

Antonio Carlos Luppi Junior

**Aplicação do modelo de atores em um
programa de teste e diagnóstico de circuitos
eletrônicos**

**Florianópolis
Julho 2017**

Antonio Carlos Luppi Junior

**APLICAÇÃO DO MODELO DE ATORES EM UM
PROGRAMA DE TESTE E DIAGNÓSTICO DE CIRCUITOS
ELETRÔNICOS**

Monografia submetida ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina como requisito da disciplina EEL7890 - Trabalho de Conclusão de Curso.

Orientador: Prof. Fernando Rangel de Sousa

Florianópolis

Julho 2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Luppi Junior, Antonio Carlos

Aplicação do modelo de atores em um programa de teste e diagnóstico de circuitos eletrônicos / Antonio Carlos Luppi Junior ; orientador, Fernando Rangel de Sousa, 2017.

108 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2017.

Inclui referências.

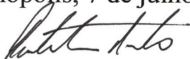
1. Engenharia Elétrica. 2. Engenharia Elétrica. 3. Teste Funcional. 4. Modelo de Atores. 5. Manufatura de Placas de Circuito Impresso. I. Rangel de Sousa, Fernando. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Antonio Carlos Luppi Junior

**APLICAÇÃO DO MODELO DE ATORES EM UM PROGRAMA
DE TESTE E DIAGNÓSTICO DE CIRCUITOS ELETRÔNICOS**


Este Trabalho foi julgado adequado para aprovação na disciplina
EEL7890 – Trabalho de Conclusão de Curso e aprovado em sua forma
final pelo programa de Graduação em Engenharia Elétrica.

Florianópolis, 7 de julho de 2017.

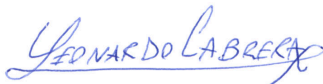


Prof. Renato Lucas Pacheco, Dr.
Coordenador do Curso

Banca Examinadora:



Prof. Fernando Rangel de Sousa, Dr.
Orientador
Universidade Federal de Santa Catarina



Prof. Fabian Leonardo Cabrera Riño, Dr.
Universidade Federal de Santa Catarina



Prof. Renato Lucas Pacheco, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado à minha mãe.

AGRADECIMENTOS

Agradeço primeiramente ao Professor Fernando Rangel pela paciência e profissionalismo em me orientar neste trabalho. Agradeço à Universidade Federal de Santa Catarina por me proporcionar educação gratuita e de qualidade, não só profissional como também cidadã.

Agradeço à Heloisa pelas muitas revisões textuais e pelo suporte e carinho em todos os momentos. Aos meus amigos de graduação pelo companheirismo neste anos de UFSC, em especial ao Lucas Stéfano, por revisar meu trabalho. E por fim, agradeço à Mirian, minha mãe, pelo apoio incondicional.

RESUMO

Este trabalho relata o processo de desenvolvimento de um software de testes funcionais para validação e detecção de falhas em placas eletrônicas em seu processo produtivo assim como nos retornos de manutenção. Os requisitos de flexibilidade e modularidade levaram a escolha do *framework* de atores, a implementação do Modelo de Atores de computação concorrente em LabVIEW. Foram criadas classes de atores para cada área de competência de teste: interface com multímetros, comunicação com o dispositivo sob teste, teste de potência RF, gerador de registros de teste e o controle de execução. Melhorias em ergonomia foram consideradas durante o processo de desenvolvimento, e testes prévios de execução das rotinas de teste apontaram melhorias no tempo de execução de testes e automatização do processo.

Palavras-chaves: teste funcional. teste estrutural. modelo de atores. programação concorrente. LabVIEW. montagem de placas de circuito impresso.

ABSTRACT

This work reports the development process of a functional test software for fail detection and validation of assembled circuit boards on their productive process and in field returns. The requirements in flexibility and modularity led to the choice of the actor framework, the Labview implementation of the Actor Model of concurrent computing. Actor classes were developed for each field of competence: multimeter interface, device under test communication, RF power test, test log generator, and execution control. Enhancements in ergonomics were considered during the development process, and prior tests runs of the routines indicated improvements in the test execution time and process automation.

Key-words: funcional test. structural test. actor model. concurrent programming. LabVIEW. printed circuit board assembly.

LISTA DE ILUSTRAÇÕES

Figura 1 – Uma típica linha de montagem de PCIs. Adaptado de Keysight (2003)	35
Figura 2 – A complementariedade dos testes estruturais e funcionais para a cobertura de teste de um dispositivo. Adaptado de Wenzel and Zimmermann (2016).	37
Figura 3 – Célula de Verificação Periférica	46
Figura 4 – Esquemático conceitual da lógica de teste <i>on-chip</i> . . .	47
Figura 5 – PCI com as portas JTAG em <i>daisy-chain</i>	48
Figura 6 – Rede IEEE 1687 conceitual de múltiplas cadeias de varredura	52
Figura 7 – Diagrama de blocos de um esquema de teste funcional .	55
Figura 8 – Um exemplo de máquina de estados finita	57
Figura 9 – Laço acionado por eventos	58
Figura 10 – Diagrama do produtor-consumidor com 4 laços independentes comunicando por uma fila	59
Figura 11 – Diagrama do tratador de mensagens em fila	59
Figura 12 – Um exemplo de uma rede com alguns atores, com destaque para um ator, sua caixa de mensagens e seu estado interno. Adaptado de Erb (2012)	62
Figura 13 – Mensagens no <i>framework</i> de atores em LabVIEW: (1) atores podem mandar mensagens para seus filhos; (2) atores podem mandar mensagem para os seus pais; (3) atores podem mandar mensagem para si mesmos	65

Figura 14 – Foto do GT650 com destaque aos componentes da placa: o <i>modem</i> EHS6 (1); o microcontrolador monitor <i>watchdog</i> (2); o conector DB9 com duas portas RS232 (3.a); as E/S isoladas (3.b); o expansor de E/S por I ² C (3.c) ; o painel de LEDs indicativos (3.d); os detectores de tensão trifásica (3.e); o conversor buck (4.a); a bateria e o seu sistema de gerenciamento (4.b); os pontos de teste de alimentação (4.c); e o conector da antena (5) (foto do autor).	68
Figura 15 – Fluxograma da bateria de testes	77
Figura 16 – Estrutura de pacote de comunicação do multímetro	78
Figura 17 – NI 5680 - instrumento utilizado para medição de potência RF	79
Figura 18 – Modelo do programa de teste reimplementado à partir do <i>framework</i> de atores, e adequado para a jiga de teste legada	84
Figura 19 – Diagrama UML de classes simplificado retratando as classes do <i>framework</i> , em conjunto com as classes de atores propostas e implementadas neste trabalho	86
Figura 20 – Diagrama de blocos do ModCOM, ator de comunicação serial	88
Figura 21 – Diagrama de blocos do DMM, ator de aquisição de medidas do multímetro digital	90
Figura 22 – Painel da VI de aquisição de medidas do multímetro digital	91
Figura 23 – Diagrama de blocos do logger, ator atribuído de gerar arquivos de registro das baterias de testes	92

Figura 24 – Diagrama de blocos do PowerMeter, ator de interface
com o NI 5680 94

Figura 25 – Diagrama de blocos do Controller, ator supervisor . 95

Figura 26 – Tela do painel principal de interface de usuário 96

Figura 27 – Captura de tela do Painel de teste de led 97

LISTA DE TABELAS

Tabela 1 – Tipos de ICT 44

Tabela 2 – Padrões IEEE de varredura periférica (Jutman et al.,
2014) 49

Tabela 3 – Média e desvio padrão das 20 amostras de teste de ban-
cada 100

LISTA DE ABREVIATURAS E SIGLAS

AOI	<i>Automated Optical Inspection</i>
ASIC	<i>Application-Specific Integrated Circuit</i>
ATS	<i>Automated Test Systems</i>
AXI	<i>Automated X-ray Inspection</i>
BERT	<i>Bit-error Rate</i>
BISD	<i>Build-in Self Diagnosis</i>
BIST	<i>Build-in Self Test</i>
BST	<i>Boundary Scan Test</i>
CA	Corrente Alternada
CAN	<i>Controller Area Network</i>
CC	Corrente Contínua
DfD	<i>Design for Debug</i>
DfT	<i>Design for Testability</i>
DfY	<i>Design for Yield</i>
EDGE	<i>Enhanced Data rates for GSM Evolution</i>
E/S	Entrada/Saída
FPGA	<i>Field-programmable Gate Array</i>

FPT	<i>Flying Probe Test</i>
GPRS	<i>General Packet Radio Service</i>
ICL	<i>Instrument Connectivity Language</i>
ICT	Teste Intra-circuito ou <i>In-Circuit Test</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IMEI	<i>International Mobile Equipment Identity</i>
IP	<i>intellectual property</i>
JSON	<i>JavaScript Object Notation</i>
JTAG	<i>Joint Test Action Group</i>
LAN	<i>Local Area Network</i>
LIN	<i>Local Interconnect Network</i>
LVDS	<i>Low Voltage Differential signaling</i>
M2M	<i>Machine-to-machine</i>
MDA	<i>Manufacturing Defect Analysis</i>
NFF	<i>No failure found</i>
NoC	<i>Network-on-a-Chip</i>
OEM	<i>Original Equipment Manufacturer</i>
OTAP	<i>Over-the-air Provisioning</i>
PCI	Placa de circuito Impresso

