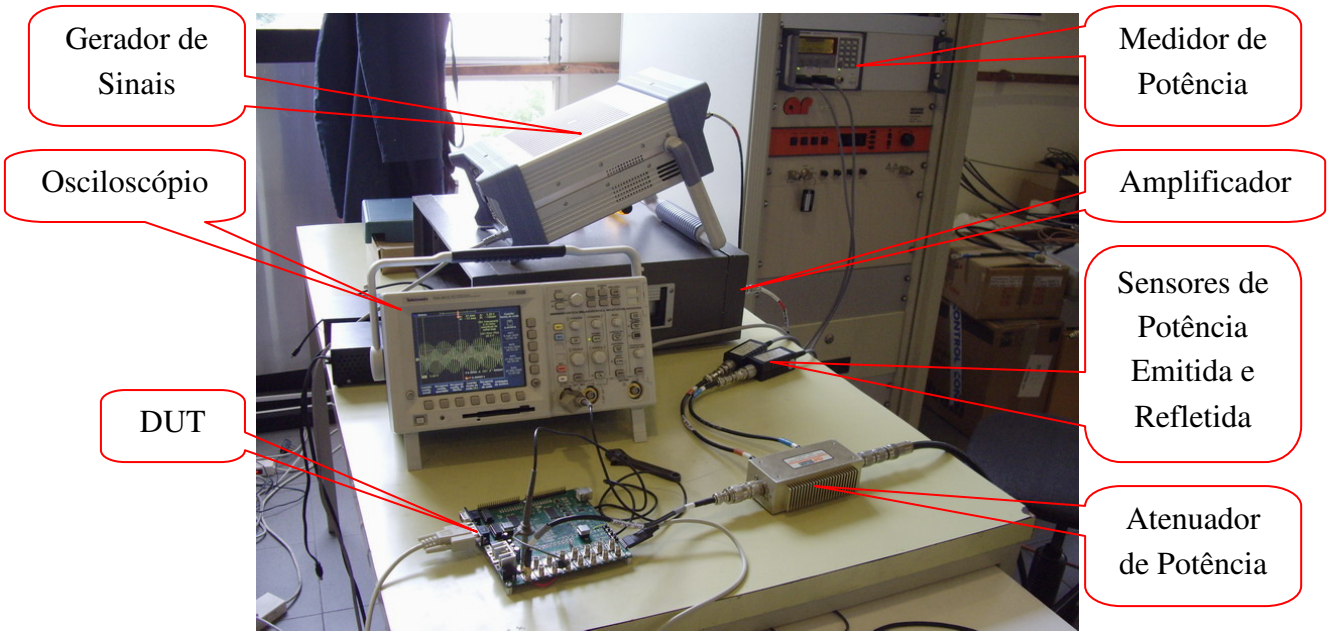


Figura 8.30 – Equipamentos utilizados nos testes de ensaios de interferência conduzida.

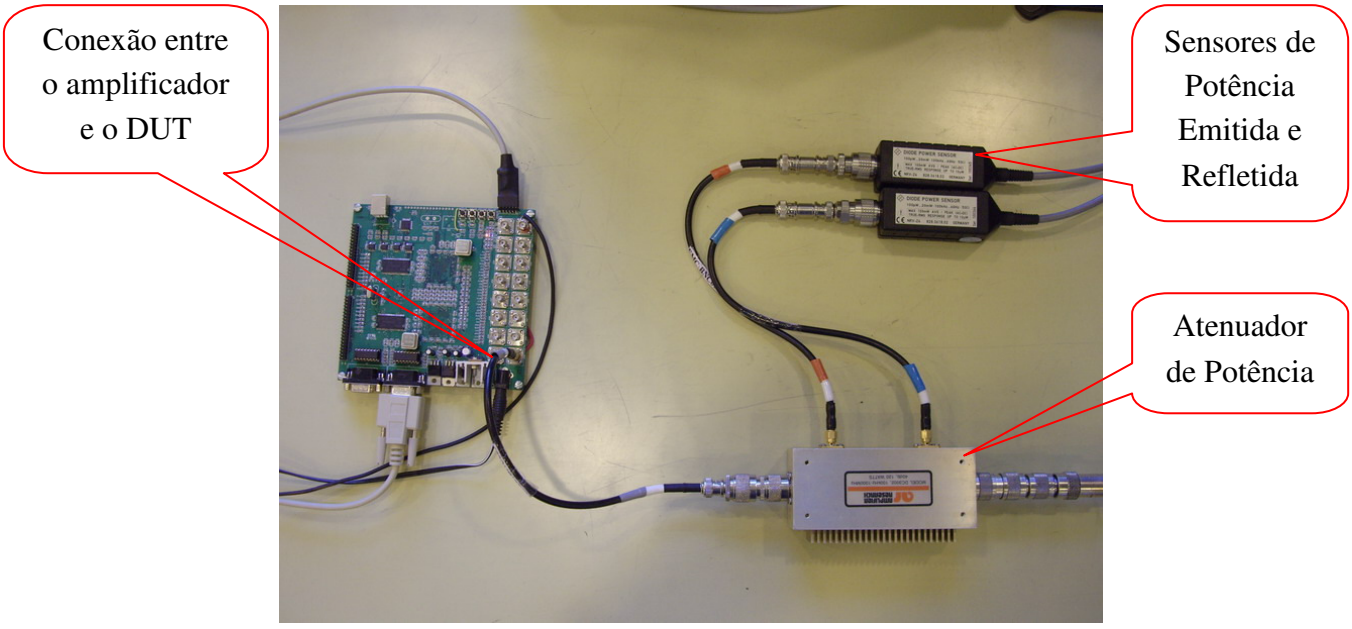


Figura 8.31 – Detalhe da injeção feita diretamente no pino do CI a ser testado.

Nesta etapa de testes, foram utilizados freqüências de 150kHz a 3GHz e uma potência injetada de 15 dBm a 37 dBm que correspondem de 31,62 mW a 5,01 W conforme conversão mostrada na Tabela 8.3.

Conversão de Potência

$$P(dBm) = 10 \times \log\left(\frac{P(W)}{1mW}\right)$$

P(dBm) = Potência em dBm
P(W) = Potência em Watts

dBm	mW	dBm	mW	dBm	W	dBm	W
-20	0,010	0	1,00	20	0,10	40	10,0
-19	0,013	1	1,26	21	0,13	41	12,6
-18	0,016	2	1,58	22	0,16	42	15,8
-17	0,020	3	2,00	23	0,20	43	20,0
-16	0,025	4	2,51	24	0,25	44	25,1
-15	0,032	5	3,16	25	0,32	45	31,6
-14	0,040	6	3,98	26	0,40	46	39,8
-13	0,050	7	5,01	27	0,50	47	50,1
-12	0,063	8	6,31	28	0,63	48	63,1
-11	0,079	9	7,94	29	0,79	49	79,4
-10	0,100	10	10,00	30	1,00	50	100,0
-9	0,126	11	12,59	31	1,26	51	125,9
-8	0,158	12	15,85	32	1,58	52	158,5
-7	0,200	13	19,95	33	2,00	53	199,5
-6	0,251	14	25,12	34	2,51	54	251,2
-5	0,316	15	31,62	35	3,16	55	316,2
-4	0,398	16	39,81	36	3,98	56	398,1
-3	0,501	17	50,12	37	5,01	57	501,2
-2	0,631	18	63,10	38	6,31	58	631,0
-1	0,794	19	79,43	39	7,94	59	794,3
0	1,000	20	100,00	40	10,00	60	1000,0

Tabela 8.3 – Tabela de conversão de dBm para Watts (RF, 2007).

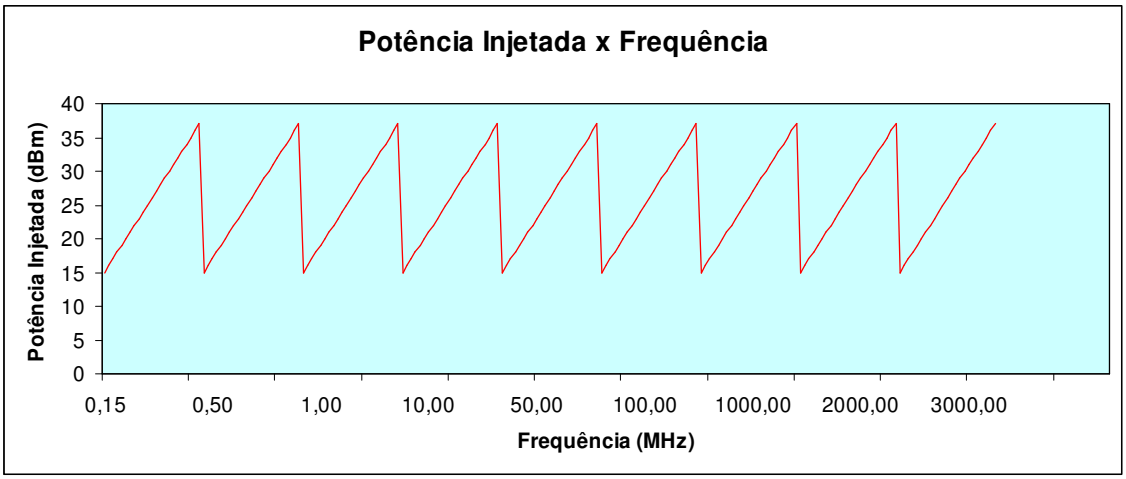


Figura 8.32 – Gráfico da potência injetada versus freqüência aplicada nos testes para ensaios conduzidos segundo a norma IEC 62.132-4.

Conforme o gráfico da Figura 8.32, as freqüências utilizadas foram de 150kHz a 3GHz divididos nos seguintes intervalos conforme mostrado na Tabela 8.4:

FAIXA DE FREQUÊNCIA	INTERVALO	EXEMPLO DE VARIAÇÃO
150kHz a 1MHz	100kHz	250, 350, 450kHz...1MHz
1MHz a 50MHz	1MHz	2,3,4...50MHz
49.100MHz a 49.200MHz	2kHz	49.102, 49.104...49.200MHz
50MHz a 100MHz	10MHz	60, 70, 80.. 100MHz
100MHz a 500MHz	20MHz	120, 140...500MHz
500MHz a 3GHz	50MHz	550, 600, 650...3GHz

Tabela 8.4 - Tabela dos intervalos de freqüência utilizados nos testes para ensaios conduzidos.

Como se observa na Tabela 8.4 nas freqüências próximas a freqüência do *clock*, que é de 49.152MHz, utilizamos um intervalo mais fino de freqüência com o objetivo de procurar uma maior susceptibilidade do sistema por se tratar de uma freqüência crítica para este sistema.

Para esta etapa dos testes, foram utilizadas as mesmas técnicas de detecção de falhas utilizadas pelos ensaios da Parte II e da Parte III, assim como as mesmas configurações das aplicações utilizadas para os testes.

Os seguintes resultados foram obtidos com esta etapa de testes:

Para os ensaios para imunidade conduzida foi utilizado a potência máxima e freqüência máxima sugerida pela norma IEC 62.132-4 que eram de até 5 Watts (37dBm) para a potência de RF injetado e de 150kHz até 3GHz para freqüência, que foram aplicados ao DUT e não observou-se nenhuma falha. Ou seja, para a faixa de freqüências de 150kHz a 3GHz e para uma potência de RF injetado de 15dBm (31,62 mW) a 37dBm (5,01W), não se observou nenhuma ocorrência de falhas.

Com isso na tentativa de se injetar falhas, os equipamentos dos laboratórios do INTI nos permitiram aplicar uma maior potência ao DUT. Chegou-se a aplicar uma potência de 38dBm que corresponde a 6,31 Watts, conforme mostrado na Figura 8.33 capturada pelo osciloscópio (o que pôde ser observado que no pino injetado tinha uma tensão pico a pico de 20 volts), onde com isso obtivemos a queima de uma de nossas placas e mesmo assim não foram observadas falhas, o FPGA simplesmente parou de funcionar no meio da aplicação. Assim observamos que o CI sob teste, neste caso o FPGA 1 estava muito quente e posteriormente constatamos sua queima.

Ainda com o intuito de injetar falha, modificamos uma de nossas aplicações, mais precisamente o algoritmo multiplicador de matrizes. As seguintes modificações foram feitas:

Como a FPGA 1 e a FPGA 2 tem um conjunto de pinos ligados entre si, fizemos a FPGA 1 gerar as matrizes A e B e enviar por um barramento de 8 pinos para a FPGA 2 no qual fazia a multiplicação das duas matrizes A.B e devolvia pelo mesmo

barramento para a FPGA 1 o resultado para verificação do mesmo e envia-lo pela serial para o computador gerenciador. Esta modificação foi feita com o intuito de se ter um barramento externo ao chip, ao nível de placa com maior exposição. Já que os aplicativos eram executados dentro da *Block* RAM do FPGA e então para que pudéssemos injetar diretamente em um dos pinos deste barramento a fim de verificarmos se alguma falha ocorria, sendo ela de dados (algum dado tinha seu valor modificado) ou induzida para dentro do FPGA causando algum outro tipo de falha. Nesta configuração, foram utilizados dois pinos para a injeção no barramento: D0 (bit de dado menos significativo) e D7 (bit de dado mais significativo). Tal como visto na seção 8.3, fica também demonstrada a imunidade desta placa para EMI conduzido segundo a norma IEC 62.132-4, pois mesmo com a injeção direta no "barramento de dados" entre os FPGAs, não ocorreu nenhuma falha, nem de dados e nem induzidos.

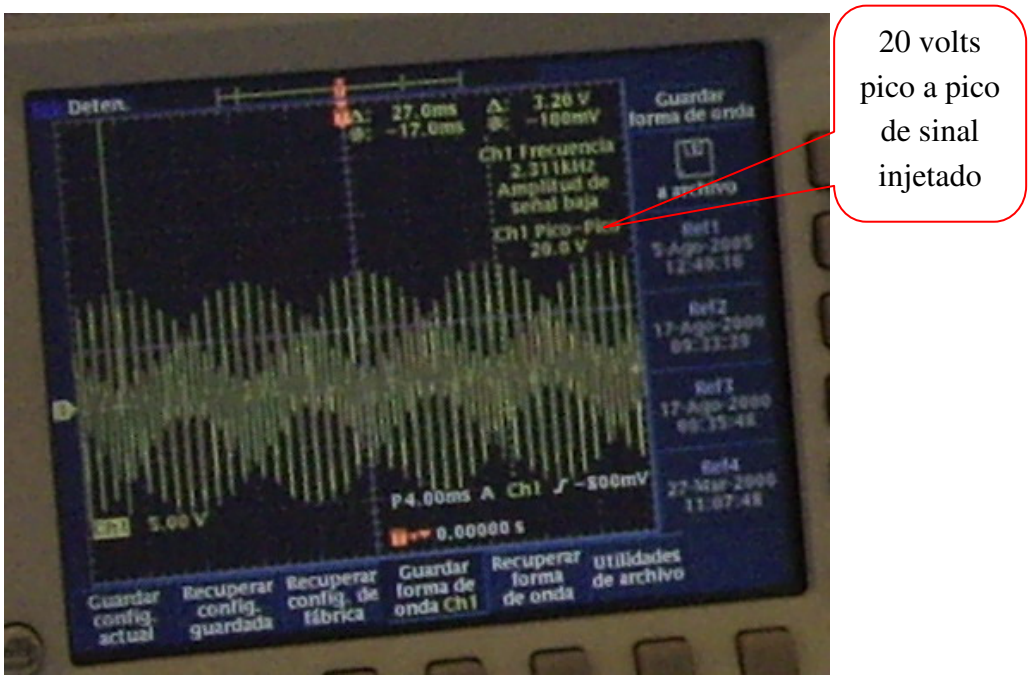


Figura 8.33 – Captura da tela do osciloscópio mostrando o sinal de ruído injetado na placa com uma tensão pico a pico de 20 volts.