PCIe	<i>Peripheral Component Interconnect Express</i>
PCB	<i>Printed Circuit Board</i>
PCIM	Placa de circuito impresso e montada
PDL	<i>procedural description language</i>
PRPG	<i>Pseudo-random Pattern Generator</i>
QA	<i>Quality Assurance</i>
RSN	<i>Reconfigurable Scanning Networks</i>
SATA	<i>Serial AT Attachment</i>
SIM	<i>Subscriber Identification Module</i>
SIMcard	<i>Subscriber Identification Module card</i>
SNR	<i>Signal-Noise Ratio</i>
SPI	<i>Serial Peripheral Interface</i>
SoC	<i>System-on-a-chip</i>
SubVI	<i>Sub Virtual Instrument</i>
UART	<i>Universal Asynchronous Receiver/Transmitter</i>
VI	<i>Virtual Instrument</i>
VILW	<i>Very long instruction word</i>
XML	<i>eXtensible Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO	27
1.1	Objetivos	27
1.2	Motivação	27
1.3	Contexto produtivo e cenário de teste	28
1.3.1	Processo Produtivo Terceirizado	29
1.3.2	Ciclos rápidos de lançamento de novos produtos	30
1.3.3	Crescimento, maturidade e melhores práticas	30
1.3.4	Visão geral do problema	31
1.4	Organização deste manuscrito	32
2	O TESTE SISTÊMICO NA PRODUÇÃO DE PLACAS DE CIRCUITO IMPRESSO	33
2.1	Teste e diagnóstico pela cadeia produtiva: do circuito integrado ao teste sistêmico	33
2.1.1	Inspeção de Pasta de Solda	38
2.1.2	Inspeção ótica automática - AOI	39
2.1.3	Inspeção radiográfica automática - AXI	40
2.1.3.1	Tipos de AXI	41
2.1.3.2	Casos onde a inspeção radiográfica é importante	41
2.1.4	Testes intra-circuito (<i>In-Circuit Testers - ICT</i>)	42
2.1.4.1	Tipos de ICT	43
2.1.4.2	Cobertura de falta	43
2.1.5	Varredura periférica	45
2.1.6	Teste Controlado por FPGA ou Processador e Instrumentação Embarcada	49
2.1.7	Teste Funcional	53

2.2	Padrões de projeto em LabVIEW	56
2.2.0.1	Máquinas de Estados Finitos	56
2.2.0.2	Laços Acionados por Eventos	57
2.2.0.3	O Produtor - Consumidor	57
2.2.0.4	Tratador de mensagens em fila	58
2.2.1	Framework de Atores	60
2.2.1.1	Conceitos fundamentais do modelo	60
2.2.1.2	Distribuição e Escalabilidade	61
2.2.1.3	Robustez e tolerância a falhas em tempo de execução	63
2.2.2	Implementação em LabVIEW	63
3	ESPECIFICAÇÃO DO PROGRAMA	67
3.1	O dispositivo sob teste	67
3.1.1	Módulo M2M	69
3.1.2	Monitor <i>Watchdog</i>	70
3.1.3	Interfaces de usuário e interconexões externas	71
3.1.4	Conector e transmissor de radiofrequência (RF)	72
3.1.5	Alimentação, bateria e BMS	72
3.2	Cobertura e roteiro de teste	72
3.3	A Jiga e os instrumentos de teste	73
3.3.1	Multímetro	73
3.3.2	Medidor de potência do transmissor	73
3.4	Requisitos de Software	74
4	METODOLOGIA	81
4.1	Material Utilizado	81
5	MODELO E IMPLEMENTAÇÃO	83
5.1	Modelo do sistema no <i>framework</i> de atores	83

5.2	Implementação	85
5.2.1	Ator de comunicação serial	87
5.2.2	Ator Multímetro	89
5.2.3	Registrador de testes	91
5.2.4	Sensoriamento de potência do <i>front-end</i> RF	93
5.2.5	Controlador	93
5.2.6	Ergonomia	95
6	RESULTADOS E ANÁLISE	99
6.1	Resultados e Análise de Desempenho	99
6.2	Análise da implementação	100
7	CONCLUSÃO	103
7.1	Trabalhos futuros	103
7.2	Palavras finais	104
	Referências	105

1 INTRODUÇÃO

Este trabalho trata da reescrita e implementação de um programa de teste funcional e estrutural de PCIM (Placa de circuito impresso montada), utilizando-se do modelo de atores de computação concorrente (Hewitt, 2010; Hewitt and Zenil, 2013) com o foco em flexibilidade, reusabilidade e escalabilidade para grandes produções.

O projeto e implementação ocorreram durante um estágio na empresa de soluções M2M em Florianópolis, cuja expansão de produção de módulos, em conjunto com um maior rigor nas políticas de Garantia de Qualidade, implicou na necessidade de uma testagem sistêmica mais rápida e uma cobertura de testes mais abrangente. Além disso, pelos ciclos de desenvolvimento apresentarem passo acelerado e serem exigidas customizações de hardware quase que por cada lote produzido, o desafio torna-se ainda maior, sendo necessário maior facilidade e flexibilidade de adequação dos roteiros e hardware de teste.

1.1 OBJETIVOS

Implementar um programa de computador modular, concorrente e escalonável para testes estruturais, sistêmicos e funcionais em linha de produção.

1.2 MOTIVAÇÃO

Este trabalho se propõe a resolver problemas relacionados a teste de hardware em escala industrial, onde é necessário garantir velocidade e cobertura de teste, que são objetivos quase antagônicos.

O desafio torna-se ainda maior considerando os seguintes fatores:

- O aumento da densidade e a miniaturização de componentes na placa de circuito impresso, assim como o uso de encapsulamentos BGA, encarecem ou até mesmo impossibilitam o uso de técnicas tradicionais de teste estrutural *In-circuit*;
- Tecnologias de comunicação sem-fio e barramentos de alta velocidade sendo incorporadas em praticamente todo sistema embarcado fabricado, exigindo assim hardware de teste especializado.

Da mesma forma que os sistemas eletrônicos, embarcados ou não, tornam-se mais complexos, os desafios na validação de hardware e controle de qualidade em fábrica acompanham essa tendência.

Ao se tratar de programas de teste, é comum que, por falta de método ou pela dinâmica de trabalho imposta, o engenheiro de teste desenvolva programas de difícil reutilização ou manutenção. Em casos mais graves, todo o processo produtivo depende de um programa desenvolvido por algum ex-funcionário, e que, por falta de documentação ou técnicas de engenharia de software, é incompreensível para a equipe mantenedora do processo.

Situações como esta colocam processos produtivos inteiros em risco, ou, no melhor dos casos, fundamentados em técnicas antiquadas e legadas.

1.3 CONTEXTO PRODUTIVO E CENÁRIO DE TESTE

A análise do contexto produtivo é necessária para levantar as necessidades de desenvolvimento da empresa.

1.3.1 Processo Produtivo Terceirizado

No processo de manufatura estudado, a fabricação das placas de circuito impresso e montagem não é feita dentro da empresa, mas através de terceiros especializados. Isso permite que o trabalho seja feito por empresas com conhecimento e maquinário avançados, com custos geralmente mais baixos. Além disso, permitem que produtos menos complexos possam ser fabricados com um processo de fabricação mais simples e barato.

É certo que estas vantagens tenham seu preço e alguns problemas de uma montagem terceirizada precisam ser tratados: a necessidade de treinamento e instrução de trabalho; acompanhamento e controle de qualidade; deslocamento da equipe de suporte técnico de produção para as fábricas para realizar tais funções; incompatibilidades entre as políticas organizacionais das companhias envolvidas, como em políticas de qualidade, prazos, entre outros; e falta de controle sobre a produção no geral. A maior parte desses problemas são relativamente fáceis de tratar e remediar, razão pela qual a maior parte de empresas com pequena e média produção optam por manter a fabricação e montagem de PCB sob competência de terceiros.

No caso dos testes de manufatura, precisa-se levar em conta as questões ergonômicas do operador do teste que, caso não tratadas, resultam em custos maiores, lentidão, exaustão e, em casos extremos, danos permanentes à saúde do trabalhador.

Sendo assim, a automatização das baterias de teste, como também uma boa interface do usuário, se colocam como requisitos importantes no projeto do hardware e software de teste, diminuindo assim o estresse da mão de obra envolvida.

Algumas etapas de teste, como os testes de inspeção de PCB, de-

pendem da empresa de montagem ter um equipamento de inspeção ótica ou radiográfica — AOI e AOX, respectivamente. Caso não possuam, as falhas de PCB nas soldas ou interconexões somente serão detectadas nas etapas de teste final de linha de produção.

Estes últimos, que consistem de testes funcionais e estruturais do Dispositivo sob Teste (do inglês *Device-Under-Test* ou DUT), são feitos pelas jigas de testes fabricadas e disponibilizadas pela contratante, assim como os computadores e softwares de teste.

1.3.2 Ciclos rápidos de lançamento de novos produtos

O ciclo de desenvolvimento e lançamento de novos produtos da empresa também é outro fator importante. No caso estudado, novos produtos são lançados a cada semestre, isso sem contar a flexibilidade de hardware e customizações de produto que podem aparecer a cada lote fabricado – depende do acordado pelo departamento comercial com os clientes. Tal variedade de produto e velocidade de inovação tornam-se um exercício para o engenheiro de testes em pensar em modularidade e reusabilidade no sistema de testes.