8.5 Conclusões dos Ensaios

As conclusões para os ensaios serão explanadas em cinco partes:

1. Nos ensaios realizados na plataforma de testes para ensaios irradiados onde a placa foi encaixada na porta da Célula GTEM utilizando a placa de 10cm x 10cm (Figura 8.10) não foi observada nenhuma falha injetada na placa. Três prováveis motivos levaram a não ocorrência de falhas nesta situação: a potência de campo injetada não foi suficiente para provocarem falhas; a blindagem (planos de VCC e GND) da placa mostra que as trilhas de sinais e o restante dos componentes não

sofreram interferência; e finalmente que o componente testado no caso um FPGA (XC3S200) da família Spartan3 da Xilinx é um componente bastante robusto para as potências dos campos e freqüências utilizados nos testes. Com isso, a placa de testes foi colocada inteiramente dentro da Célula GTEM (Figura 8.11) com o objetivo de se ter uma maior probabilidade de injeção de falhas e validar as técnicas de detecção de falhas propostas por Piccoli (PICCOLI, 2006) em sua dissertação de mestrado que obteve as seguintes conclusões:

- Analisando as tabelas e gráficos dos testes e em todos os casos em que o WDP-IP/SW detectou erro, o WDP-IP/HW também detectou. O WDP-IP/HW foi capaz de detectar vários tipos de erros, tais como erros de assinatura gerada pelo CRC na área de memória da aplicação, e erros no *bitstream* de configuração do FPGA;
- As técnicas utilizando o algoritmo YACCA e WDP-IP+ em software foram capazes de detectar erros ou falhas tanto permanentes quanto transientes. Sendo a maioria deles falhas do tipo transiente (erros detectados pelo CRC e outras falhas) enquanto que um número menor de erros ou falhas do tipo permanente foi detectado (erro no *bitstream* da configuração da FPGA);
- O RTOS uC/OS-II detectou somente erros quando alocava recursos de semáforo no momento em que falhas não identificadas (outras falhas) foram geradas no processo de injeções de falhas;

2. Nos ensaios realizados na plataforma de ensaios conduzidos, onde realizamos também ensaios irradiados utilizando uma Célula GTEM não apenas para tamanhos de placa de 10cm x 10cm como a utilizada nos testes realizados por Piccoli, essa Célula (Figura 7.5) tem uma tamanho consideravelmente maior, podendo-se assim ser utilizada para ensaios irradiados com diversos tamanhos de placa. Em uma primeira etapa de testes, a placa de testes foi acoplada em uma caixa metálica (Figura 8.21) com a finalidade de se aproximar das características indicadas na norma IEC 62.132-2 que define que somente os componentes a serem testados devem sofrer a interferência e que o restante fica protegido. É o que foi feito quando colocamos a placa na caixa metálica, somente a parte de cima da placa foi exposta, o restante ficou protegido pela caixa (Gaiola de Faraday). Após os ensaios realizados observamos que nenhuma falha ocorreu o que confirma que a blindagem da caixa metálica funcionou perfeitamente, não deixando espúrios induzidos provocarem falhas. Com isso temos três prováveis motivos que levaram a não ocorrência de falhas nesta situação: a potência de campo injetada não foi suficiente para provocarem falhas (os equipamentos estavam em seu limite de operação); a blindagem

(planos de VCC e GND) da placa mostra que as trilhas de sinais e o restante dos componentes não sofreram interferência; e finalmente que os componentes testados que no caso foram duas FPGAs (XC3S500E) da família Spartan3E da Xilinx e um microcontrolador MSC1211 (uP8051) da Texas Instruments se mostraram componentes bastante robustos para as potências de campos e freqüências utilizados nos testes;

3. Após os ensaios realizados com a placa utilizando a caixa metálica de blindagem, retiramos a caixa metálica e colocamos a placa de testes inteiramente dentro da Célula GTEM (Figura 8.26) com o objetivo de se ter uma maior probabilidade de injeção de falhas. Não observamos a ocorrência de falhas pelos mesmos motivos descritos anteriormente;

4. Na etapa de ensaios conduzidos, foram utilizados todos os pontos de injeção previstos na placa que são 16 no total. Também foram utilizadas a potência e freqüência máxima descritas na norma IEC 62.132-4 e não observamos a ocorrência de falhas, o que mostra a grande robustez dos componentes utilizados para esta faixa de potência aplicada e freqüência utilizada e que os planos de VCC e GND ajudam a dissipar os distúrbios inseridos na placa;

5. Como conclusão final dos ensaios, observou-se uma grande robustez da placa para testes conduzidos (e que também foi validada para testes irradiados) de forma a atender plenamente a norma IEC 62.132-2 e IEC 62.132-4. Este resultado é muito importante para que evidencie a utilização desta placa para o desenvolvimento de sistemas robustos segundo a mesma norma. Em outras palavras, SoCs prototipados nesta plataforma e ensaiados para EMI segundo a norma devem funcionar sem falhas. Os componentes, assim como a própria placa (trilhas, conectores, chaves, planos de VCC e GND etc.) demonstraram robustez a EMI estando assim validados, ficando a critério de projetistas utilizarem estes componentes e regras de projeto para desenvolvimento de sistemas robustos a EMI.

9. CONCLUSÕES FINAIS

Este trabalho teve como principais objetivos: demonstrar a aplicabilidade da plataforma para avaliar SoCs em desenvolvimento e verificar a robustez da plataforma em função de ensaios. Para cumprir esses objetivos, foram necessários realizar as etapas abaixo:

1) Pesquisar, implementar e avaliar o comportamento de plataformas comerciais de teste que não seguem nenhuma norma para testes de compatibilidade eletromagnética, que foram realizados nos laboratórios do INTI em Buenos Aires;
2) Desenvolver uma plataforma de testes de acordo com a norma IEC 62.132 para teste de EMC para circuitos integrados;
3) Implementar aplicativos de acordo com as técnicas propostas;
4) Realizar os testes de EMI irradiada e conduzida que também foram realizados nos laboratórios do INTI em Buenos Aires na Argentina.

Assim, as principais conclusões alcançadas neste trabalho são relacionadas a seguir:

- Os testes realizados utilizando EMI irradiada e conduzida seguiram normas da IEC. Assim, foi possível reproduzir as interferências encontradas em ambientes reais para verificar como a plataforma e as técnicas de detecção de falhas se comportaram sob o efeito da EMI;
- Nos testes preliminares feitos em uma plataforma comercial da Xilinx podemos avaliar o comportamento da plataforma de testes e seus componentes onde obtivemos a conclusão de que uma minoria de falhas (9,33%) ocorre na configuração (*bitstream*) do FPGA e que 38,33% das falhas ocorreram na lógica (falhas no fluxo de controle do processador, falha de dados e outras falhas) o restante são falhas não classificadas tais como paralisação total do sistema. Nestes ensaios também podemos avaliar a capacidade das três técnicas de detecção de falhas utilizadas onde conclui-se que: que os erros de configuração do FPGA, ou seja erros no *bitstream* são 100% detectados pela técnica de SADR (*readback*), mostrando-se uma técnica poderosa em se tratando de erros de configuração de hardware; a técnica de CFCSS tem uma taxa de detecção de no máximo 33% (obtidos nestes testes) o que para aplicações mais críticas não se trata de uma técnica muito robusta de detecção. Porém para sistemas onde 33% de cobertura de falhas seja um bom número, trata-se de uma ótima técnica, pois se trata de uma técnica de baixo custo, de baixo *overhead* e de baixa complexidade de

implementação; a técnica de ECI tem a capacidade de detecção de falhas de 100% das falhas que ocorrem nos espaços não utilizados de memória, ou seja, esta técnica varia sua taxa de detecção de acordo com o espaço livre de memória e a capacidade de endereçamento do processador. Neste caso o processador Microblaze tem a capacidade de endereçamento de 4Gbytes, a memória contida na plataforma possui 1Mbyte sendo que 94,2% dela não foi utilizada, então sua capacidade aproximada de detecção de 24%. Dado que é uma técnica que não agrega custo algum ao projeto (tanto de software, de hardware quanto de *overhead*) é uma boa opção como detecção de falhas de fluxo de controle de sistemas operando em ambientes de interferência eletromagnética;

- Como conclusão geral dos ensaios, observou-se uma grande robustez da placa para testes conduzidos (e que também foi validada para testes irradiados) de forma a atender plenamente a norma IEC 62.132-2 e IEC 62.132-4. Este resultado é muito importante para que evidencie a utilização desta placa para o desenvolvimento de sistemas robustos segundo a mesma norma. Em outras palavras, SoCs prototipados nesta plataforma e ensaiados para EMI segundo a norma devem funcionar sem falhas. Os componentes, assim como a própria placa (trilhas, conectores, switches, planos de VCC e GND etc.) demonstraram robustez a EMI estando assim validados, ficando a critério de projetistas utilizarem estes componentes e regras de projeto para desenvolvimento de sistemas robustos à EMI.

9.1 Trabalhos Futuros

Devido ao avanço acelerado da tecnologia, e o desenvolvimento de novos sistemas eletrônicos, faz-se necessário criar novos e mais completos ambientes de testes para sua validação.

Em relação à plataforma de testes são sugeridas as seguintes implementações futuras:

- Inserir o controle de alimentação utilizando reguladores de tensão programáveis, com o objetivo de submeter os circuitos integrados aos seus limites quanto a sua alimentação;
- Inserir um oscilador programável, onde se possa variar a freqüência de *clock* do sistema, submetendo os circuitos integrados a falhas de referência de *clock*;
- Construção de uma Célula GTEM de testes em um tamanho reduzido e de baixo custo onde se possa alcançar maiores potências de campo

elétrico, podendo-se chegar a valores do tipo 1000V/m com o intuito de aumentar a injeção de falhas;

- Inclusão de circuitos analógicos na plataforma tais como amplificadores operacionais e conversores A/D e D/A para avaliar seu comportamento e sua robustez quando expostos a um ambiente de EMI;
- Utilização de FPGAs de outras famílias e de outros fabricantes com o objetivo de se fazer uma comparação entre o comportamento das mesmas;
- Realização de ensaios utilizando variação de temperatura;
- Como outros exemplos de aplicação da plataforma proposta, esperam-se implementar, testar e avaliar sistemas de redundância de hardware e de software, sistemas de processamento paralelo e distribuído, por exemplo.

10. REFERÊNCIAS BIBLIOGRÁFICAS

(ABRAHAM, 1995) - J.A. Abraham, "Challenges in Fault Detection," *Proc. 25th Ann. Int'l Symp. Fault-Tolerant Computing, IEEE CS Press, Los Alamitos*, Calif., pp. 96-114, **1995**.

(ALGOTRONIX, 1989) - Algotronix, Ltd. CAL 1024 Datasheet. Edinburgh, Scotland, **1989**.

(ALKHALIFA, 1997) - Z. Alkhalifa, V.s.s. Nair. Design of a portable control-flow checking technique. *Proceedings of the 2nd High-Assurance Systems Engineering Workshop*, pp. 120-23, **1997**.

(ALKHALIFA, 1999) - Z. Alkhalifa, V.s.s. Nair, N. Krishnamurthy, J.a. Abraham. Design and Evaluation of System-Level Checks for On-Line Control Flow Error Detection. *IEEE Transactions on Parallel and distributed systems*, volume 10, número 6. pp. 627-641, junho **1999**.

(ALTERA, 2001) – Altera. ACEX 1K, Embedded Programmable Logic Device Family Data Sheet, version 4.1, March, **2001**.

(ALTERA, 2007) - ALTERA. Avalon Bus Specification. Reference Manual. Capturado em: *http://www.altera.com/products/software/products/sopc/avalon/nio-avalon_features.html*, Janeiro **2007**.

(AMD, 1990) - AMD Inc. Mach Devices High Density EE Programmable Logic Data Book, **1990**.

(ANDERSON, 1981) - Anderson, T.; Lee, P. A. Fault tolerance -principles and practice. *Englewood Cliffs, Prentice-Hall*, **1981**.

(ARLAT, 1989) - J. Arlat, Y. Crouzet, and J.C. Laprie, "Fault Injection for Dependability Validation of Fault-Tolerant Computer Systems," *Proc. 19th Ann. Int'l Symp. Fault-Tolerant Computing, IEEE CS Press, Los Alamitos*, Calif., pp. 348-355, **1989**.

(ARM, 2007) - ARM Corp. AMBA 2.0 Specification. Capturado em: *http://www.arm.com/products/solutions/AMBAHomePage.html* , Janeiro **2007**.

(ATMEL, 1999) - ATMEL Corporation. Data Acquisition Systems Using Cache Logic FPGAs, *Application Note*, **1999**.

(AVIZIENIS, 1984) - Avizienis, A.; Kelly, J. P. Fault tolerance by design diversity - Concepts and experiments. *Computer*, New York, 17(8):67-80, Aug. **1984**.

(AVIZIENIS, 1998) - Avizienis, A. Infraestructure-based design of fault-tolerant systems. In: Proceedings of the *IFIP International Workshop on Dependable Computing and its Applications. DCIA 98*, Johannesburg, South Africa, January 12-14, **1998**. p. 51-69.

(BAKER, 1991) - S. Baker. Lattice fields FPGA. *Eleci Fog Times.* no. 645, p. 1, June, **1991**.

(BARDELL, 1987) - Bardell, P., H. Built in test for VLSI: pseudorandom techniques. *New York*. **1987**.

(BARRETT, 1993) - P A Barrett and N A Speirs. "Towards an integrated approach to I fault tolerance in Delta-4", In: *Distrib. Syst. Engng* 1(1993)59-66, 15 February **1993**.

(BARROS, 2004) – Barros, J., D.; Vargas, F.; Santos, M.; Teixeira, I. & Teixeira, J. Modeling and simulation of time domain faults in digital systems *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International,* **2004**, 5-10.

(BENINI, 2001) - Benini, L.; De Micheli,G. Powering Networks on Chip. In: *International Symposium on System Synthesis*, pp. 33 –38, **2001**.

(BENINI, 2002) - Benini, L.; De Micheli,G. Networks on chips: a new SoC paradigm. *Computer*, v. 35(1), pp. 70-78, January **2002**.

(BERGAMASCHI, 2000) - Bergamaschi, R. A.; Lee, W. R. Designing systens-on-chip using cores. In: *Design Automation Conference (DAC'00)*, pp. 420-425, **2000**.

(BERGAMASCHI, 2001) - Bergamaschi, R. A.; Bhattacharya, S.; Wagner, R.; Fellenz, C.; Muhlada, M.; White, F.; Daveau, J. M.; Lee, W. R. Automating the design of SoCs using cores. *IEEE Design & Test of Computers*, v. 18(5), Sep.-Oct. pp. 32 -45, **2001**.

(BERGAMASCHI, 2002) - Bergamaschi, R.A.; Cohn, J. The A to Z of SoCs. In: *International Conference on Computer Aided Design (ICCAD)*, pp. 791-798, **2002**.

(BEZERRA, 2001) - Bezerra, E. A.; Vargas, F.; Gough, M. P. Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques. *Journal of Electronic Testing: Theory and Applications – JETTA. Kluwer Academic Publishers*, New York, USA. Vol. 17, May 1st, pp. 163-174, **2001**.

(BIENERT, 2004) - Bienert, R.W. EMC in Wireless Devices – *TUV Rheiland of North America,* **2004.**

(BIRKNER, 1991) - J. Birkner, A. Chan, H. T. Chua, A. Chao, K. Gordon, B. Kleinrnan, P. Kolze, and R. Wong. A very high-speed field programmable gate array using metal-to-metal anti-fuse programmable elements. In: *New Hardware Product Introduction at Custom Integrated Circuits Conference, CICC'91*, **1991**.

(BOLZANI, 2005) - BOLZANI, Letícia Maria Veiras. Explorando uma Solução Híbrida: Hardware + Software para a Detecção de Falhas em Systems-on-Chip (SoCs) 2005. Dissertação de Mestrado – *Programa de Pós-Graduação em Engenharia Elétrica da Universidade Católica do Rio Grande do Sul* (PUCRS), Porto Alegre, **2005.**

(BOSCH, 2004) – Bosch, R. *Electromagnetic Compatibility – Definition and Need*. S. India - Suganya, **2004**.

(BOSTOCK, 1996) - G. Bostock. FPGA Combines Multiple Serial Interfaces and Logic. *Butterworth-Heinemann*, Oxford, **1996**.

(CARREIRA, 1995) - J. Carreira, H. Madeira, and J.G. Silva, "Xception: Software Fault Injection and Monitoring in Processor Functional Units," *Proc. Fifth Ann. IEEE Int'l Working Conf. Dependable Computing for Critical Applications, IEEE CS Press, Los Alamitos*, Calif., pp. 135-149, **1995**.

(CHEN, 1978) - Chen, L.; Avizienis, A. N-version programming: a fault tolerance approach to reliability of software operation. In: *Annual International Symposium on Fault-Tolerant Computing*, 8, 1978. Proceedings. New York, IEEE, **1978**. p. 3-9.

(CONCURRENT LOGIC, 1991) - Concurrent Logic. CFA6006 Field-Programmable Gate Array Data Sheet, **1991**.

(CREVELING, 1956) - Creveling, C.J. Increasing the reliability of electronic equipment by the use of redundant circuits. *Proceedings of the IRE*. New York, 44(4):509- 515, abr. **1956**.

(CROUCH, 1999) - Crouch, A. L. Design-for-Test for Digital IC's and Embedded Core Systems. *Prentice Hall* PTR,349p. ISBN: 0-13-084827-1, **1999**.

(DE CASTRO, 2005) - De Castro, F.C.C. Apostila de Eletrônica Digital, **2005**. Disponível em: *http://www.ee.pucrs.br/~decastro/pdf/ED_C4.pdf* . Acesso em: Dezembro de 2006.

(EL GAMAL, 1989) - El Gamal, A.; Greene, J.; Reyneri, J.; Rogoyski, E.; El-Ayat, K. & Mohsen, A. An architecture for electrically configurable gate arrays. *Solid-State Circuits, IEEE Journal* of, **1989**, 24, 394-398.

(FCC, 2007) – Site oficial da *Federal Communications Commission*. Disponível em: http://www.fcc.gov/. Acesso em: Janeiro de **2007**.

(GAJSKI, 2000) - Gajski, D., Zhu,J., Dömer,R., Gerstlauer,A., Zhao,S. SpecC: Specification Language and Methodology. *Kluwer Academic Publishers, Norwell*, MA, 336 p, **2000**.

(GIRARD, 2002) - Girard, P. Survey of low-power testing of VLSI circuits. *Design & Test of Computers, IEEE,* **2002***, 19*, 80-90.

(GOLOUBEVA, 2003) - O. Goloubeva, M. Rebaudengo M, M. S. Reorda, M. Violante. Soft-error Detection Using Control Flow Assertions. *Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, **2003**.

(GUERRIER, 2000) - Guerrier. P.; Greiner. A. A generic architecture for on-chip acket-switched interconections. In: *Design Automation and Test in Europe (DATE'00)*, pp. 250- 256, **2000**.

(GUNNETLO, 1989) - O. Gunnetlo, J. Karlsson, and J. Tonn, "Evaluation of Error Detection Schemes Using Fault Injection by Heavy-ion Radiation," *Proc. 19th Ann. Int'l Symp. Fault-Tolerant Computing, IEEE CS Press, Los Alamitos, Calif.*, pp. 340-347, **1989**.

(GUPTA, 1997) - Gupta, R. K., Zorian, Y. Introducing Core-Based System Design. *IEEE Design & Test of Computers*, v.14(4), pp. 15-25, Oct.-Dec. **1997**.

(HAN, 1995) - S. Han, K.G. Shin, and H.A. Rosenberg, "Doctor: An Integrated Software Fault-Injection Environment for Distributed Real-Time Systems," *Proc. Second Annual IEEE Int'l Computer Performance and Dependability Symp., IEEE CS Press, Los Alamitos*, Calif., pp. 204-213, **1995**.

(HSUEH - 1997) – M.C. Hsueh, T. K. Tsai, R. K. Iyer. "Fault Injection Techniques and Tools". In: *IEEE Computer Society Los Alamitos,* CA, USA. Vol. 30 , pp: 75 - 82 , ISSN:0018-9162, **1997**.

(HWANG, 1993) - Hwang, K. Advanced Computer Architecture: Parallelism, Scalability, Programmability. New York: *McGraw-Hill*, 771p, **1993**.

(IBM, 2007) - IBM. The CoreConnectTM Bus Architecture. Capturado em: http://www.ibm.com/chips/ products/coreconnect, Janeiro **2007**.

(IEC, 2004) - INTERNATIONAL ELECTROTECHNICAL COMMISSION – International Standard IEC 62.132 Normative, **2004**. Disponível em : *http://www.iec.ch*. Acesso em 20 de novembro de **2005**.

(IYER, 1996) - D. Tang and R.K. Iyer, "Experimental Analysis of Computer System Dependability," in: *Fault-Tolerant Computer System Design*, D.K. Pradhan, ed., *Prentice-Hall* Prof. Tech. Ref., Upper Saddle River, N.J., pp. 282-392, **1996**.

(IYER, 2002) - R. K. Iyer, Z. Kalbarczyk. Hardware and Software Error Detection. 2002. Disponível: http://www.crhc.uiuc.edu/~kalbar/MotorolaCourse/HW&SW_ ErrorDetection.pdf. Acesso em: Dezembro de 2006.

(JANSCH, 1997) - Jansch-Porto, I. E. S; Weber, T. S. Recuperação em Sistemas Distribuídos. In: *XVI Jornada de Atualização em Informática, XVII Congresso da SBC*, Brasília, 2-8 agosto de **1997**. anais. pp. 263-310.

(JUNEIDI, 2001) - Juneidi, Z.; Torki, K.; Martinez, S.; Nicolescu, G.; Courtois, B.; Jerraya, A. Global Modeling and Simulation of System-on-Chip embedding MEMS devices. In*: International Conference on ASIC*, pp. 666 –669, **2001**.

(KANAWATI, 1992) - G.A. Kanawati, N.A. Kanawati, and J.A. Abraham, "FERRARI: A Tool for the Validation of System Dependability Properties," *Proc. 22nd Ann. Int'l Symp. Fault-Tolerant Computing, IEEE CS Press, Los Alamitos*, Calif., pp. 336-344, **1992**.

(KANAWATI, 1996) - G. A. Kanawati, V. S. S. Nair, N. Krishnamurthy, J. A. Abraham. Evaluation of integrated system-level checks for on-line error detection. *Proceedings of IEEE International Computer Performance and denpendability Symposium*, p.p. 292-301, **1996**.

(KARLSSON, 1995) - J. Karlsson, J. Arlat, and G. Leber, "Application of Three Physical Fault Injection Techniques to the Experimental Assessment of the MARS Architecture," *Proc. Fifth Ann. IEEE Int'l Working Conf. Dependable Computing for Critical Applications, IEEE CS Press, Los Alamitos, Calif.*, pp. 150-161, **1995**.

(KATHAIL, 2002) - Kathail, V.; Aditya, S.; Schreiber, R.; Ramakrishna Rau, B.; Cronquist, D. & Sivaraman, M. PICO: automatically designing custom computers. *Computer, 35*, 39-47, **2002**.

(KIMMEL, 1995) – Kimmel, W.D. Gerke, D.D. Electromagnetic Compatibility in Medical Equipment: a guide for designers and installers. USA: *IEEE and Interpharm Press, Inc*. p. 1-29. ISBN 0-935184-80-5 and 0-7803-1160-4, **1995**.

(KRAUSS, 1999) - Krauss, J. D. e Fleisch, D.A , Electromagnetics with aplications , *Mc Graw-Hill*, **1999.**

(KUMAR, 2002) - Kumar, S.; Jantsch, A.; Soininen, J. P.; Fonsell, M. A Network on Chip Architecture and Design Methodology. In: *Computer Society Annual Symposium on VLSI (ISVLSI'02)*, pp. 105-112, **2002**.

(KUMAR, 2003) - Kumar, S. On Packet Switched Network for Chip Communication. In: *Axel Jantsch and Hannu Tenhunen, editors, Networks on Chip, chapter 5, Kluwer Academic Publishers*, pp. 85-106, **2003**.

(KYNETT, 1988) - V. N. Kynett, A. Baker, M. L. Fandrich, G. P. Hoekstra, O. Jungroth, J. A. Kreifels, S. Wells, and M. D. Winston. An In-System Reprogrammable 32Kx8 CMOS Flash Memory. *IEEE Journal of Solid-State Circuits, JSSC*, vol. 23. no. 5, October **1988**.

(LALA, 1990) - P. K. Lala, "Digital System Design Using Programmable Logic Devices," *Prentice Hall*, **1990**.

(LAPRIE, 1985) - LAPRIE, J. C. Dependable computing and fault-tolerance: concepts and terminology. In: *Annual International Symposium on Fault Tolerant Computing*, 15. Ann Arbor, jun. 19-21, 1985. Proceedings. New York, IEEE, p. 2-11, **1985**.

(LAPRIE, 1995) - Laprie, J. Dependable computing and fault-tolerance: concepts and terminology. *Fault-Tolerant Computing, 1995, 'Highlights from Twenty-Five Years'. Twenty-Fifth International Symposium on,* **1995**.

(LAPRIE, 1998) - LAPRIE, J. C.; Dependability: From concepts to limits. In: Proceedings of the IFIP *International Workshop on Dependable Computing and its Applications. DCIA 98*, Johannesburg, South Africa, January 12-14. p. 108-126, **1998**.

(LOPES, 2005) - LOPES, Danniel Cavalcante. Estimação da Robustez de Sistemas Eletrônicos Via Injeção de Falhas por Interferência Eletromagnética. 2005. Dissertação de Mestrado – *Programa de Pós-Graduação em Engenharia Elétrica da Universidade Católica do Rio* Grande *do Sul* (PUCRS), Porto Alegre, **2005**.

(LYONS, 1962) - LYONS, R.E.; Vanderkulk, W. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*. New York, 6(3): 200-209, abr. **1962**.

(MADISETTI, 1997) - MADISETTI, V. K., Shen L. Interface Design for Core-Based Systems. *IEEE Design & Test of Computers*, v. 14(4), pp. 45-51, October-December **1997**.

(MAGNUS, 2001) – MAGNUS, Élio Freitas. Desenvolvimento de uma Ferramenta para Ensaios de EMI Conduzida de Baixo Custo 2001. Dissertação de Mestrado – *Programa de Pós-Graduação em Engenharia Elétrica da Universidade Católica do Rio Grande do Sul* (PUCRS), Porto Alegre, **2001.**

(MARPLE, 1992) - D. Marple and L. Cooke. An MPGA compatible FPGA architecture. In: *ACM First International Workshop on Field Programmable Gate Array, FPGA 92*, pp. 39-44, February, **1992**.

(MARTIN, 2001) - Martin, G.; Chang, H. System on Chip Design. In: International Symposium on Integrated Circuits, Devices & Systems (ISIC'01), Tutorial 2, **2001**.

(MCCLUSKEY, 2002) - McCluskey, E. J.; OH, N.; Shirvani, P. P. Control-Flow Checking by Software Signatures. *IEEE Transactions on Reliability.* Vol. 51, No. 2, pp. 111-122, March **2002**.

(MICHELI, 1996) - G. de Micheli and M. Sami. Hardware/Software Co-Design. *Kluwer Academic Publishers*, **1996**.

(MIREMADI, 1995) - Miremadi, G.; Torin, J. Evaluating Processor-Behavior and Three Error-Detection Mechanisms Using Physical Fault-Injection. *IEEE Transactions on Reliability*. Vol. 44, No. 3, pp. 441-454, September **1995**.

(MUROGA, 1991) - H. Muroga, H. Murata, Y. Sacki, T. Hibi, Y. Ohash. T. Nuguchi, and T. Nishimura. A large scale FPGA with 10K core cells with CMOS 0.8 μm 3-layered metal process. In: *Custom Integrated Circuits Conference, CICC'91*, pp. 6.4.1-6.4.4, May, **1991**.

(NEWMANN, 1956) - Von Newmann, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In: *Automata Studies,* Shannon & McCarthy eds. Princeton Univ. Press, p. 43-98, **1956**.

(OST, 2004) - OST, Luciano Copello. Redes Intra-Chip Parametrizáveis com Interface Padrão para Síntese em Hardware. 2004. Dissertação de Mestrado – *Programa de Pós-Graduação em Ciência da Computação da Universidade Católica do Rio Grande do Sul* (PUCRS), Porto Alegre, **2004**.