1.3.3 Crescimento, maturidade e melhores práticas

À medida que uma organização e seu volume de produção cresce, seu método produtivo amadurece e começa-se a dar importância para melhorias de processo, assim como, o uso de ferramentas mais avançadas. No caso de manufatura, testagem de produtos e instrumentação, equipamentos de medição mais refinados são necessários para um diagnóstico preciso dos problemas de processo ou de projeto.

Pensando nisso, a empresa optou por escolher uma solução de software com alta compatibilidade com os equipamentos de teste disponíveis no mercado e, de certa forma, padrão para empresas do ramo: o ambiente de desenvolvimento e programação LabVIEW. Criado em 1986 pela National Instruments, baseia-se em uma linguagem de programação gráfica chamada G e é comumente aplicada em programas de aquisição de dados, automação industrial e controle de instrumentos de medição e teste. Existe também uma solução equivalente que faz uso de uma linguagem textual baseada em ANSI C, chamada Labwindows/CVI, sendo mais direcionada aos engenheiros de software e profissionais mais habituados com linguagens textuais. Geralmente é aplicada em contratos de prestação de serviços de desenvolvimento de sistemas, pela maior facilidade na realização de auditorias nos arquivos de código fonte. Pensando nos planos de escalonamento da empresa, foi decidido investir no uso dessa ferramenta nos processos de teste industriais. Outro problema encontrado é que, à medida que surgia a necessidade de criar novas práticas produtivas, soluções de software monolíticas e atômicas foram sendo criadas e aplicadas. Vendo isso, também foi estipulada a integração dessas soluções, ou até mesmo refazê-las, para que possam funcionar como uma só e agilizar os processos. Pensando também em tornar o roteiro de teste mais concorrente, usando recursos de *multi-threading*, facilmente alcançáveis usando o LabVIEW.

1.3.4 Visão geral do problema

A etapa de teste de linha de produção da fábrica apresentava problemas, como exemplo, a lentidão e a rudimentariedade deste estágio em relação ao crescimento da produção. Baseado nisso, foram levantados

quais aspectos o novo programa de teste deveria atender:

- Projeto de software modular, flexível e reutilizável;
- Uso das ferramentas específicas para instrumentação e teste;
- Paralelização de rotinas de teste, mantendo o cuidado para com possíveis problemas de programação concorrente como *deadlocks*;
- Uma interface de usuário mais ergonômica e funcional;
- Integração com outros testes realizados pela empresa.

1.4 ORGANIZAÇÃO DESTE MANUSCRITO

O documento está organizado em sete capítulos.

Este primeiro capítulo introduz o problema e os objetivos do trabalho. O capítulo 2 faz uma revisão das técnicas de teste e diagnóstico ao longo da manufatura de sistemas eletrônicos e PCIMs, como também revê os principais padrões de projeto de programas em LabVIEW. Em seguida, o capítulo 3 especifica e limita o problema tratado, enquanto o capítulo 4 aponta a metodologia utilizada para a solução.

No capítulo 5 é proposta a estrutura do programa, assim como a descrição detalhada da implementação. No capítulo 6 é feita a análise de resultados. E por fim, no capítulo 7, o trabalho é concluído com propostas para trabalhos futuros.

2 O TESTE SISTÊMICO NA PRODUÇÃO DE PLACAS DE CIRCUITO IMPRESSO

Os testes de circuitos eletrônicos estão presentes em toda a cadeia de projeto, fabricação e operação de um produto. Sua relevância se nota desde as primeiras etapas de validação de um projeto de um sistema eletrônico, passando pelas etapas de produção de circuitos integrados, controle de qualidade do processo de montagem de placa, e até mesmo, no diagnóstico em campo do produto final. Como o presente trabalho discorre sobre testes de fim de linha de produção, o foco será nas etapas de teste finais na cadeia produtiva.

2.1 TESTE E DIAGNÓSTICO PELA CADEIA PRODUTIVA: DO CIRCUITO INTEGRADO AO TESTE SISTÊMICO

O teste de circuitos inicia-se no teste de circuitos integrados, na verificação de possíveis danos aos *wafers* de silício. Em Mitra et al. (2010), vê-se que estes testes se caracterizam pelo uso de jigas de teste complexas e uso de técnicas de projeto orientado à testabilidade (DfT) e, também, pela introdução de varredura periférica, autoteste embutido e compressão de dados de teste.

Já na linha de montagem de placas de circuito impresso, existe variedade de abordagens possíveis para a realização de testes, medições e inspeções. Dentre as técnicas específicas desta etapa destacam-se a inserção de pontos de teste, inspeções óticas e de raio-x automatizadas, o uso de camas de prego nas jigas de teste e o uso do IEEE 1149.1, o JTAG. Cada uma dessas técnicas atendem objetivos específicos de teste e serão descritas nas próximas sessões ainda neste capítulo. Muitas vezes, a própria infraestrutura e funcionalidades de teste dos componentes integrados

podem ser reutilizados no teste de placa, como no trabalho de Cook et al. (2012), que descreve o reuso da infraestrutura de varredura periférica em testes em campo.

Um esquema típico de uma linha de montagem de placas de circuito impresso é representado pela figura 1. Cada uma dessas categorias de teste tem por objetivo detectar um tipo específico de falha, de forma que juntos consigam atingir uma boa cobertura de teste. No trabalho de Hird et al. (2002), o conceito de cobertura de teste é apresentado como a medida do quanto da placa de circuito impresso está verificada ou também como um indicador numérico da qualidade do teste. Por ser uma medida de cobertura, há vários padrões de modelagem de cobertura de faltas. Lotz et al. (2006) apresenta alguns padrões:

- MPS (*Material, Placement, Solder* - Material, Colocação e Solda), desenvolvido pela *Philips Research*;
- PPVS (*Presence, Polarity, Value, Solder* - Presença, Polaridade, Valor e Solda), desenvolvido pela *ASTER Ingénierie*;
- PCOLA/SOQ (*Presence, Correctness, Orientation, Live Alignment / Short, Opens, Quality* - Presença, Exatidão, Orientação, Vivo, Alinhamento / Curtos, Circuito Aberto, Qualidade), criado pela *Agilent Technologies*, estende as métricas de teste para questões de interconexão;
- FAM (*Feature, At-speed, Measurement* - Característica, em Velocidade Nominal, Medição) ou PCOLA/SOQ/FAM é uma extensão do item anterior. Foi desenvolvido pelo iNEMI para cobrir métricas funcionais (Ley and InterTech, 2009).

- DMPSF (Design, Material, Placement, Solder, Function - Projeto, Material, Colocação, Solda, Função), desenvolvido por Lotz et al. (2006).

A partir de um destes padrões criam-se listas de controle para análise de cobertura de cada componente e interconexão da placa. Dessa forma, pode-se mensurar a cobertura de testes de um produto nos aspectos materiais, estruturais e funcionais.

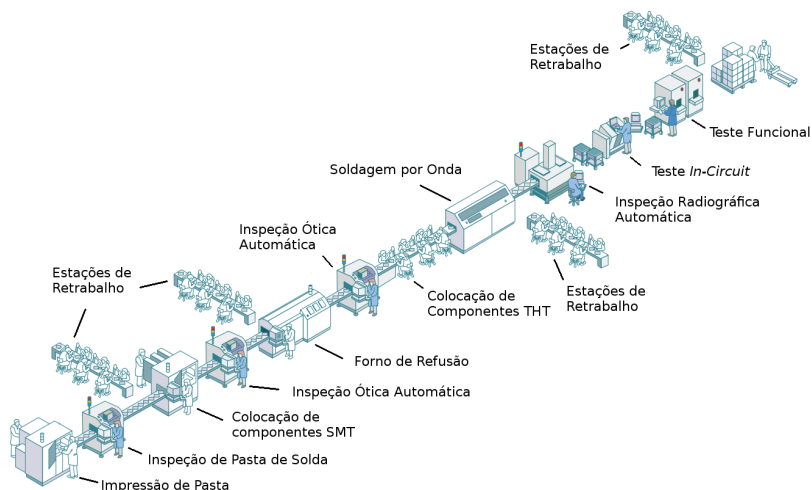


Figura 1 – Uma típica linha de montagem de PCIs. Adaptado de Keysight (2003)

Juntamente com as métricas de cobertura de teste, os modelos de falta também cumprem um papel importante no estudo de testes de manufatura. Jutman et al. (2014) separa os modelos de falta nas seguintes

classes: defeito materiais; defeitos em pinos e malha de circuito; problemas funcionais; e problemas de desempenho.

Defeitos em materiais e mecânicos podem ser mensurados por métricas padrão de processo de qualidade e exemplificados por defeitos em placas, processo de montagem: solda ruim terminal levantado, componente defeituoso, desalinhamento, efeito lápide¹.

Os defeitos em pinos e malhas de circuito são modelados por análise estrutural e já possuem uma base bem estabelecida. Entram nessa classe as falhas de circuitos abertos ou em curto, defeito de driver (*buffer*, pino).

Há ainda os problemas *funcionais*, como falhas de *boot*, cujas métricas são estabelecidas e investigadas por pesquisadores em desenvolvimento de software e lógica descritiva. Por final, tem-se os problemas de desempenho em linhas de comunicação: alta taxa de erro, *crosstalk*, *jitter*, *delay*, e outros.

Tradicionalmente na indústria, defeitos em materiais são solucionados durante a montagem da placa. Já os defeitos de pino, malha de circuito e funcional são testados no fim de linha de produção por duas categorias de teste denominados *testes estruturais* e *testes funcionais*. A relação entre estas duas categorias de testes é descrita em Wenzel and Zimmermann (2016) e sintetizada na figura 2. Porém, em Wenzel (2013) e Jutman et al. (2014), nota-se que essa segmentação vem mudando com os avanços nas técnicas de verificação periférica e de teste centralizado por microcontrolador, que incorporam boa parte dos testes de desempenho e até mesmo funcionais. Tais técnicas serão descritas na seção 2.1.6 deste mesmo capítulo.

¹ Efeito Lápide, ou *tumbstone effect*, é a elevação parcial ou total de componentes SMT passivos durante a refusão, assemelhando-se muitas vezes à uma lápide.

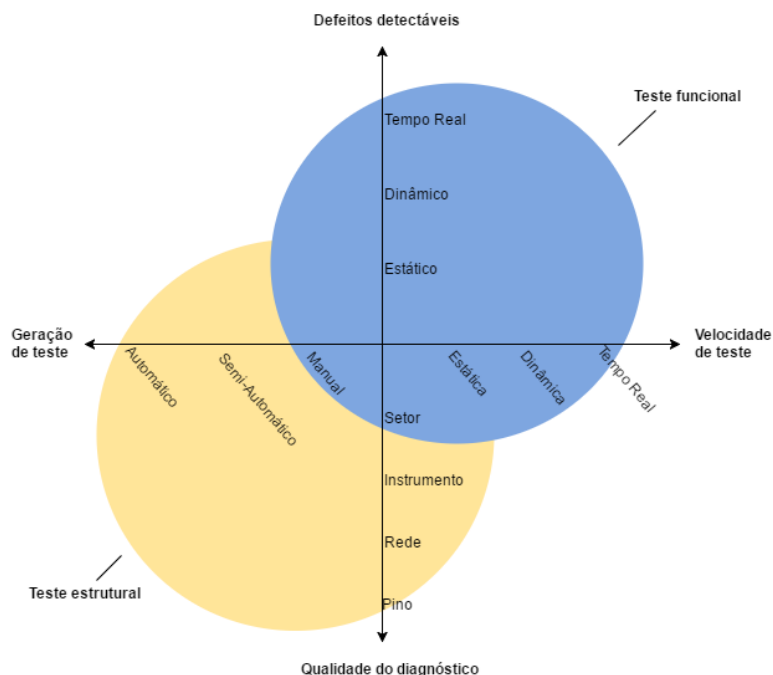


Figura 2 – A complementariedade dos testes estruturais e funcionais para a cobertura de teste de um dispositivo. Adaptado de Wenzel and Zimmermann (2016).

Viu-se aqui, portanto, que diversas técnicas e abordagens de teste e diagnóstico integram-se e são parte importante da produção, operação e manutenção de um sistema eletrônico. A seguir, estas técnicas serão descritas detalhadamente.

2.1.1 Inspeção de Pasta de Solda

A inspeção de pasta de solda é o processo, normalmente automático, de achar falhas após a aplicação da pasta de solda. A inspeção de pasta de solda previne o retrabalho de placas com os componentes montados porque, uma vez com os erros de solda detectados, a placa defeituosa pode ser separada das demais. A detecção de falhas ainda nesta etapa custa 10 vezes menos do que uma falha pós refusão, ou após a solda, e 70 vezes menos que uma falha no teste elétrico intra-circuito (*In-circuit test*) (Owen and Hawthorne, 2000).

Os sistemas mais simples de inspeção de pasta de solda consistem de uma imagem colorida da placa. Este método consegue detectar ausência de solda e conexões indevidas entre blocos vizinhos, porém, por ser um método 2D, não consegue estimar a altura e volume da pasta de solda. Isso pode gerar problemas indetectáveis em outras etapas, já que, por mais que passe em testes de condutividade, a falta de pasta de solda em uma conexão pode causar problemas mecânicos e rompimento da interconexão.

O volume de pasta de solda pode ser estimado por diversos métodos. Pang and Chu (2009) introduz as principais tecnologias de perfilometria:

- Triangulação a laser;
- Perfilometria por fase, utilizando luz estruturada;
- Reconstrução por Redes Neurais.

Os métodos por triangulação a laser utilizam-se das projeções do feixe de laser no relevo da placa para reconstruir o seu perfil 3D. Luo and Zhang (2010) descreve os detalhes desta técnica. Medições precisas

podem ser obtidas com este método. Entretanto, são equipamentos caros, com baixa velocidade de inspeção, e susceptíveis a ruídos refletivos e *efeito de sombras* (Pang and Chu, 2009).

Na perfilometria por fase, uma luz estruturada projeta um padrão, como uma grade ou uma série riscos, na PCI com a pasta de solda aplicada. Este padrão é então pouco a pouco deslocado e uma série de imagens são capturadas por uma câmera de alta resolução. Os sistemas mais avançados empregam um sistema livre de sombras onde ambos os lados da deposição de solda são caracterizados simultaneamente. Luo and Zhang (2010) oferece uma abordagem para o problema das sombras e Pang and Chu (2009) faz uma revisão dos principais trabalhos em perfilometria por fase.

O método de reconstrução por redes neurais tenta resolver os problemas do *shape-from-shading* (SFS), que até então era um método cujo desempenho era inconsistente. Pang and Chu (2009) revisa o estado da arte e propõe uma nova abordagem.

2.1.2 Inspeção ótica automática - AOI

A inspeção ótica automática (AOI), como o próprio nome indica, inspeciona visualmente a placa de circuito por meio de fotos, processa e identifica automaticamente as falhas, seja de componente ou solda. Esta técnica é usada para detectar componentes trocados, faltantes ou mal posicionados e pode ser realizada na etapa pré-forno de refusão ou após, dependendo da particularidade do processo produtivo da fábrica. O trabalho de Huang and Pan (2015), citado por de Melo (2015), indica quatro categorias de inspeção ótica automática:

- **Métodos de projeção**, que correlacionam a entrada com modelos aprendidos e suas características;

- **Abordagem baseada em filtro:** filtro espaciais com base em transformadas de sinal, como por exemplo, Fourier, Wavelet, discreta cosseno;
- **Aprendizagem de máquinas:** Os mais utilizados são as redes neurais, algoritmos genéticos, e máquinas de vetor suporte;
- **Abordagens híbridas:** associam diversas técnicas para classificações mais complexas.

A maior limitação da AOI é a impossibilidade de detecção de defeitos em pinos ou partes do circuito não visíveis, como abaixo de componentes *Ball Grid Array* (BGA) ou em circuitos eletromagneticamente blindados. Nesses casos, é necessário o uso de outras ferramentas, como a verificação de placas por radiografia.

2.1.3 Inspeção radiográfica automática - AXI

A inspeção radiográfica automática (AXI) possui uma vantagem única em relação às outras tecnologias de inspeção estrutural: os materiais absorvem os raios-x proporcionalmente à sua massa atômica. A solda utilizada na montagem das PCIs consiste de materiais pesados como o chumbo e a prata. A maioria restante dos materiais usados na fabricação das PCIs é composta por elementos leves como carbono, silício, alumínio, oxigênio, hidrogênio e cobre. Dessa forma, a inspeção radiográfica mostra-se muito interessante na geração de imagens do processo de solda: os pontos de solda aparecem muito bem, enquanto o restante da placa apresenta-se transparente. Outra vantagem é a possibilidade de inspecionar os pinos escondidos de encapsulamentos complexos como o BGA ou

o *chip-scale packages* (CSP), além de exibir características internas dos pontos de solda.

2.1.3.1 Tipos de AXI

As técnicas AXI utilizadas em processos de manufatura e diagnóstico se dividem entre bidimensional e tridimensional (Huang et al., 2015; McClure, 2000).

A inspeção 2D é semelhante à radiografia convencional, porém com recursos de visão computacional para análise automática. É melhor aplicada em placas com soldas em um só lado da placa. Em placas com junções de solda em alta densidade ou em ambos os lados, a imagem gerada pode ficar confusa.

Com a laminografia e tomografia computadorizada é possível obter a imagem de um recorte da placa e, associada a técnicas de processamento digital, reconstruir um modelo tridimensional da placa. A laminografia utiliza-se do movimento do emissor e detector que pode ser rotacional (Huang et al., 2015) ou linear (Suzuki et al., 2013). A desvantagem da radiografia tridimensional é o tempo necessário para a captura da imagem e seu processamento (Suzuki et al., 2013).

2.1.3.2 Casos onde a inspeção radiográfica é importante

Leinbach and Oresjo (2001) apresenta três casos aonde a inspeção radiográfica se faz importante. Primeiro, na inspeção de componentes visualmente ocultos, como terminais de encapsulamentos CSP e BGA ou componentes sob blindagem RF. Segundo, em aplicações que exigem alta qualidade de solda, como as PCI expostas a ambientes de estresse mecânico ou térmico. Terceiro, em projetos de PCI complexos, cujas taxas de

erro de produção por placa são mais frequentes e a AXI consegue garantir o rigor do processo de inspeção.