(OTT, 1988) – Ott, H. W. Noise Reduction Techniques in Electronic Systems. USA: *John Wiley & Sons*, Inc. p. 21-23, p. 145-146. ISBN 0-471-85068-3, **1988**.

(PAUL, 2001) - Paul C.R. Introduction to Electromagnetic Compatibility. USA: *John Wiley & Sons*, Inc. p. 1-75, 667 - 668. ISBN 0-471-54927-4, **1992**.

(PERRY, 1998) - Perry, Douglas. VHDL 3rd Edition, *McGraw-Hil*l, New York, US, 493 p, **1998**.

(PICCOLI, 2006) - PICCOLI, Leonardo Bisch. Soluções Híbridas de Hardware/Software para a Detecção de Erros em Systems-on-Chip (SoC) de Tempo Real. 2006. Dissertação de Mestrado – *Programa de Pós-Graduação em Engenharia Elétrica da Universidade Católica do Rio Grande do Sul* (PUCRS), Porto Alegre, **2006**.

(PLESSEY, 1989) - Plessey Semiconductor. ERA60100 preliminary data sheet, **1989**.

(PRADHAN, 1995) - J.A. Clark and D.K. Pradhan, "Fault Injection: A Method for Validating Computing-System Dependability," *Computer*, pp. 47-56, June **1995**.

(PRADHAN, 1996) - Pradhan, D., K. In: Fault-Tolerant Computer System Design. Editora: Prentice Hall, **1996**.

(REIS - 2000) - Reis, R. A. L.; Concepção de Circuitos Integrados. Instituto de Informática da UFRGS. *Editora Sagra Luzzatto*, 252 p, **2000**.

(RF, 2007) – Medidas de Sinais Modulados. Capturado em: *http://rf.com.br/catalog2006/0405.pdf*, Janeiro **2007**.

(RIBEIRO, 2004) - Ribeiro, J.A.J , Propagação de Ondas Eletromagnéticas - Princípios e Aplicações. *Editora Érica* , **2004.**

(RODRIGUEZ, 2004) - Rodriguez-Irago, M.; Barros Jr. D.; Vargas, F.; Santos, M. B.; Teixeira, I. C.; Teixeira, J. P. Modeling of Power Supply Transients for EMI Compliance in Digital Systems. *10th IEEE International Mixed-Signal Testing Workshop - IMSTW'04*. Portland-Oregon, USA, June 23-25, **2004**.

(ROSE, 1993) - Rose, J.; El Gamal, A. & Sangiovanni-Vincentelli, A. Architecture of field-programmable gate arrays. *Proceedings of the IEEE*, 81, 1013-1029, **1993**.

(RUNNER, 2000) - Runner, S.; Sanaka, V.; Yu, E. Building an Infrastructure for IP Reuse. EE Times, May 15, **2000**. Capturado em: *www.eetimes.com*, dezembro **2006**.

(SEMATECH, 2002) - International Sematech. International Technology Roadmap for Semiconductors Update, *Capturado em: http://public.itrs.net*, dezembro **2006**.

(SIA, 1999) - Semiconductor Industry Association, The National Technology Roadmap for Semiconductors 1999. Capturado em: *http://public.itrs.net/*, dezembro **2006**.

(SOUZA, 2006) – SOUZA, Antônio Marcos. Eletromagnetismo, Notas de Aula. Disponível em: http://cict.inatel.br/cict/acervo%20publico/Material_Did%C3%A1tico_P%C3%BAblico/Eletrica/P05/T507/01%20-%20not%20cap%201%20ao%206.pdf. . Acesso em: Janeiro de **2007**.

(STROUD, 2002) – Stroud, E. C. A designer's guide to built-in self-test. Boston : Kluwer Academic, **2002**.

(SYSTEMC, 2003) - SystemC v2.0.1 Language Reference Manual, Capturado em: *http://www.systemc.org/projects/systemc/document/SystemC_v201_LRM/*, **2003**.

(THOMAS, 1991) - Thomas, D., Moorby, P. The Verilog Hardware Description Language. *Kluwer Academic Publishers*, p 376, **1991**.

(TOROK, 2001) - TOROK, Delfim Luiz. Projeto Visando a Prototipação do Protocolo de Acesso ao Meio em Redes Ethernet. 2001. Dissertação de Mestrado – *Programa de Pós-Graduação em Ciência da Computação da Universidade Católica do Rio Grande do Sul* (PUCRS), Porto Alegre, **2001**.

(TSAI, 1996) - T.K. Tsai and R.K. Iyer, "An Approach to Benchmarking of Fault-Tolerant Commercial Systems," *Proc. 26th Ann. Int'l Symp. Fault-Tolerant Computing, IEEE CS Press, Los Alamitos*, Calif., pp. 314-323, **1996**.

(VARGAS, 2006) - Vargas, F.; Benfica, J.; Farina, A.; Bezerra, E.; Gatti, E.; Garcia, L.; Lupi, D.; Hernandez, F.. Observing SRAM-Based FPGA Robustness in EMI-Exposed Environments. *7th IEEE Latin-American Test Workshop - LATW 2006*, Buenos Aires, Argentina. Proceedings, v. 01. p. 201-206, **2006**.

(VARGAS, 2007) - VARGAS, F., BENFICA. J., PICCOLI, L., GATTI, E., GARCIA, L., LUPI, D., HERNANDEZ, F.. Configurable Platform for SoC Electromagnetic Immunity Measurement. *IBERCHIP 2007*, Lima, Perú, Março de **2007**.

(VARGAS, 2007a) - VARGAS, F., BENFICA. J., PICCOLI, L., GATTI, E., GARCIA, L., LUPI, D., HERNANDEZ, F.. SoC Prototyping Environment for Electromagnetic Immunity Measurements. *EMC WORKSHOP 2007*, Paris, França, Junho de **2007**.

(VARGAS, 2007b) – Vargas, F., Rocha, C.A., Alecrim, A., Becker, C.A.. Embedded Signature Insertion Based on Profiling Deployed Software Technique. . *IBERCHIP 2007*, Lima, Perú, Março de **2007**.

(WESTON, 2001) - Weston, D.A. Electromagnetic Compatibility; Principles and Applications 2. ed. New York, USA: *Marcel Dekker*, Inc, p.1-17. ISBN 0-8247-8889-3, **2001**.

(WOLF, 2004) - Wolf, W. Embedded is the new paradigm(s). *Computer,37*, 99-101, **2004**.

(WONG, 1989) - S. C. Wong et al. A 5000-gate CMOS EPLD with multiple logic and interconnect arrays. In: *Proc. 1989 Custom Integrated Circuits Conference, CICC´89*, pp. 5.8.1- 5.8.4. May **1989**.

(XILINX, 1999) - Xilinx Corporation, Xilinx Data Book. *Manual de fabricante*, **1999**. Disponível em: *http://www.xilinx.com/partinfo/databook.htm* . Acesso em: Dezembro de **2006**.

(XILINX, 2000) - Xilinx Corporation. FastFLASH XC9500XV High-Performance, Low-Power CPLD Family. *Advance Product Information*, DS049 (v1.1) February, **2000**.

(XILINX, 2005) - MicroBlaze Soft Processor Core. Capturado em: *http://www.xilinx.com/ise/embedded /mb_ref_guide.pdf*, Janeiro **2007**.

(XILINX, 2007) - Designing Custom OPB Slave Peripherals for MicroBlaze. Capturado em: *http://www.xilinx.com/ipcenter/processor_central/microblaze/doc/opb_tutorial.pdf*, Janeiro **2007**.

(ZEFERINO, 2003) - Zeferino, C. A. and Susin, A. A. "SoCIN: A Parametric and Scalable Network-on-Chip", In: *Symposium on Integrated Circuits and Systems (SBCCI'2003)*, pp. 169-174, **2003**.

ZEFERINOa, 2003) - Zeferino, C. A. "Redes-em-chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho". Tese de doutorado, *Universidade Federal do Rio Grande do Sul - UFRGS/PPGC*, Porto Alegre, RS, **2003**.

(ZEIDMAN, 2001) - Bob Zeidman. Introduction to CPLD and FPGA Design. President The Chalkboard Network, **2001**.

(ZEVZIKOVAS, 2004) – ZEVZIKOVAS, Marcos. Efeitos da Interferência Eletromagnética Conduzida em Equipamentos Eletromédicos no Ambiente Hospitalar 2004. Dissertação de Mestrado – *Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Estadual de Campinas* (UNICAMP), São Paulo, **2004.**

ANEXOS

I. PUBLICAÇÕES

SoC Prototyping Environment for Electromagnetic Immunity Measurements

F. Vargas*, J. Benfica*, L. Piccoli*, E. Gatti**, L. Garcia**, D. Lupi**, F. Hernandez***

* Electrical Engineering Dept. Catholic University – PUCRS. Av. Ipiranga, 6681. 90619-900 Porto Alegre – Brazil.
** INTI. Av. Gral. Paz 5445 - San Martín. (1650) Buenos Aires – Argentina.
*** URSEC. Av. Uruguay, 988. Montevideo – Uruguay.
vargas@computer.org

Abstract: *We present a configurable standard environment[1] for electromagnetic (EM) immunity measurement of prototype system-on-chip (SoC). The environment is composed of a board compliant with the 62.132-2 and 62.132-4 IEC Std Parts and is conceived for radiated and conducted measurements, respectively. The SoC under test is settled around two types of ICs: two FPGAs and a microcontroller. Practical experiments have been carried out. The obtained results demonstrate the effectiveness of this tool to estimate the behavior of embedded systems based on these types of components when operating in EM environment.*

The Environment

The board contains two Xilinx Spartan 500E FPGAs, a Texas 8051-like microcontroller, 16MBytes of SDRAM, and 8MBytes of serial Flash memory, among other glue logic required for communication with the test host computer (see Fig.1 for details). With this infrastructure, multiple embedded microprocessors like MicroBlaze[2] and PowerPC 601 running uCLinux or uCOS-II[3] can be prototyped. Additionally to the hardware parts, several implementations of VHDL-described embedded intellectual property (IP) cores and C-code programs can also have their immunity response measured and compared to each other in order to leverage the final dependability level for the SoC on the design.

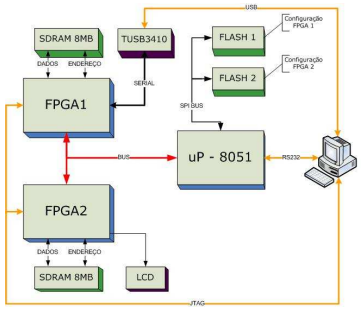


Fig. 1. Basic blocks of the environment.

Fig. 2 presents a board photograph. In this figure, side ***(a)*** contains the components under test, i.e., the parts whose EM measurements have to be done; whereas side ***(b)*** contains the remainder of the logic (processor bus, memories, crystals, connectors and external environment communication-support ICs, among other devices). Fig. 3 depicts the shielding box for radiated testing. Fig. 3a shows the board side under test.

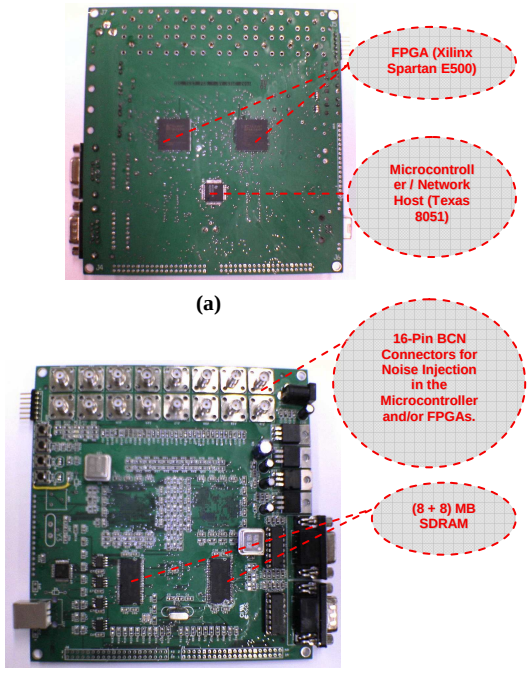
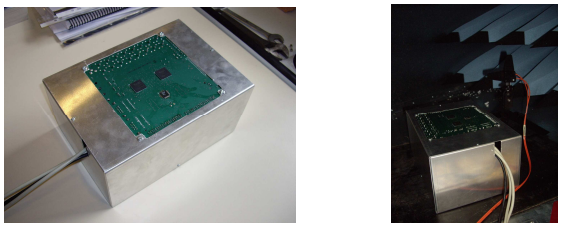


Fig. 2. Standard 6-Layer Board for Immunity Measurement. Views: **(a)** Top; **(b)** Bottom.



(a) **(b)**

Fig. 3. Shielding box for radiated test: **(a)** General view; **(b)** inside the GTEM Cell.

1 *This work is partially supported by CNPq.*

2 *MicroBlaze™ is a true 32-bit soft RISC processor optimized for use in Xilinx's FPGA architectures. The processor's main memory interface conforms to the IBM CoreConnect specification for the On-Chip Peripheral Bus (OPB).*

3 *MicroC/OS-II has been certified to RTCA DO-178B Level A for use in avionics systems where failure could result in catastrophic loss of the aircraft, and approved for use in FDA Class III medical devices where failure could result in loss of life for the patient or clinician.*

Configurable Platform for SoC Electromagnetic Immunity Measurement

F. Vargas*, J. Benfica*, L. Piccoli*, E. Gatti**, L. Garcia**, D. Lupi**, F. Hernandez***

* Electrical Engineering Dept. Catholic University – PUCRS. Av. Ipiranga, 6681. 90619-900 Porto Alegre – Brazil.
** INTI. Av. Gral. Paz 5445 - San Martín. (1650) Buenos Aires – Argentina.
*** URSEC. Av. Uruguay, 988. Montevideo – Uruguay.
vargas@computer.org

ABSTRACT

We present a configurable standard platform[1] for electromagnetic (EM) immunity measurement of prototype system-on-chip (SoC). The environment is composed of two boards: one is based on the 62.132-2 IEC Std and is applied for radiated measurements. The other is compliant with the 62.132-4 IEC Std Part and is conceived for conducted measurements. The SoC under test is settled around two types of ICs: an FPGA and a microcontroller. Practical experiments have been carried out. The obtained results demonstrate the effectiveness of this tool to estimate the behavior of embedded systems based on these two components when operating in EM environment.

1. INTRODUCTION

The electromagnetic (EM) environment in which electronic systems have to operate is becoming increasingly hostile while dependence on electronics is widespread and increasing. The need for assurance that application upsets due to the EM environment will not occur is fundamental to acceptance of systems as fit for purpose [1-3]. Thus, it is important to understand how future technologies impact on next-generation complex systems-on-chips (SOCs). Note that although the reduction of supply voltages (at least for the core part) rises the hope for less EM emission (conducted and radiated), this benefit is immediately compensated by [4]: *(a)* a drastically increased number of simultaneously switching transistors per die, combined with faster switching edges due to increasing clock rates. Thus, increasing the total RF noise that can affect embedded functional blocks inside the die itself, as well as affecting other dies or ICs placed nearby it; and *(b)* the reduced power supply voltage minimizes the noise margins in which the IC was designed to operate. Thus, rendering the IC critical embedded functional blocks more sensitive to EMI [5,6].

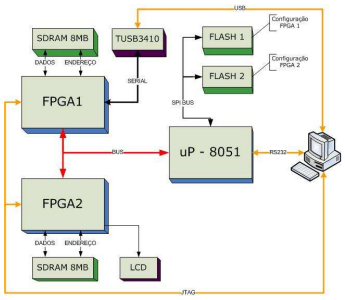
It is in this scenario that we propose a configurable prototyping platform to estimate SoC immunity to EM noise. This platform is based on two custom complementary boards and specific international standards and procedures. The remainder of this paper is divided as follows: *Section 2* presents briefly the platform and its two boards. *Section 3* summarizes a case-study implemented to demonstrate the usefulness of the platform. Finally, *Section 4* presents the final conclusions of this work.

[1] *This work is partially supported by CNPq.*

2. THE PLATFORM

The developed platform is based on two specific complementary boards. The first one is devoted to *radiate noise immunity test* in a Gigahertz Transverse Electromagnetic Cell (GTEM Cell) according to the IEC 62.132-2 Std [7]. The second board is dedicated to the *conducted RF noise test* and is compliant with the IEC 62.132-4 Std [7]. These boards contain a Xilinx Spartan E500 FPGA, a Texas 8051-like microcontroller, 16MBytes of SDRAM, and 8MBytes of serial Flash memory, among other glue logic required for communication with the computer test host. (Fig.1 presents the basic block diagrams of these boards.) With this infrastructure, multiple embedded microprocessors like MicroBlaze[2] and PowerPC 601 running uCLinux or uCOS-II[3] can be prototyped. Additionally to the hardware parts, several implementations of VHDL-described embedded intellectual property (IP) cores and C-code programs can also have their immunity response measured and compared to each other in order to leverage the final dependability level for the SoC on the design.

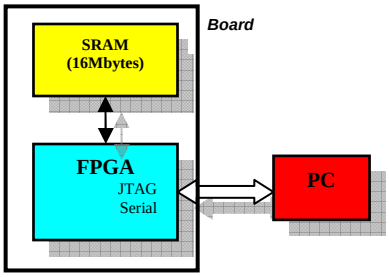
Figs. 2 and 3 present a photograph of both sides of these boards. In these figures, side ***(a)*** contains the components under test, i.e., the parts whose EM measurements have to be done; whereas side ***(b)*** contains the remainder of the logic (processor bus, memories, crystals, connectors and external environment communication-support ICs, among other devices).



(a)

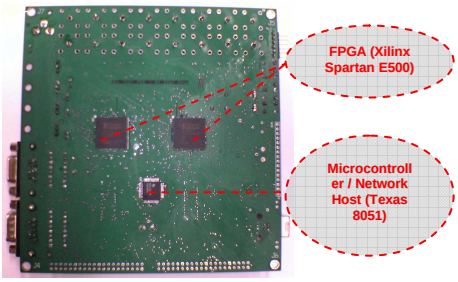
[2] *MicroBlaze™ is a true 32-bit soft RISC processor optimized for use in Xilinx's FPGA architectures. The processor's main memory interface conforms to the IBM CoreConnect specification for the On-Chip Peripheral Bus (OPB).*

[3] *MicroC/OS-II has been certified to RTCA DO-178B Level A for use in avionics systems where failure could result in catastrophic loss of the aircraft, and approved for use in FDA Class III medical devices where failure could result in loss of life for the patient or clinician.*

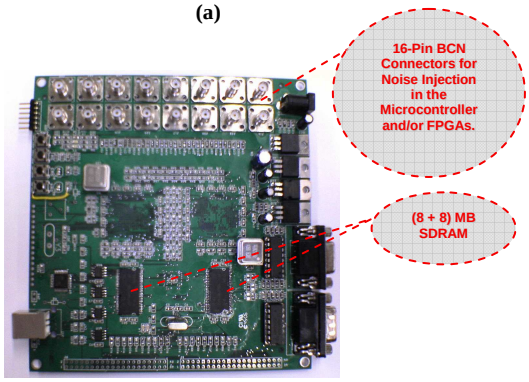


(b)

Fig. 1. Basic blocks of the platform. Views: **(a)** Board for RF Conducted Immunity Measurement; **(b)** Board for Radiate Immunity Measurement.

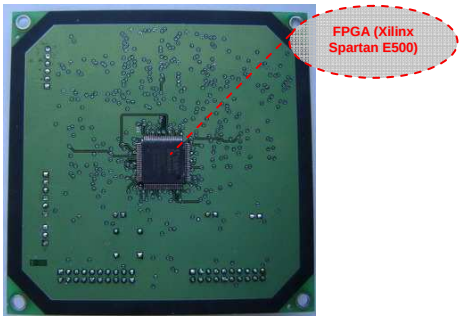


(a)

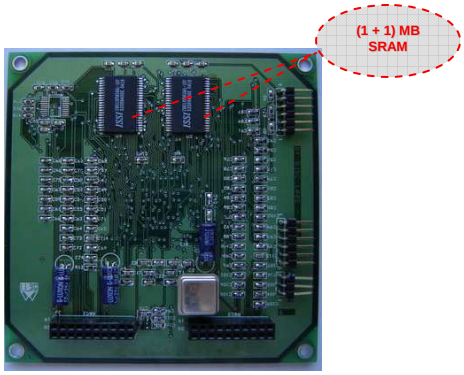


(b)

Fig. 2. Standard 6-Layer Board for RF Conducted Immunity Measurement. Views: **(a)** Top; **(b)** Bottom.



(a)



(b)

Fig. 3. Standard 4-Layer Board for Radiated Immunity Measurement. Views: **(a)** Top; **(b)** Botton.

3. FINAL CONCLUSIONS

We presented a configurable standard platform for electromagnetic (EM) immunity measurement of prototype system-on-chip (SoC). The environment is composed of two boards: one is based on the 62.132-2 IEC Std and is applied for radiated measurements. The other is compliant with the 62.132-4 IEC Std Part and is conceived for conducted measurements. The SoC under test is settled around two types of ICs: an FPGA and a microcontroller. Practical experiments have been carried out. The obtained results demonstrate that this tool is very effective to estimate the behavior of SoCs based on these two components when operating in EM environment.

For additional information on how to obtain detailed documentation about this platform, or how to retrieve the board's layout, please contact the first author by email.

REFERENCES

[1] Bernardi, P.; Veiras Bolzani; L. M.; Rebaudengo, M.; Sonza Reorda, M.; Vargas, F. L.; Violante, M. *A New Hybrid Fault Detection Technique for Systems-on-a-Chip*. IEEE Transactions on Computers, Feb. 2006, Vol. 55, No. 2. pp.185-198.

[2] Bolzani, L.; Rebaudengo, M.; Sonza Reorda, M.; Vargas, F.; Violante, M. *Hybrid Soft Error Detection by Means of Infrastructure IP Cores*. 10th IEEE International On-Line Testing Symposium (IOLTS'04), 2004.

[3] Bezerra, E. A.; Vargas, F.; Gough, M. P. *Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques*. Journal of Electronic Testing: Theory and Applications – JETTA. Kluwer Academic Publishers, New York, USA. Vol. 17, May 1st, 2001, pp. 163-174.

[4] Steinecke, T. *Design-In for EMC on CMOS Large-Scale Integrated Circuits*. International Symposium on Electromagnetic Compatibility – EMC'2001. Vol. 2 , 2001. pp. 910 –915.

[5] Perez, R. *Signal Integrity Issues in ASIC and FPGA Design*. International Symposium on Electromagnetic Compatibility – EMC'1997. 1997. pp. 334 -339.

[6] Whyman, N. L.; Dawson, J. F. *Modelling RF Interference Effects in Integrated Circuits*. International Symposium on Electromagnetic Compatibility – EMC'2001. Vol.2, 2001. pp. 1203 –1208.

[7] *International Electrotechnical Commission* - www.iec.ch

Observing SRAM-Based FPGA Robustness in EMI-Exposed Environments [1]

F. Vargas[*], J. Benfica[*], A. Farina[*], E. Bezerra[*],
E. Gatti[**], L. Garcia[**], D. Lupi[**], F. Hernandez[***]
vargas@computer.org, lupi@inti.gov.ar, fhernandez@ursec.gub.uy

[*] Electrical Engineering Dept. Catholic University – PUCRS. Av. Ipiranga, 6681. 90619-900 Porto Alegre – Brazil.
[**] INTI. Av. Gral. Paz 5445 - San Martín. (1650) Buenos Aires – Argentina.
[***] URSEC. Av. Uruguay, 988. Montevideo – Uruguay.

Abstract

This paper aims at observing the behavior of SRAM-based Field Programmable Gate Arrays (FPGAs) as operating in EMI-exposed environments. The case-study was composed by a commercial board containing an FPGA and SRAM memories configured with a microprocessor embedded soft core. Three traditional techniques used in the field of on-line testing of digital circuits were used to monitor the FPGA behavior and grade the SEU types between faults affecting logic or faults affecting configuration memory elements. The first technique was based on the periodical readback of the FPGA hardware configuration bitstream. The other two techniques were focused on the detection of microprocessor control-flow faults by using software signatures and capturing instructions. The electromagnetic environment was generated by a GigaHertz Transverse Electromagnetic (GTEM) cell according to the Std. Normative IEC-62132. In the following, the test setup, fault-detection mechanisms and results are detailed and discussed.

Index Terms: Electromagnetic Interference (EMI); Single-Event Upset (SEU); SRAM-type FPGA.

1. Introduction

The electromagnetic (EM) environment in which electronic systems have to operate is becoming increasingly hostile while dependence on electronics is widespread and increasing. The need for assurance that application upsets due to the EM environment will not occur is fundamental to acceptance of systems as fit for purpose [1-3]. Thus, it is important to understand how future technologies impact on next-generation complex systems-on-chips (SOCs). Note that although the reduction of supply voltages (at least for the core part) rises the hope for less EM emission (conducted and radiated), this benefit is immediately compensated by: *(a)* a drastically increased number of simultaneously switching transistors per die, combined with faster switching edges due to increasing clock rates. Thus, increasing the total RF noise that can affect embedded functional blocks inside the die itself, as well as affecting other dies or ICs placed nearby it; and *(b)* the reduced power supply voltage minimizes the noise margins in which the IC was designed to operate. Thus, rendering the IC critical embedded functional blocks more sensitive to EMI.

Considering reconfigurable devices, these introduce to the hardware (HW) level the flexibility for system modifications provided at the software (SW) level. In the case of embedded applications, two important advantages of having the whole electronic system, including the HW and the SW parts, implemented using only HW components (reconfigurable HW) can be: the *economy in area* usage and *board complexity*; and the increased processing speed with *lower clock rates*. Additionally, SRAM-type FPGA components present the intrinsic advantage of being possible to be easily reconfigured *on-the-field*, as response to the occurrence of a system failure, or simply because the HW component is being adapted to different application profiles.

The *drawback* of using SRAM-type devices is that these components are very *sensitive* to transient faults (SEUs) [9]. Depending on the application and the (noisy) operating environment, the minimum reliability level cannot be ensured. As a consequence, embedded system designers are frequently prevented from using this type of components. Instead, EPROM- or ROM-type FPGA components are commonly taken.

Considering the above exposed the present work deals with observing the behavior of SRAM-based FPGAs as operating in EMI-exposed environments. To do so, it was used a commercial board containing an FPGA and SRAM memories configured with a microprocessor embedded soft core. To monitor the FPGA behavior and grade the SEU types between faults affecting *logic* or faults affecting *configuration* memory elements, it has been implemented some traditional techniques used in the field of on-line testing of digital circuits. The first technique, namely "*Signature Analysis-Driven Refresh*", was proposed by Bezerra *et*

[1] *This work is partially supported by CNPq and Red PUCARA-CYTED.*

al. [4]. It is based on the periodical readback of the FPGA HW configuration bitstream, which is compacted by a linear feedback shift register (LFSR) operating in the parallel signature generator (PSG) mode. Then, the compacted signature is compared with a golden one, previously determined and stored into the system. The second and third techniques, *Control-Flow Checking by Software Signatures – CFCSS* and *Error Capturing Instructions – ECI* are based on the McCluskey's [5] and Mireamdi's [6] approaches, respectively, and deal with performing processor on-line control-flow fault detection. The EM environment was generated by using a GigaHertz Transverse Electromagnetic (GTEM) cell according to the Std. Normative IEC 62132[8].

The remaining of this paper is organized as follows: *Section 2* introduces the basic concepts and mechanisms by which EMI affects electronics. *Section 3* describes briefly the fault-detection techniques used to observe the FPGA behavior. *Section 4* describes the system under test (SUT) and the test setup used for practical experiments. *Section 5* compiles the practical experiments results. Finally, *Section 5* summarizes the main conclusions of this work.

2. EMI Effects on Electronics

In the presence of electromagnetic (EM) incident field (produced for example by a mobile phone transmitting antenna), cables connected to and tracks routed on the surface of PCBs or dies behave as receiving antennas capturing disturbances. When the EM-induced currents reach the input pins or internal latches of ICs, they may often produce system malfunctioning [1-3].

EM incident field can also couple with power lines, Vdd and Gnd, resulting in voltage fluctuations that can increase the critical path propagation delay. Thus changing the logic gates settling and hold times. In this case, errors due to EMI come from delay failures along with critical paths at the board or IC level. These errors are propagated to the system outputs or are latched in some memory element. For instance, when the radiated EMI (Noise Source) couples with ground lines and substrate material, it generates voltage fluctuations and current spikes. *These fluctuations are added to the simultaneous switching noise, also known as ground bounce* and may provoke system failure. Ground bounce is commonly present in large ICs when their logic gates change output states during clock transition. During this phenomenon, the local ground voltage is elevated from system ground, causing the IC to have a different ground than the rest of the system. If the ground bounces large enough, logic thresholds can be exceeded, and then, causing false logic transitions to be propagated to the IC outputs or latched in some memory element.

Also, when an output bounces below ground, a negative voltage is fed to whatever the output is driving. If this negative voltage is large enough, transient currents in excess can be generated, which may damage the IC power distribution lines.

3. Fault Detection Techniques

This section describes briefly the techniques used to "probe" the FPGA faults under EM environment.