Huang et al. (2015) faz um estudo comparativo entre tecnologias de inspeção radiográfica para detecção de defeitos em furos galvanizados. Xu et al. (2016) aplicou AXI na inspeção de componentes *press-fit* e Liu et al. (2016) na verificação do resinamento de placas de circuito impresso montadas. Combinada com técnicas de ICT, a maior parte dos defeitos podem ser cobertos. O trabalho de Oresjo (2002) comenta melhor a relação entre AOI, AXI, e os testadores intra-circuito (*in-circuit testers* - ICT).

2.1.4 Testes intra-circuito (*In-Circuit Testers* - ICT)

Jigas de teste são uma grande ferramenta de teste de placas de circuitos eletrônicos. Por meio de pontas de teste, pode-se obter acesso a pontos internos do circuito, como também, testar componentes isoladamente, separando-os de outros circuitos. Por meio de ICT, é possível detectar componentes defeituosos ou faltantes, circuitos abertos ou em curto e, até mesmo, componentes errados. O ICT trabalha testando partes isoladas da placa, medindo resistência, capacitância e, em alguns casos, indutância de subcircuitos da placa, normalmente acessíveis por conectores, pads, ou *test points*. Dessa forma, consegue-se realizar testes estruturais e até mesmo funcionais sobre o circuito, garantindo que sua fabricação foi correta e que se encontra funcional. Muitas vezes, o ICT é usado também para a gravação de *firmware* nas placas. Os ICT são compostos pelos seguintes elementos (Poole, 2016):

- O ICT em si: composto por uma matriz de pares de acionadores e sensores que são usados para realizar as medições. Podem ser em torno de centenas, ou até milhares, por ICT.

- A jiga de teste (*fixture* ou fixação): é a interface entre o ICT e a placa testada, roteando as entradas e saídas do ICT com o Dispositivo em Teste, através de pinos, num arranjo conhecido como cama de pregos.
- O software de teste: software com a rotina de testes e condições de aprovação/reprovação.

Dentre os três, somente o ICT em si não é customizado para cada placa. O sistema de jiga de testes, por ser relativamente caro, é mais bem aplicado a grandes volumes de produção. Uma análise de custos deve de ser realizada para garantir que os custos de montagem da jiga e do programa sejam viáveis.

2.1.4.1 Tipos de ICT

A tabela 1 sintetiza três categorias principais de testadores (Poole, 2016): O ICT tradicional; O *flying probe test* - *FPT*, cujas pontas de prova são controladas por comando numérico computadorizado (CNC); e o Analisador de Defeito de Fabricação: versão mais enxuta do ICT, com funcionalidades reduzidas.

2.1.4.2 Cobertura de falta

Em sua tese, de Carvalho Felgueiras (2008) relata que os testes funcionais e estruturais realizados por ICT foram progressivamente dificultados por duas questões.

A primeira é a miniaturização dos circuitos e componentes com valores muito baixos, cuja verificação é comprometida devido à influência de capacitâncias parasitas do próprio testador. O mesmo acontece com as

Tabela 1 – Tipos de ICT

Tipo de ICT	Princípio de Funcionamento	Vantagens	Desvantagens
ICT tradicional	<ul style="list-style-type: none"> - Pares de I/O em grande número; - Fixação e cama de pregos feitos sob medida. 	<ul style="list-style-type: none"> - Alta velocidade de execução de teste; - Capacidade para medição de capacidade e indutância. 	<ul style="list-style-type: none"> - Inviável para pequenas produções; - Custo adicional para qualquer atualização da jiga;
<i>Flying probe test (FPT)</i>	<ul style="list-style-type: none"> - Pontas de prova móveis e controladas por CNC; - A fixação de placa é genérica; 	<ul style="list-style-type: none"> - Dispensa uma jiga de teste e cama de pregos; - Alterações realizadas via software; 	<ul style="list-style-type: none"> - Número limitado de pontos de testes simultâneos; - Baixa velocidade de teste;
Analizador de Defeitos de Fabricação	<ul style="list-style-type: none"> - Similiar ao ICT padrão, mas com funções limitadas. 	<ul style="list-style-type: none"> - Alta velocidade de teste e custo reduzido; 	<ul style="list-style-type: none"> - Limita-se a testes de continuidade e resistência elétrica.

indutâncias mas, ao menos, é possível detectar a presença do componente ao medir a resistência entre os terminais.

O segundo problema, e também o mais grave, é relacionado ao acesso aos nós do circuito, seja pelas dimensões reduzidas da placa sob teste, que dificultam a construção de uma cama de pregos correspondente, como também pela completa inacessibilidade a uma região eletromagneticamente blindada ou a pinos inacessíveis de encapsulamentos de alta densidade, como os BGAs, que, hoje em dia, estão quase sempre presentes em placas de circuito impresso.

Estes fatores deram origem ao desenvolvimento da infra-estrutura normatizada de circuitos de varredura periférica IEEE 1149.1 (IEEE, 1990) para facilitar o teste de integrados digitais que, nas últimas décadas, expandiu-

se em outros padrões que cobrem desde medições analógicas a testes de canais de comunicação.

2.1.5 Varredura periférica

Testes de varredura periférica (*boundary scan*) como o *Joint Test Action Group - JTAG* são hoje o padrão da indústria em teste de placas de circuito impresso e vem, ao longo dos últimos anos, substituindo os ICT. Isso se deve ao custo baixo desta tecnologia e eficiência em termos de cobertura de testes. Os primeiros esforços para a criação de um padrão de varredura periférica ocorreram na década de 80 e foram encabeçados pelo *Joint Test Action Group - JTAG*, cujo trabalho se consolidou no padrão IEEE (1990), que obteve rápida adoção pela indústria eletrônica. Sua revisão mais atual é a IEEE (2013). A necessidade de um padrão de varredura periférica surgiu devido aos crescentes custos e dificuldades na criação de jigas de teste para ICTs, principalmente pelo aumento da densidade de componentes e pela falta de espaço para a inserção de pontos de teste no leiaute de circuito impresso, como também, por causa da distribuição de componentes em ambas as superfícies.

Embora as primeiras aplicações do JTAG fossem direcionadas para teste de placas de circuito impresso, hoje o padrão também cumpre um papel essencial na depuração de sistemas embarcados, graças ao acesso de baixo nível às entradas/saídas e estados internos do circuito integrado, fornecendo um meio barato e confiável de depuração. A porta JTAG também serve para a gravação de *firmware* na memória flash (IEEE, 2003b), sendo uma alternativa mais rápida ao uso de portas seriais e *bootloaders*.

Outra aplicação desta interface é a geração de testes automatizados para diagnóstico de placa em campo no conceito de BIST, ou auto teste

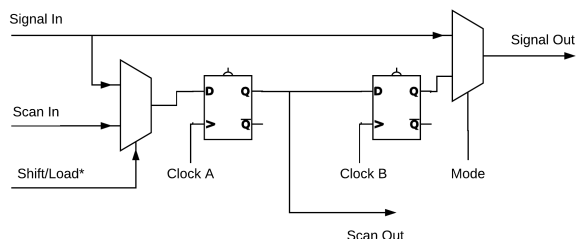


Figura 3 – Célula de Verificação Periférica

embutido, possibilitando que uma placa faça auto-diagnóstico de problemas, como em circuitos periféricos. Estes conceitos serão abordados na próxima seção deste trabalho.

O padrão IEEE 1149.1 se baseia em células de teste embutidas em todos os pinos de E/S do circuito integrado (figura 3). As células são encaadeadas em uma topologia em anel, um esquema conhecido como *daisy-chain*. O gerenciamento da cadeia de verificação é realizado pela porta de acesso ao teste (ou, em inglês, *test access port - TAP*), permitindo que todos os pinos de E/S do circuito integrado sejam acessíveis e testados pela porta JTAG. A figura 4 mostra o esquemático conceitual da lógica interna da porta JTAG. A conexão em *daisy-chain* entre os pinos de um dispositivo com suporte ao IEEE 1149.1 pode facilmente ser estendida para outros integrados compatíveis com o padrão, possibilitando o teste de múltiplos circuitos integrados através de um único conector (figura 5). Como boa prática de DfM, recomenda-se a escolha de integrados que atendam ao padrão.

Outro componente importante do JTAG é a linguagem de descrição de varredura periférica (BSDL em inglês), que foi inserida numa revisão

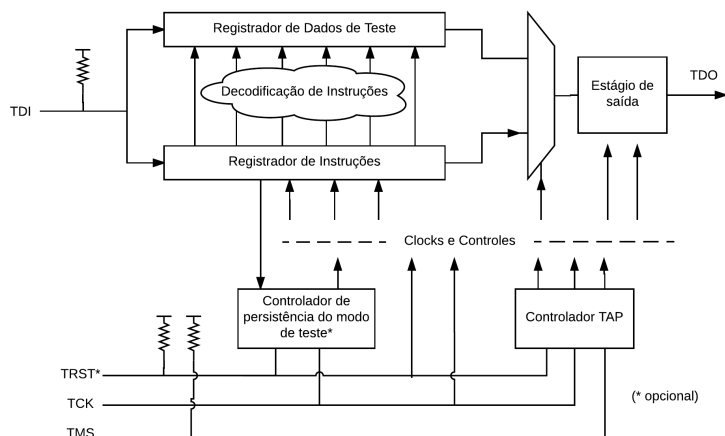


Figura 4 – Esquemático conceitual da lógica de teste *on-chip*

posterior do IEEE 1149.1 (IEEE, 1995). A BSDL é um subconjunto do VHDL e é voltada para a descrição da infraestrutura de teste de um CI com o objetivo de tornar mais consistente a geração de testes e toda a cadeia de desenvolvimento envolvida nisso.