3.1. Signature Analysis-Driven Refresh (SADR)

The method presented here [4] makes use of a Linear Feedback Shift Register (LFSR) operating in two different modes: as a *Pseudo-Random Pattern Generator* (PRPG) for determining the mean-time between two FPGA configuration bitstream readback operations and as a *Parallel Signature Generator* (PSG) for signature generation and analysis purpose. The signature analysis method is used to determine whether the FPGA configuration bitstream refresh operation is necessary. Fig. 2 summarizes the main ideas behind the proposed approach, which assumes the availability of at least two FPGAs in the system (or an FPGA and a microprocessor). Each one of the FPGAs runs the following two processes:

a) PRPG/PSG. This process operates in two different ways: ***i)*** in the pseudo-random pattern generator mode, when the LFSR reaches a given value, this process signals the moment when a readback operation should be started; and ***ii)*** in the parallel signature generator mode, to compact the FPGA configuration bitstream into a unique signature.

b) RDB&R. The readback and refresh process is responsible to execute the readback and refresh operations.

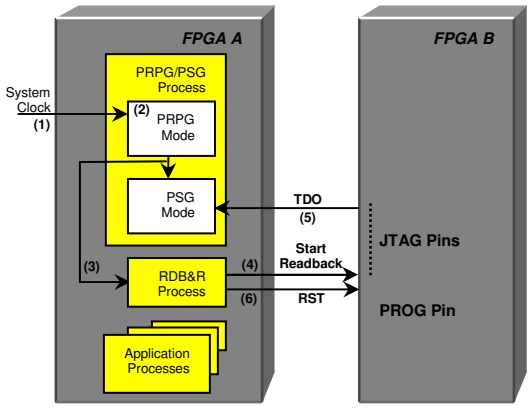


Fig. 1. Signature Analysis-Driven Refresh approach.

With the purpose of illustrating as the approach works, it is described hereafter as the PRPG/PSG and RDB&R processes operate from the point of view of

FPGA A (Fig. 2). Similar reasoning should be applied for FPGA B.

The PRPG is clocked during the time the PRPG/PSG process is operating under the pseudo-random pattern generator mode. When the LFSR reaches a predefined trigger value, it signals the moment when the FPGA B configuration bitstream readback operation should be started. Then, the RDB&R process executes the readback of FPGA B, whereas the PRPG/PSG process switches to the parallel signature generator mode to compact the configuration bitstream read from FPGA B into a unique signature. At the end, if the generated signature differs from the golden one stored into FPGA A, then this FPGA refreshes FPGA B. Refresh is achieved by the RDB&R process resetting the FPGA B PROG pin, which leads to the FPGA being reconfigured. Thus, preventing transient error occurrences from affecting the system functioning.

This approach is very effective for applications where periods of downtime and data loss are not allowable because the bitstream configuration refresh operation is performed *only* if a mismatched signal is reported as result of the signatures comparison.

The base clock used by the PRPG utilizes an internal FPGA clock generator and a frequency divider. By defining the parameters of the frequency divider, it is possible to setup the time step by which the PRPG reaches the predefined trigger value, thus defining the mean-time by which the FPGA configuration bitstream readback operation should be executed.

Note that there is a probability of *aliasing*, that is, of masking eventual errors in the bitstream to be compacted during the signature generation process, which is inversely proportional to the length of the PSG implemented. In other words, the probability of *aliasing* is given by: $(2^{k-r}-1)*(2^{k}-1)^{-1}$

where:

k: is the sequence length, given in bits (that is, the length of the FPGA configuration bitstream whose signature is to be generated);

r: is the LFSR length (that is, the polynomial degree, or the number of flip-flops required for its implementation).

At this moment the input clock is switched from the slow (divided) clock to the system clock, used by the RDB&R process to compact the configuration bitstream and generate the signature.

When the readback is concluded, the PRPG/PSG process in FPGA A uses the calculated signature to compare to the on-chip stored, pre-defined one. If the test fails, then a "start refresh" signal is sent to FPGA B in order to "cleanup" possible errors.

3.2. Control-Flow Fault Detection

Two techniques to detect control-flow faults have been embedded in the application code run by the microprocessor mapped into the FPGA and used in conjunction with the SADR approach. The first technique was proposed by McCluskey *et al.* and is named *Control Flow-Checking by Software Signature (CFCSS)* [5]. The second technique is named Error Capturing Instructions (ECI) and was presented by Miremadi *et al.* [6]. They are briefly commented below.

a) **CFCSS**: briefly describing, it is based on the identification of basic blocks in the workload code and in the addition of two types of signatures to these blocks [5]. The first one is added at the *compilation time*, while the second is computed at *runtime*. The approach consists basically of executing instructions of comparison between the two signatures every time the processor enters a basic block. By doing so, faults affecting control flow are detected with error latency dependent on the size of the basic block and on the position where they happen inside the basic block. To summarize, the approach consists of six steps:

1) *Divide* the program into *basic blocks*. A basic block is a minimal set of ordered instructions in which its execution begins from the first instruction and terminates at the last instruction. There is no branching instruction in a basic block except possibly for the last one. A basic block terminates at either an instruction branching to another basic block or an instruction receiving transfer of control flow (CF) from two or more places in the program.
2) *Construct* a *graph* for the program according to the instructions flow (each node represents a basic block). Note that a program can be represented by a program-graph, P, where bri,j are not necessarily explicit branch instructions; they also represent fall-through execution paths, jumps, subroutine calls, and returns.
3) *Arbitrarily assign* a *signature* for *each node* (*compilation time*).
4) *Compute* the *signature difference* between the source and the destiny blocks.
5) *Compute* the *new signature* for each node (*execution time*).
6) *Compare* both *signatures*.

b) **ECI**: This technique is based on the insertion of *trap instructions* in the unused area of the memory (including the not used areas in the program and data spaces) [6]. In other words, ECIs are inserted in the main memory locations that *are not used by the CPU during normal execution*. Thus, the execution of an ECI is an indication that a *control flow error has occurred*.

The task of an ECI is to initiate a *recovery process*. Instructions like "*reset*", "*nop*" and "*goto*" are typical examples that can be executed in this situation.

4. Test Setup

The experiment described below is based on the international standard normative IEC 62132 (2004) [8], which rules an EM incident field equal to 10V/m, in a frequency range hanging from 80MHz to 1GHz, with an 80% of modulation amplitude, and with a frequency step increment in the order of 1%. Additionally, it was used a special EM field equal to 60V/m in order to

perform system stress test. Runtime measurements of the system operating in EM field were performed in a GTEM with the usual associated equipment (signal generator, RF power amplifier, isotropic field sensor, and directional coupler) as shown in Fig. 2.

The desired EM environment was defined by combining: (***a***) the EM field incident value, Ø (generally 10V/m or another value defined the user/application); (***b***) the modulated signal frequency, F_m, (in the range: 80MHz-1GHz); (***c***) and the frequency modulation depth, D, (in general 80%).

To measure the EM field inside the GTEM cell, it was used an *isotropic field sensor* near the SUT, which provided real-time measurements through a serial connection to the Personal Computer (*Host*). During test session, this PC also monitored the SUT through another serial connection.

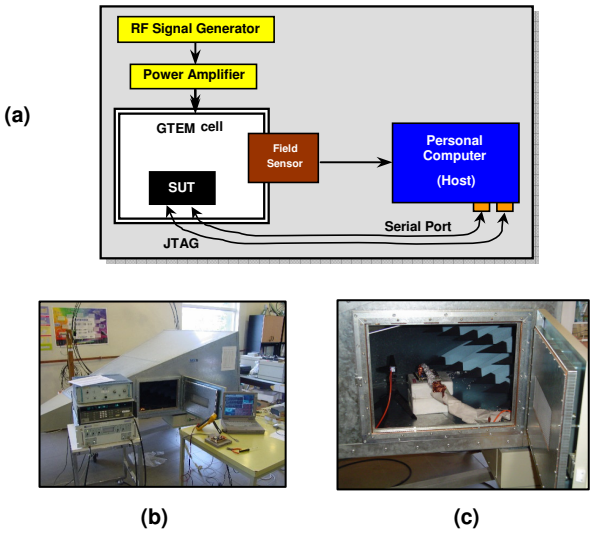


Fig. 2. EMI test setup. (**a**) and (**b**): General scheme and equipments; (**c**): SUT inside the GTEM cell.

For the purpose of observing the FPGA behavior under EM field, the SUT was arranged as follows:

a) It was used a commercial board containing a Xilinx FPGA[2] on one side and two SRAM memories[3] on the other side of the board. This board assumed the hole of FPGA A, as described in Section 3.1.

b) The MicroBlaze (v4.00a) embedded soft core[4], from Xilinx, was downloaded to FPGA A; whereas the application code and data were downloaded to the SRAM memories;

c) With the purpose of minimizing test complexity, test setup was implemented with only one FPGA, i.e., FPGA A. In this scenario, FPGA A executed only the application code, while he PC executed the PRPG/PSG and RDB&R processes. Fig. 3 summarizes the test execution flow.

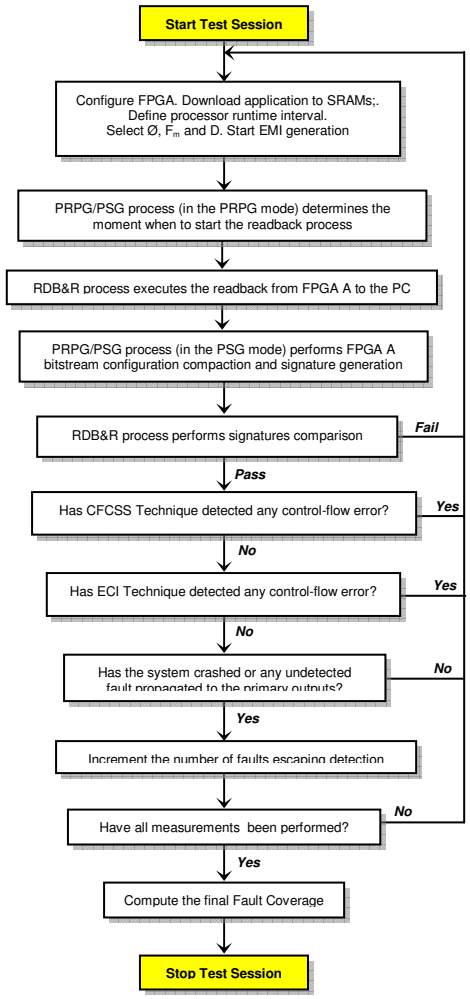


Fig. 3. Execution flow for a test session.

5. Experimental Results

The SUT described previously executed three applications: (***a***) *Prime Number Generator (PNG)*, (***b***) *Matrix Multiplication (MM)*; and (***c***) *Digital Filter (DF)*. Fig. 4 shows the SUT on-board memory address space.

For this experiment, the SUT was covered with an isolation film and then, shielded with a cooper foil connected to ground in two different ways:

a) The commercial board was completely covered and shielded by the materials mentioned above, except the FPGA IC, which was left uncovered by a window that fitted its packaging. Through this opening, EMI was irradiated during test (refer to this test configuration as "*Open Window*"). With this configuration, it was expected that most of the faults induced in the FPGA would be the result of faults generated by the coupling

[2] 200,000-gate Xilinx Spartan-3 XC3S200 FPGA in a 256-ball thin Ball Grid Array, made on a 90nm, eight layer metal process.

[3] Two independent 256Kx16 10ns SRAM chips.

[4] MicroBlaze™ is a true 32-bit soft RISC processor optimized for use in Xilinx's FPGA architectures. The processor's main memory interface conforms to the IBM CoreConnect specification for the On-Chip Peripheral Bus (OPB).

effect between the EM field and the FPGA die tracks/substrate material. This coupling effect would increment the ground bounce effect into the FPGA IC, which in turn rises critical path propagation delays.

b) The commercial board was left uncovered, except for the FPGA IC, which was covered by isolation film and shielded with cooper foil that fitted only its packaging. Then, EMI was irradiated all over the FPGA side of the board (the SRAM memory side was not irradiated). By doing so, it was expected that most of the faults affecting the FPGA would result from the coupling effect between the EM field and the board tracks/connectors and then, conducted towards the FPGA input pins. This test configuration is the complement of the *Open Window*. Refer to this test configuration as "*Closed Window*".

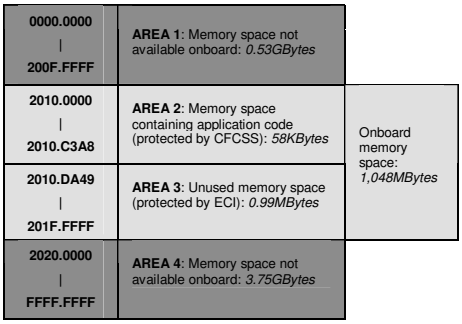


Fig. 4. SUT memory address space (Total size: 4GBytes).

Appli-cation Code	*Open Window*			*Closed Window*		
	Configuration Errors (%)	*Logic Errors (%)*		*Configuration Errors (%)*	*Logic Errors (%)*	
	Detected by SADR	Detected by CFCSS	Detected by ECI	Detected by SADR	Detected by CFCSS	Detected by ECI
PNG	100	33.33	100	100	28.86	NA*
MM	100	12.50	NA*	100	33.33	NA*
DF	100	NA*	NA*	100	33.33	NA*
Average (%)	100	22.92	100	100	31.84	0.00

* NA: *it was not observed faults affecting CPU control-flow.*

Table 1. Test Summary for *Open* and *Closed Windows*.

Table 1 summarizes the observed types of faults for the three codes (PNG, MM, and DF) running on the MicroBraze processor. "*Configuration Errors*" indicate those faults that change the FPGA configuration bitstream, whereas "*Logic Errors*" indicate those faults that drive the MicroBraze processor into an erroneous control-flow execution.

Table 1 was constructed with data extracted from 642 measurements (214 measurements for each of the three application codes: 107 for *Open* + 107 for *Close* configuration windows). Each measurement corresponds approximately to a time slot of 30 seconds of CPU execution under EMI exposure.

Table 1 is interpreted as follows:

a) Column "*Configuration Errors*" indicates those faults affecting *FPGA configuration* and that were detected by the SADR technique. (*Some of these faults resulted in CPU control-flow error; others did not change control flow, but resulted anyway in data processing error.)* The majority (~75%) of these faults produced *multiple bit flips* in the configuration bitstream, which resulted in system crash (i.e., no response to external stimulus, except to the JTAG communication and reset signal). The remaining faults (25%) were single faults or multiple faults in the configuration bitstream that *did not* induce to system crash. Regardless the type of fault, once requested by the PC, the JTAG mechanism was always able to perform the readback operation from the FPGA to the PC, where the bitstream was compacted and compared to the reference one.