O padrão IEEE 1149.1 serviu de base para a criação de uma família de padrões de varredura periférica, que são exibidos na tabela 2 - adaptada de Jutman et al. (2014).

O padrão IEEE 1149.4 (IEEE, 2011) foi criado para testes paramétricos de componentes passivos, resistores pull-ups, componentes ativos como diodos, transistores e de redes de impedância. Serve como uma extensão da varredura periférica para sinais analógicos. Este padrão ainda tem um número limitado de dispositivos compatíveis.

Já o padrão IEEE1149.6 (IEEE, 2003a) estende a varredura peri-

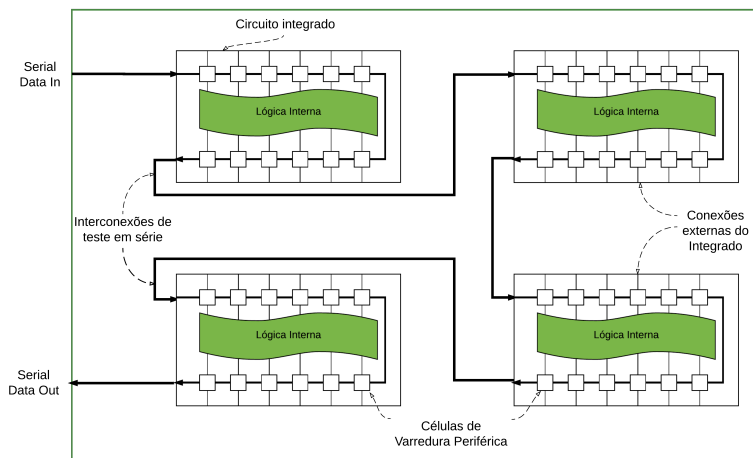


Figura 5 – PCI com as portas JTAG em *daisy-chain*

férica a portas de comunicação com acoplamento em corrente alternada, como por exemplo, portas que atendem ao padrão de LVDS, operando em paralelo com os padrões 1149.1 e 1149.4. O que possibilita esta tecnologia são as inserções de geradores de pulso nos pinos de saída e *receivers* CA comuns e diferenciais nas entradas dos dispositivos que dão suporte a este padrão.

É possível citar outros exemplos significativos, como é o caso do IEEE1149.7 - cJTAG (IEEE, 2010). Este padrão opera em pinos reduzidos e possibilita uma topologia em estrela ao invés do *daisy-chain*, permitindo um acesso mais rápido, além de oferecer uma infraestrutura adicional de suporte a tecnologias mais avançadas. O IEEE P1687 (IEEE, 2014) introduz o conceito de instrumentação embarcada para a aplicação de tarefas de teste, medição e diagnóstico, assunto abordado na próxima seção.

Tabela 2 – Padrões IEEE de varredura periférica (Jutman et al., 2014)

Foco principal de aplicação	Propósito Principal	Base da Tecnologia	Classes de falta a serem cobertas
IEEE 1149.1 - Boundary Scan (IEEE, 1990, 2013)			
Teste de Manufatura de PCI	Melhorias de acesso aos testes	Registradores de sondagem on-chip	Faltas em pinos e integridade de circuito
IEEE 1149.4 - Barramento para testes de sinais mistos (IEEE, 2011)			
Medição de sinais analógicos	Melhorias de acesso aos testes	Chaves <i>on-chip</i>	Valores paramétricos
IEEE 1149.6 - Teste BST de Redes Digitais Avançadas (IEEE, 2003a)			
Teste de redes LVDS de alta velocidade	Teste de malhas acopladas em corrente alternada	Geradores de pulsos <i>on-chip</i>	integridade de malha
IEEE 1149.7 - Pinos reduzidos e TAP aprimorado (IEEE, 2010)			
Teste de placa e depuração de software	Acesso à teste flexível e em alta velocidade por dois pinos	SERDES, endereçamento	Contempla todos acima
IEEE 1149.8.1 - Alternância de pinos e sensoriamento sem contato(IEEE, 2012)			
Teste de interconexão de PCIM	Ligações para componentes passivos	chapas de sensoria-mento capacitivo	Circuitos abertos: AC e DC
IEEE P1149.10- TAP de alta velocidade (P1149.10, 2016)			
O mesmo que todos acima	Permutação de dados de teste em alta velocidade	Reuso de pinos I/O de alta velocidade	O mesmo que todos acima
IEEE 1500 - Teste de núcleo embarcado (IEEE, 2005)			
Teste à nível de SoC e IP	Acesso ao teste de <i>IP cores</i> em um SoC	invólucros de núcleos	Faltas no domínio digital dentro de um CI
IEEE 1687 - Acesso por Instrumentação Embarcada (IEEE, 2014)			
Teste de CI, depuração, diagnóstico	Padrão de acesso por instrumento	Cadeias de sonda re-configuráveis	Específicas do instrumento
IEEE P1838 - Acesso a teste para CIs 3D (P1838, 2016)			
Teste de integração 3DSIC	Acesso à teste através das vias TSV	O mesmo que os padrões 1500, 1149.1, 1687	Integridade das vias TSV

2.1.6 Teste Controlado por FPGA ou Processador e Instrumentação Embarcada

Conforme mencionado anteriormente, o JTAG não permite o uso de padrões de teste em velocidade nominal, deixando a classe de falhas de desempenho sob responsabilidade de testes funcionais. Atualmente, parte da indústria tem utilizado os componentes centrais de seus sistemas, como por exemplo os FPGAs e microcontroladores, como dispositivos de teste embarcado (Crouch, 2011; Wenzel, 2013), cunhando os termos *FPGA-Controlled Test - FCT* e *Processor-Controlled Test - PCT*.

Além de cobrir os testes em domínio CA, como atrasos e *crosstalk*, ainda possibilitam um acesso de baixo nível no sistema, já que esses são normalmente componentes centrais do projeto. O custo marginal é ínfimo, já que o todo o hardware de teste é reutilizado do produto e a memória de teste é apagada, voltando o dispositivo para sua funcionalidade original. Jutman et al. (2014) afirmou que ainda são poucas as empresas que fornecem ferramentas automatizadas para esta classe de testes.

O conceito de teste controlado por FPGA é possibilitado pelo uso de instrumentos embarcados, que são definidos como qualquer estrutura lógica dentro de um dispositivo cuja função é de teste, diagnóstico, *Design for Testability (DfT)*, *Design for Debug (DfD)*, *Design for Yield (DfY)*, e outros. Neste escopo, enquadram-se diversas estruturas lógicas:

- Testadores de memória e BIST/BISD;
- Analisadores de taxa de erro (BERT) em canais de comunicação;
- Geradores de padrões de teste e *buffers* de captura digital;
- Caracterizadores e Calibradores de E/S complexas;
- Testadores de barramentos de comunicação LAN, SATA, PCI, CAN, LIN, I2C, SPI e UART;
- Testadores sistêmicos e programação de memórias não voláteis;
- Instrumentos definidos por usuário.

Stollon (2011) mostra que estas estruturas de instrumentação embutida surgiram, em parte, pela necessidade de medições que causem menos interferências ou que não comprometam a integridade do sinal de barramentos de alta velocidade, internos ou externos ao CI. Dessa forma, os

fabricantes passaram a embutir núcleos de propriedade intelectual - *IP cores* - de instrumentação interna, permitindo testes não intrusivos e facilitando a validação e diagnóstico de placas.

Originalmente, cada um destes instrumentos são acessados e gerenciados por uma variedade de instrumentos externos, usando diversos mecanismos e protocolos. Isso dificulta a integração entre CIs e o reuso de tecnologia e equipamento. Logo, existia uma necessidade de padronização destes protocolos, de forma a garantir uma metodologia eficiente e organizada para a preparação de testes e para o acesso e controle destes instrumentos embarcados.

O IEEE 1687 (IEEE, 2014), também conhecido como *Internal JTAG* ou *IJAG*, foi o padrão criado para definir e descrever as interfaces de acesso à instrumentação embarcada pela porta padrão IEEE1149.1, possibilitando a realização de testes avançados e não intrusivos em uma PCI inteira pela porta JTAG. Como ilustrado na figura 6, o IJTAG não define os instrumentos ou suas funcionalidades por si, mas sim os padrões de infraestrutura de acesso e como descrever os métodos e funcionalidades dos instrumentos. Isso se consolidou em duas linguagens de descrição dentro deste padrão: uma para as características destas funcionalidades, a PDL (*procedural description language*), e outra para os requerimentos de interface com estas funcionalidades - ICL (*instrument connectivity language*). Estas duas linguagens de descrição - PDL e ICL - facilitam o reuso e a descoberta de qualquer instrumento que seja compatível com o padrão IEEE 1687, independente do seu tipo, propósito ou origem.

Dessa forma, o padrão IEEE 1687 funciona como uma extensão do IEEE 1149.1, de maneira que a porta JTAG de um CI ou uma placa possa ser usada para configurar, operar, e coletar dados de instrumentos embarcados.