Since EM field may be responsible for multiple errors in the configuration bitstream, it can be concluded that classic hardening techniques developed accordingly to the *single-fault assumption* are not adequate to cope with the EMI-induced faults. For instance, for TMR-based systems such multiple faults would provoke routing or logic configuration functions to induce errors in more than one replica which could force the voter to stop the system.

b) Column "*Detected by CFCSS*" is related to those faults that resulted in processor control-flow error and that the CPU stopped executing into the memory space allocated for the application code. (*These faults corrupted FPGA configuration logic, the functional logic, or a combination of these elements.)* Since the CFCSS technique *cannot ensure* the detection of faults that modify the CPU control-flow execution and the FPGA configuration bitstream, the detection capability of this technique is affected. Thus complimentary techniques like SADR and ECI are necessary to have increased the "probing" of EMI-induced faults.

c) Column *"Detected by ECI*" is related to those faults that resulted in CPU control-flow error and that the CPU stopped executing the application code into the physical memory space not used for application code or data. (*As above, these faults corrupted FPGA configuration logic, the functional logic, or a combination of these elements.)* Even though these faults were all detected (100%), they represented only 1.28% of the total number of faults occurred. This is explained due to the fact that only the unused memory space for application code and data (~ 1MBytes available onboard; Fig. 4: Area 3) was protected with "ECI" instructions. This unused available onboard memory represents only a small fraction (0.0244%) of the overall CPU addressable memory space (~ 4.2GBytes; Fig. 4: Areas 1 and 4). In conclusion, only 0.0244% of the memory could be protected by the ECI technique.

Note: those faults that were not detected by the CFCSS or ECI techniques, and did not modify the FPGA configuration bitstream as well, were classified as "*faults affecting exclusively application data processing*".

d) Finally, faults generated due to the coupling effect between the EM field and the board tracks/connectors, and then conducted towards the FPGA input pins were *the main FPGA failure mechanism* in the *Closed Window* test configuration. These faults induced mainly errors in the FPGA configuration bitstream (let's say, *hard faults*). On the other hand: faults induced by EM field over the FPGA packaging were *the main failure mechanism for this component* in the *Open Window* test configuration. These faults induced mainly errors in the FPGA application logic (let's say, *soft faults*).

Table 2 summarizes the distribution of faults affecting the FPGA in terms of bitstream configuration (*HW Configuration*), application logic (*Logic*), and *fail silence* for both *Open* and *Closed* test configurations.

As observed in this figure, the number of faults affecting FPGA configuration is twice greater in the *Closed Window* test configuration than in the *Open* test (9.33% versus 4.67%). Also, the number of faults affecting logic (control-flow faults) is roughly constant (around 35 to 38% of the total number of occurred faults). Considering that: ***a)*** in the *Closed Window* configuration, faults are generated mostly from *conducted EMI*, which couples with board cables and tracks, and then reaches the IC input pins; and ***b)*** in the *Open Window* configuration, faults are mostly consequence of radiated EMI through the packaging, which affects the IC through the coupling between the EM field and IC tracks and substrate material; it can be concluded that:

- FPGA configuration logic is more sensitive to *conducted EMI*, while logic is equally affected by conducted as well as radiated EMI;
- FPGA functional logic memory elements are from 4 to 5 times more sensitive to EMI than the memory elements used for configuration logic.

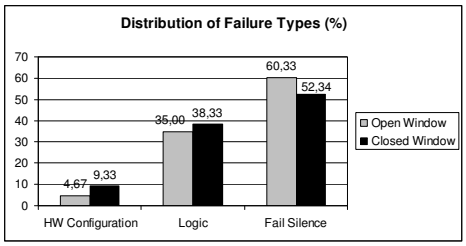


Table 2. Distribution of faults affecting the FPGA.

6. Final Considerations

This work observed the behavior of SRAM-based FPGAs as operating in EMI-exposed environments. To do so, it was used a commercial board containing an FPGA configured with an embedded microprocessor soft core and SRAM memories. To probe the FPGA behavior and classify the SEU types between faults affecting *logic* or faults affecting *configuration* memory elements, it was implemented some traditional techniques used in the field of on-line testing of digital circuits. The first technique was the "*Signature Analysis-Driven Refresh*". It aimed at detecting faults affecting FPGA HW configuration memory elements. The second and third techniques, *Control-Flow Checking by Software Signatures – CFCSS* and *Error Capturing Instructions – ECI* dealt with performing processor on-line control-flow fault detection. The EM environment was generated by means of a GigaHertz Transverse Electromagnetic (GTEM) cell according to the Std. Normative IEC 62132 [8].

The obtained results demonstrate the studied FPGA behavior when operating under EMI environment. Memory elements used by the FPGA *functional logic* are several times more EMI-sensitive than those used for *hardware configuration logic*.

Since the vast majority (~75%) of the faults affecting configuration bitstream resulted in multiple bit flips and system crash, classic hardening approaches (e.g., TMR-based systems) developed accordingly to the *single-fault assumption* are not adequate to cope with EMI-induced faults. The observation of a large number of multiple-bit faults can be explained due to the EM coupling mechanisms with the power supply lines and planes (at the board and IC levels), in particular GND (ground bounce), which simultaneously affected several memory elements. This large number of multiple-bit faults also explains the low detection capability presented by the CFCSS and ECI techniques.

At present, we are checking the relation between a bit modified in the FPGA configuration bitstream and the allocated (placed and routed) FPGA resources to propose an EMI-driven multiple-SEU fault detection technique.

References

[1] Steinecke, T. *Design-In for EMC on CMOS Large-Scale Integrated Circuits*. International Symposium on Electromagnetic Compatibility – EMC'2001. Vol. 2 , 2001. pp. 910 –915.

[2] Perez, R. *Signal Integrity Issues in ASIC and FPGA Design*. International Symposium on Electromagnetic Compatibility – EMC'1997. 1997. pp. 334 -339.

[3] Whyman, N. L.; Dawson, J. F. *Modelling RF Interference Effects in Integrated Circuits*. International Symposium on Electromagnetic Compatibility – EMC'2001. Vol.2, 2001. pp. 1203 –1208.

[4] Bezerra, E. A.; Vargas, F.; Gough, M. P. Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques. Journal of Electronic Testing: Theory and Applications – JETTA. Kluwer Academic Publishers, New York, USA. Vol. 17, May 1st, 2001, pp. 163-174.

[5] Oh, N.; Shirvani, P. P.; McCluskey, E. J. *Control-Flow Checking by Software Signatures*. IEEE Transactions on Reliability. Vol. 51, No. 2, March 2002. pp. 111-122.

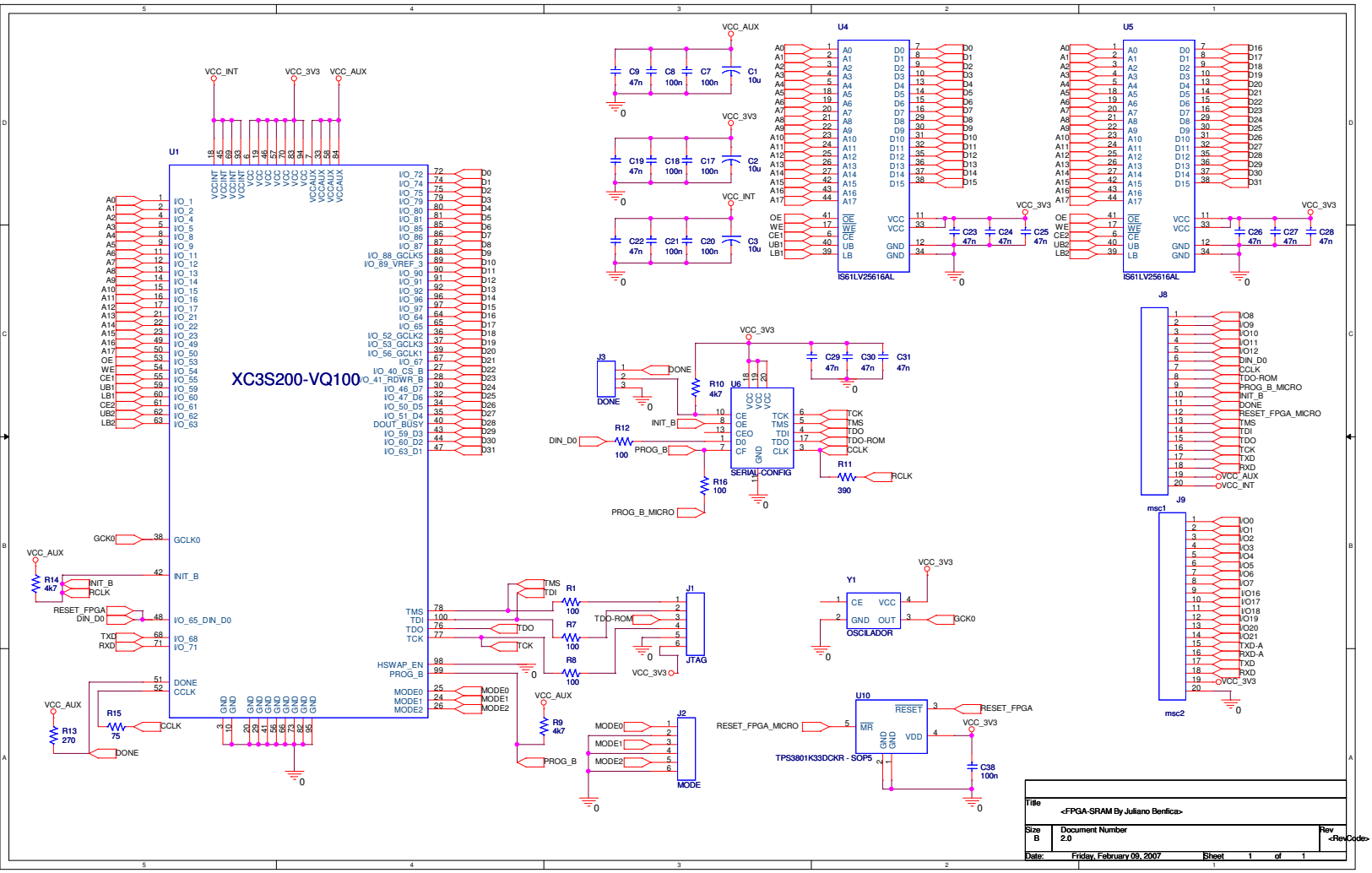
[6] Miremadi, G.; Torin, J. *Evaluating Processor-Behavior and Three Error-Detection Mechanisms Using Physical Fault-Injection*. IEEE Transactions on Reliability. Vol. 44, No. 3, September 1995. pp. 441-454.

[7] www.schaffner.com

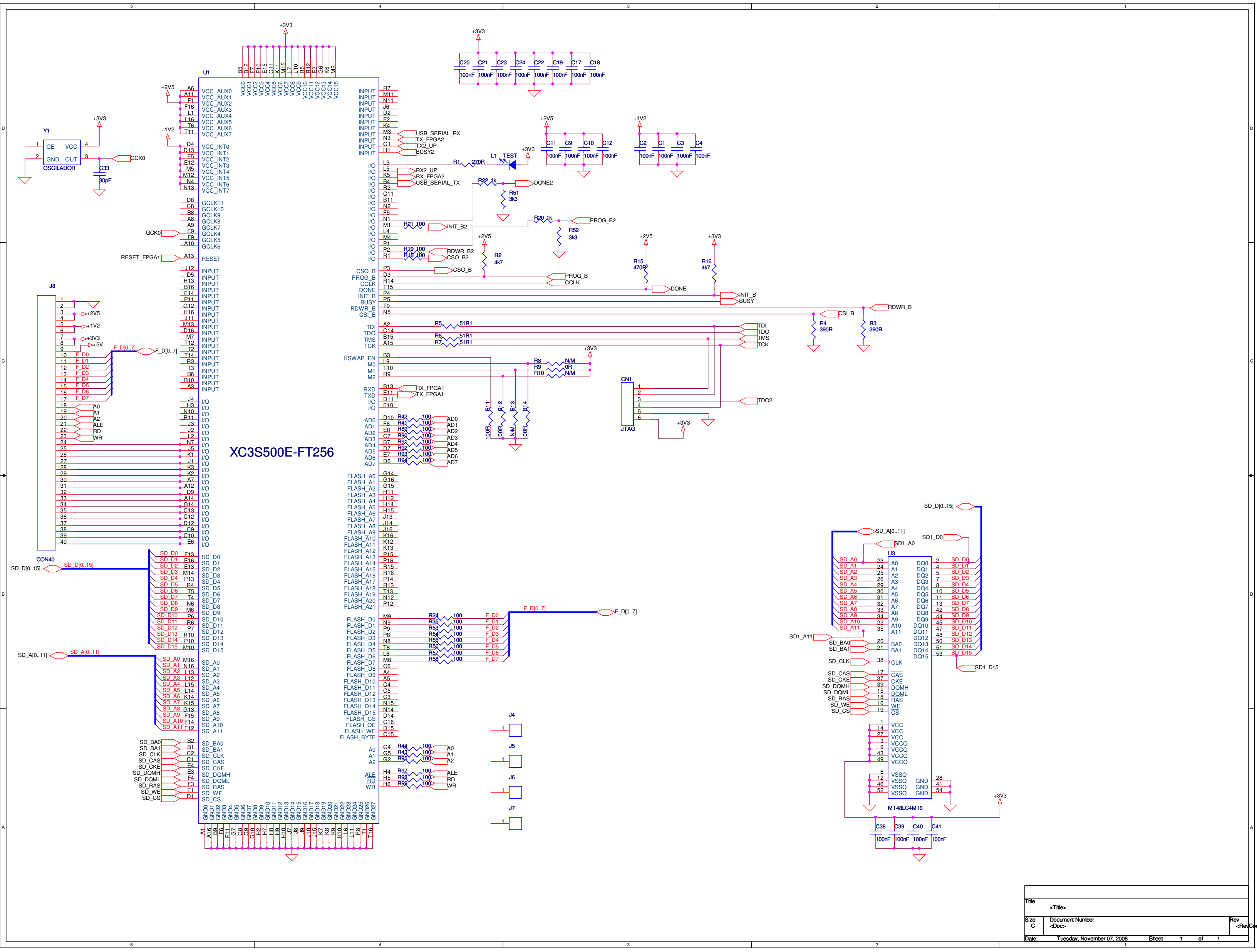
[8] *International Electrotechnical Commission - International Standard IEC 62132 (2004) Normative*. (www.iec.ch)