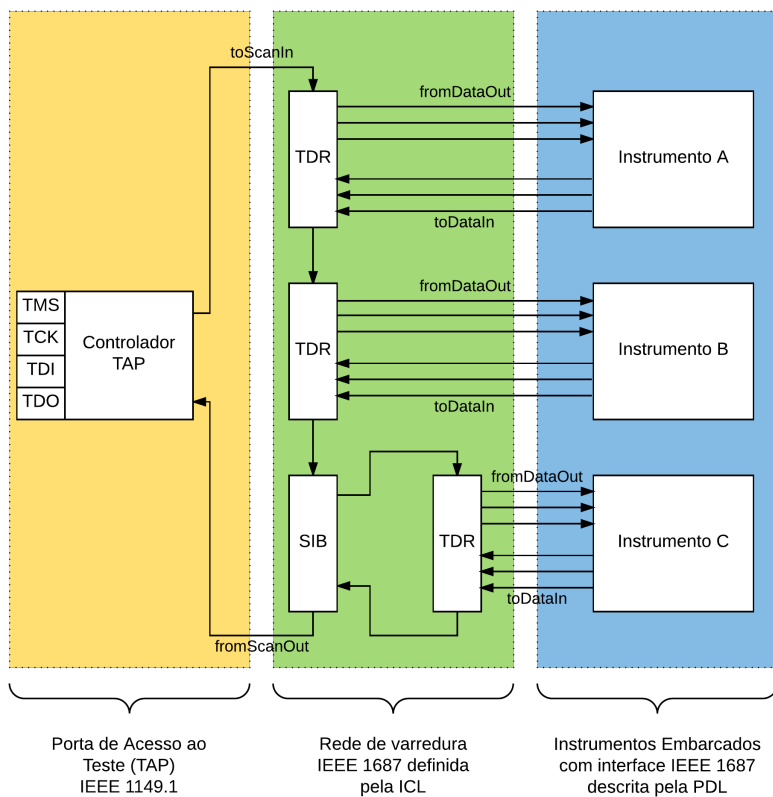


Figura 6 – Rede IEEE 1687 conceitual de múltiplas cadeias de varredura

Os FCT e PCT aliados ao IEEE 1687 oferecem uma cobertura em múltiplos domínios de falta (estruturais, CC, CA e faltas de desempenho), reduzindo ou até mesmo superando a necessidade de equipamentos de teste especializados (Wenzel, 2013). Em Jutman et al. (2014) observa-se que isso possibilita não só uma manufatura e teste de produção mais baratos e descomplicados, como também abre porta para o reuso dos testes em campo para o diagnóstico de falhas operacionais, o que não só reduz os custos logísticos de transporte de equipamentos especializados para depuração e diagnóstico, como possibilita que testes avançados sejam realizados sob condições reais de operação.

2.1.7 Teste Funcional

Define-se teste funcional como aquele que não depende da estrutura interna de DfT de um sistema, mas sim das portas de entrada e saída (Jutman et al., 2014). Outra definição, formulada pelo mesmo autor, é a de um teste que se baseia somente nas informações funcionais do sistema, sem conhecimento da estrutura interna. Essas duas definições se interseccionam na maioria das vezes.

O teste funcional serve para complementar objetivos específicos de teste, sendo normalmente empregado como etapa final de linha de produção (Jutman et al., 2014; Thibeault et al., 2006; Tumin et al., 2001). A tendência é que o teste funcional, devido aos seus custos crescentes, perca espaço com a adesão e popularização dos padrões mais avançados de varredura periférica e instrumentação embarcada, limitando-se a atingir certos objetivos específicos de teste que não podem ser cobertos por testes de varredura (Thibeault et al., 2006; Wenzel and Zimmermann, 2016).

Entretanto, existem casos onde o suporte ao IEEE 1149.1 é inexis-

tente e o teste funcional é a única opção para validação do produto. Tumin et al. (2001) afirma que, mesmo com os últimos avanços nos testes de varredura periférica, o teste funcional ainda é insubstituível. Vale lembrar da relação complementar entre essas duas categorias de teste, ilustrada pela figura 2.

Jutman et al. (2014) elenca outras aplicações de testes funcionais, dentre elas: a inspeção de recebimento de componentes de terceiros (por exemplo: uma fonte de alimentação feita por outro fabricante, ou circuito integrado de alto valor) e em casos de teste em campo do circuito eletrônico, quando se depende de equipamento externo especializado para a realização de testes ou o fabricante original do CI não disponibiliza o acesso à infraestrutura de teste interna.

Destacam-se dentre as principais vantagens do teste funcional:

- A possibilidade de execução de testes em velocidade nominal de operação, possibilitando a detecção de defeitos relacionados a temporização (Thibeault et al., 2006);
- Verificações funcionais que garantam que o projeto opera como prometido, coisa que testes estruturais e de varredura não conseguem fazer (Tumin et al., 2001).

Já entre as desvantagens:

- O processo de teste funcional em velocidade nominal é muito mais demorado, se comparado com a varredura periférica (Thibeault et al., 2006);
- O esforço e o tempo requerido para gerá-los podem ser significativos e a dificuldade cresce exponencialmente em relação ao número

de portas lógicas. Essa dificuldade persiste ainda que existam algoritmos para o teste funcional das estruturas lógicas mais comuns, como CPUs, MMUs, BPU, processadores multicore e GPUs.

Como um sistema caixa preta, o teste funcional é composto por dados de excitação, a saída resultante do teste e também um roteiro de teste. Em abordagens tradicionais, o teste funcional de interfaces com periféricos e interconexões é gerenciado por um ATE para estímulos e observação. Este esquema está representado no diagrama da figura 7. Em casos de manutenção em campo ou local de difícil acesso, pode-se utilizar uma conexão de *loopback* no dispositivo.

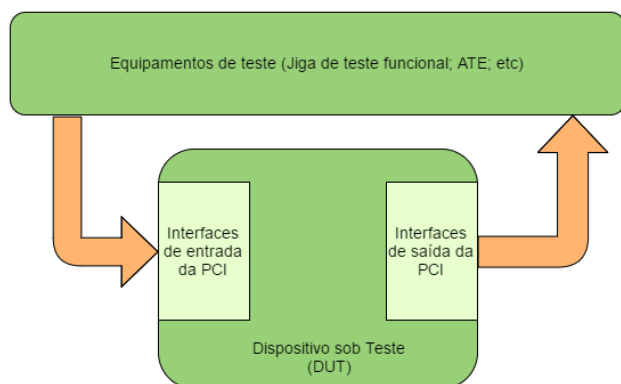


Figura 7 – Diagrama de blocos de um esquema de teste funcional

A geração do roteiro de teste geralmente é manual e os esforços de pesquisa em teste funcional se concentram em métodos de automatização da geração de testes. O trabalho de Thibeault et al. (2006) propõe uma

investigação em otimização de roteiros e reuso de padrões de teste gerados diretamente por ferramentas de validação de projeto.

Também existem trabalhos mais recentes de automatização de testes funcionais, como o de Riefert et al. (2014), que desenvolveu um método automatizado para testes funcionais em processadores utilizando métodos formais.

2.2 PADRÕES DE PROJETO EM LABVIEW

Padrões de projeto são formas bem estabelecidas para resolver problemas comumente recorrentes. Não se trata de um código fonte ou *framework*, mas de uma descrição de como se resolver um problema que pode ser aplicado em diversas situações. Padrões de projeto ganharam popularidade na ciência da computação após a obra de Gamma et al. (1994).

As vantagens de usá-los estão na facilidade dos desenvolvedores em reconhecerem no código fonte, de dispensarem a necessidade de reinventar soluções para problemas recorrentes, como também na confiança de estar aplicando uma solução bem trabalhada para um tipo de problema.

No caso específico do LabVIEW, os padrões de projeto podem ser um modelo como um *framework*, ou seja, uma base de códigos para o desenvolvimento da aplicação. Em Blume (2007) descrevem-se os principais padrões de projeto e *frameworks* da linguagem: as máquinas de estado; os laços acionados por evento; o produtor-consumidor; o tratador de mensagem em fila; a máquina de estados em fila; e o Modelo de Atores.

2.2.0.1 Máquinas de Estados Finitos

As máquinas de estados finitos (figura 8) são amplamente conhecidas e, nelas, o comportamento dinâmico do programa depende de estados

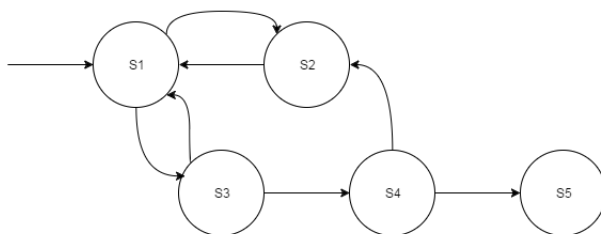


Figura 8 – Um exemplo de máquina de estados finita

cujas transições dependem de uma lógica de estados. É fundamental a criação de uma tabela de estados para uma máquina de estados efetiva.

2.2.0.2 Laços Acionados por Eventos

Nos laços acionados por eventos (figura 9), diferentemente do paradigma procedural, a execução do programa é decidida em tempo de execução: o programa permanece em estado de espera até a ocorrência de um evento e seu despacho para um tratador. Esta técnica poupa tempo de CPU e é uma boa alternativa ao uso de *pooling*, além de poder ser aplicada em qualquer sistema de acionamento de processos escravos.