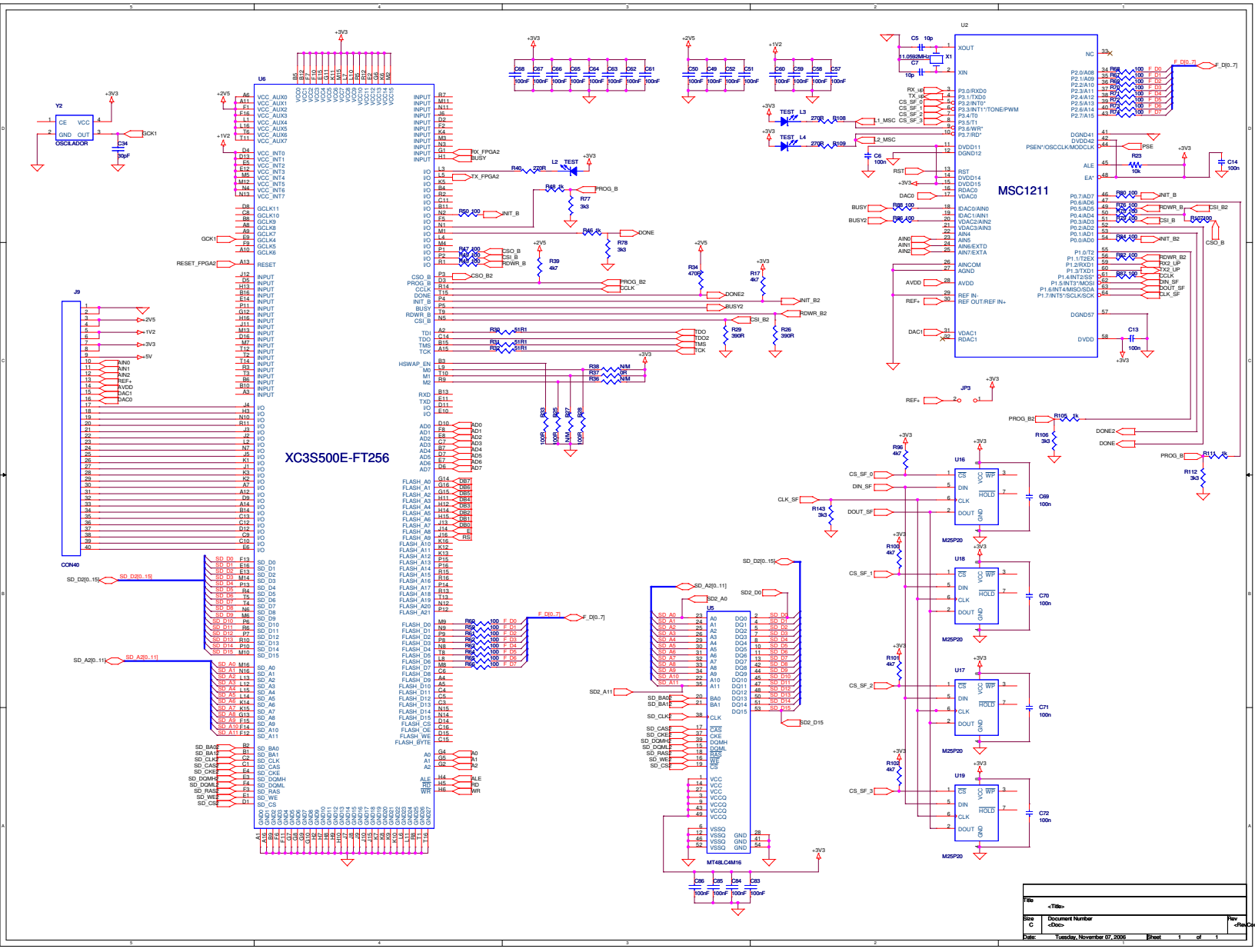
[9] Bellato, M.; Bernardi, P.; Bortolato, D.; Candelori, A.; Ceschia, M.; Paccagnella, A.; Rebaudengo, M.; Sonza Reorda, M.; Violante, M.; Zambolin, P. *Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA*. IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE'04).

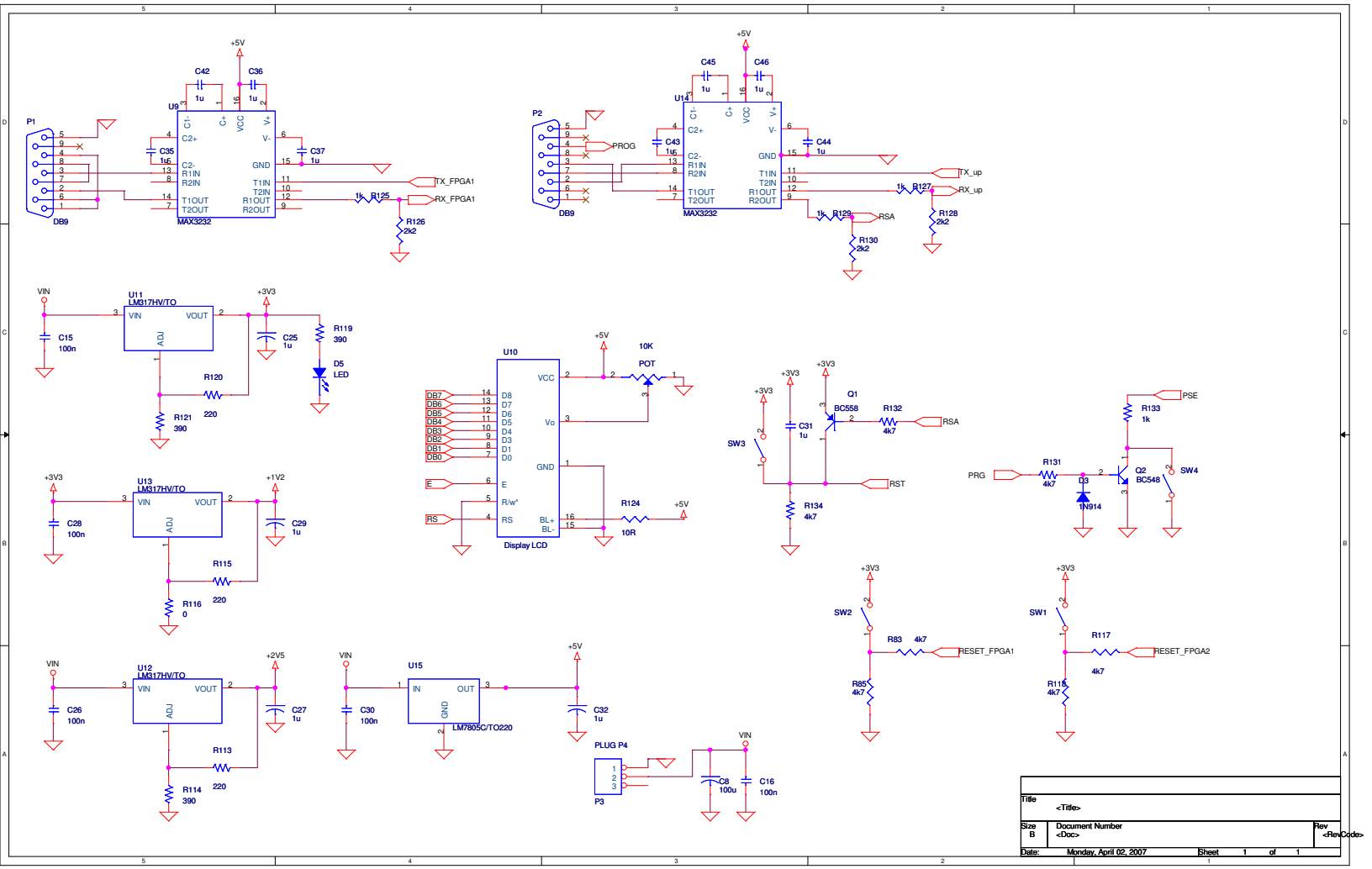
II. ESQUEMÁTICOS DA PLATAFORMA DE TESTE PARA ENSAIOS IRRADIADOS


XC3S200-VQ100
U1
U4
U5
U6
IS61LV25616AL
SERIAL-CONFIG
OSCILADOR
TPS3801K33DCKR - SOP5
JTAG
MODE
msc1
msc2
J1
J2
J3
J8
J9
Y1
VCC_INT
VCC_3V3
VCC_AUX
DONE
INIT_B
PROG_B
PROG_B_MICRO
DIN_D0
RESET_FPGA_MICRO
RESET_FPGA
TDO-ROM
RCLK
CCLK
GCK0
HSWAP_EN
TMS
TDI
TDO
TCK
TXD
RXD
<Title>
<FPGA-SRAM By Juliano Benfica>
Size B
Document Number 2.0
Rev <RevCode>
Date: Friday, February 09, 2007
Sheet 1 of 1

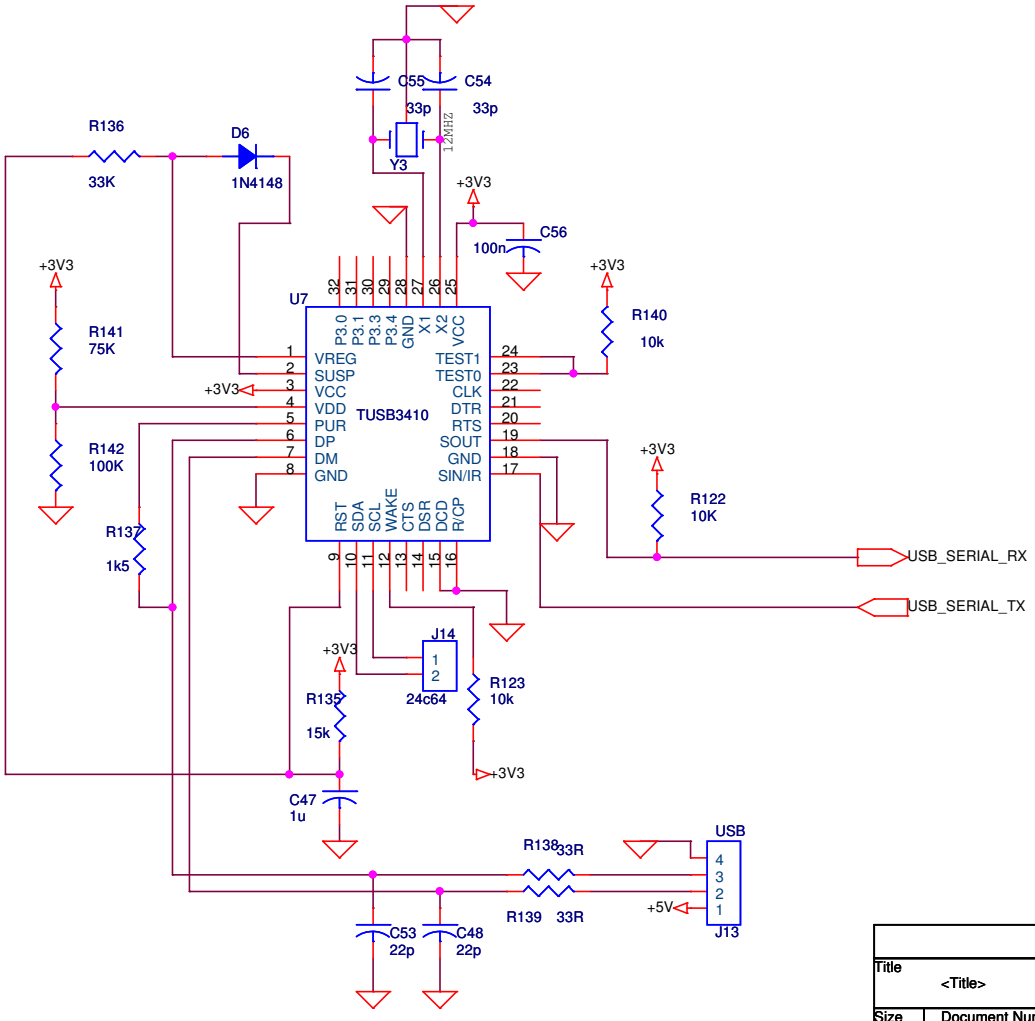
II. ESQUEMÁTICOS DA PLATAFORMA DE TESTE PARA ENSAIOS CONDUZIDOS


XC3S500E-FT256
U1
Y1
OSCILADOR
J8
CON40
CN1
JTAG
U3
MT48LC4M16
L1
TEST
USB_SERIAL_RX
TX_FPGA2
TX2_UP
BUSY2
RX2_UP
RX_FPGA2
USB_SERIAL_TX
DONE2
INIT_B2
PROG_B2
RDWR_B2
CSO_B2
CSO_B
PROG_B
CCLK
DONE
INIT_B
BUSY
RDWR_B
CSI_B
TDI
TDO
TMS
TCK
TDO2
RX_FPGA1
TX_FPGA1
RESET_FPGA1
GCK0
F_D[0..7]
SD_D[0..15]
SD_A[0..11]
SD1_A0
SD1_A11
SD1_D0
SD1_D15
SD_BA0
SD_BA1
SD_CLK
SD_CAS
SD_CKE
SD_DQMH
SD_DQML
SD_RAS
SD_WE
SD_CS
+3V3
+2V5
+1V2
+5V
Title
<Title>
Size
C
Document Number
<Doc>
Rev
<RevCode>
Date: Tuesday, November 07, 2006
Sheet 1 of 1

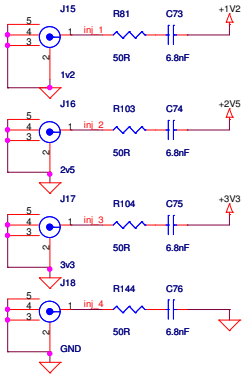

XC3S500E-FT256
MSC1211
MT48LC4M16
M25P20
OSCILADOR
CON40
Title
<Title>
Size
C
Document Number
<Doc>
Rev
<Rev>
Date:
Tuesday, November 07, 2006
Sheet
1
of
1

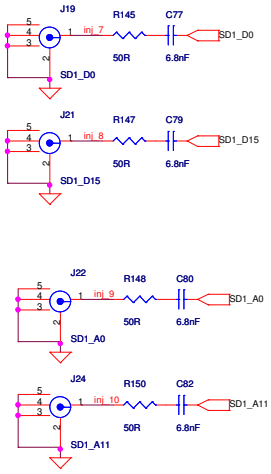


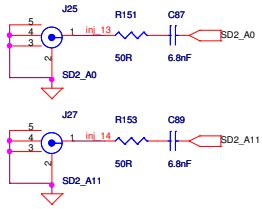
Title	<Title>	
Size B	Document Number <Doc>	Rev <RevCode>
Date: Monday, April 02, 2007	Sheet 1 of 1	

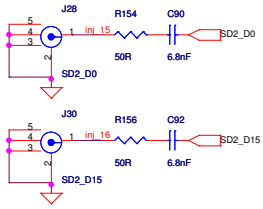


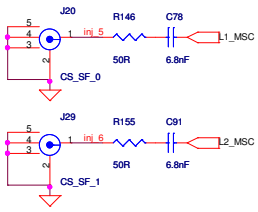
Title	<Title>	
Size A	Document Number <Doc>	Rev <RevCod
Date: Tuesday, October 24, 2006	Sheet 1 of 1	

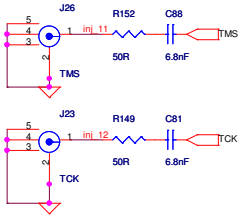












Title	<Title>	
Size: B	Document Number: <Doc>	Rev: <RevCode>
Date: Tuesday, November 07, 2006	Sheet 1 of 1	