2.2.0.3 O Produtor - Consumidor

O produtor-consumidor (figura 10) é normalmente aplicado quando se precisa executar tarefas assíncronas e comunicar entre elas sem perda de desempenho, caso uma dependa da outra. Este padrão tem por característica a relação de mestre/escravo entre os laços e a independência no fluxo de dados entre eles. Uma tarefa produzindo dados ou comandos e uma ou mais tarefas consumindo o que a produtora gera. A comunicação

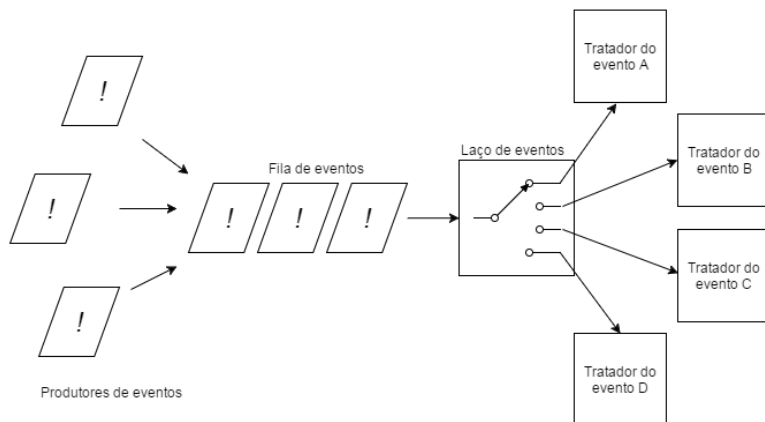


Figura 9 – Laço acionado por eventos

entre as tarefas é realizada por filas, ou seja, os dados são enfileirados e desenfileirados em um *buffer*, em esquema conhecido como *first-in-first-out* - *FIFO*.

2.2.0.4 Tratador de mensagens em fila

O tratador de mensagens em fila - TMF (figura 11) ou, em inglês, *Queue Message Handler* ou *Queue-driven State Machine*, é uma extensão do produtor-consumidor, possibilitando que processos escravos possam atuar também como produtores e comunicar entre si. Normalmente apresenta-se como na figura 11, com um laço mestre acionado por eventos criando tarefas, um processo principal e subprocessos auxiliares. O TMF não só é um padrão de projeto, mas também a arquitetura base de muitas aplicações de média a grande complexidade. É usado em aplicações que exijam interfaces de usuário responsivas, aplicações multitarefa e o

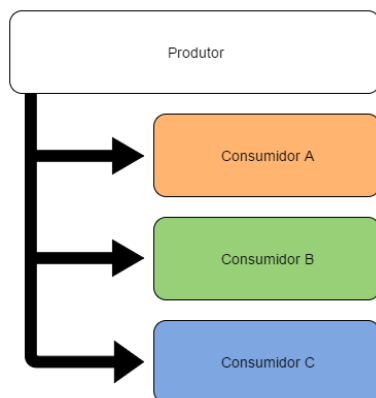


Figura 10 – Diagrama do produtor-consumidor com 4 laços independentes comunicando por uma fila

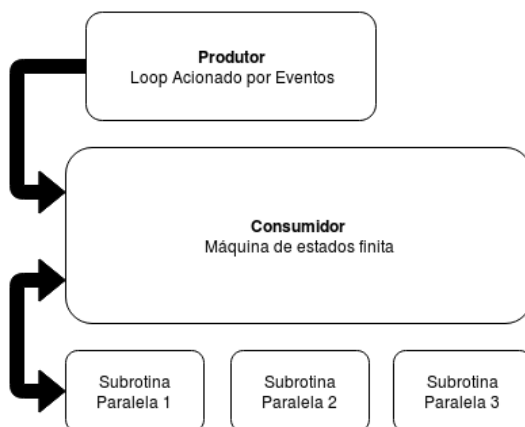


Figura 11 – Diagrama do tratador de mensagens em fila

desacoplamento de processos.

2.2.1 Framework de Atores

Na programação imperativa, assim como na maior parte dos paradigmas e linguagens de programação, as *threads* são a solução tradicional para problemas de concorrência. Entretanto, programação concorrente baseada em *threads*, *locks* e estados compartilhados são ditas difíceis de fazer e propensas a erro (Erb, 2012).

Em 1973, Hewitt et al. (1973) criou o modelo de atores como alternativa às *threads* em problemas de computação concorrente e sistemas distribuídos. Sua premissa é a de que qualquer computação fisicamente possível pode ser diretamente implementada usando Atores (Hewitt, 2010).

A abordagem no modelo de atores é totalmente diferente, já que retira inteiramente a noção de estado compartilhado. Ainda é possível que haja estados, entretanto, estes estão exclusivamente acoplados a uma única entidade, chamada Ator.

O modelo tem sido usado tanto como uma estrutura base para compreensão de problemas de concorrência, como também como base teórica para diversas implementações práticas de sistemas concorrentes. O crescimento de sistemas concorrentes massivos de computação em nuvem e processadores *multi-cores* tem aumentado o interesse por este modelo. Exemplos de sistemas modelados pelo sistema de atores incluem: sistemas de email, Serviços Web e Objetos com *locks*.

2.2.1.1 Conceitos fundamentais do modelo

Um ator é a unidade primitiva de computação. Ao receber uma mensagem, um ator pode concorrentemente (Hewitt and Zenil, 2013):

- Criar um finito número finito de novos atores;
- Enviar finitas mensagens para outros atores;
- Designar como tratar a próxima mensagem que receber, alterando seu estado interno.

Para comunicação, o modelo de atores usa a troca de mensagens assíncronas, atribuindo a cada ator a sua caixa de mensagens (figura 12), local onde as mensagens são armazenadas enquanto o ator processa a mensagem anterior. Num sistema de atores, tudo o que um ator sabe do outro é o endereço de sua caixa de mensagens.

O ator sempre trabalha em resposta às mensagens que recebe, tratando-as sequencialmente, uma de cada vez (Erb, 2012). Isso significa que, para executar múltiplas mensagens de forma concorrente, será necessário que antes se crie um ator para cada uma delas.

O ponto central do modelo de atores, e o que o diferencia das *threads*, é que não existe compartilhamento de memória ou estado, sendo completamente isolados uns dos outros. Erb (2012) enfatiza que a implementação de estados privados é fundamental para um sistema no modelo de atores.

2.2.1.2 Distribuição e Escalabilidade

A natureza assíncrona e sem compartilhamento de estados do modelo de atores assemelha-se à comunicação em rede, o que permite a implementação de sistemas distribuídos. De fato, muitas das implementações do modelo de atores oferecem essa funcionalidade e possibilitam a fácil extensão do sistema para múltiplas máquinas em um sistema concorrente e distribuído (Erb, 2012; Hewitt, 2010).

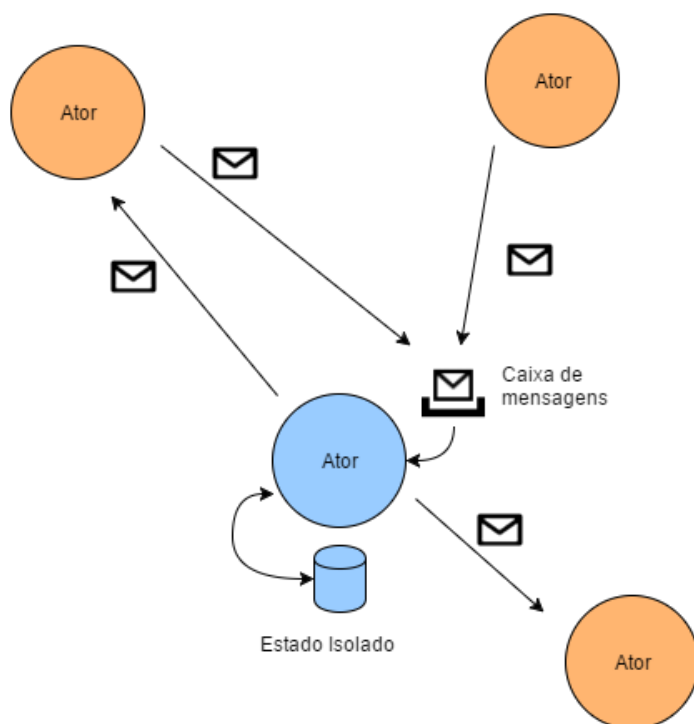


Figura 12 – Um exemplo de uma rede com alguns atores, com destaque para um ator, sua caixa de mensagens e seu estado interno. Adaptado de Erb (2012)

Pela sua simplicidade e isolamento entre atores, o modelo permite o escalonamento de aplicações pelo instanciamento e replicação de atores e distribuição por outras máquinas.

2.2.1.3 Robustez e tolerância a falhas em tempo de execução

Sistemas concorrentes baseados em *threads* dificilmente conseguem ter boas soluções de tolerância a faltas por causa do não-determinismo associado ao compartilhamento de estados e apreensão de recursos (*locks*) (Erb, 2012).

O modelo de atores normalmente adota uma política de *deixar quebrar*. Isso porque a maneira isolada e de não compartilhar nada dos atores permite que um ator quebre sem influenciar os outros atores. Além do mais, pode-se usar atores em hierarquia para supervisionar possíveis falhas de atores *filhos*, conforme visto em Erb (2012). Isso permite criar sistemas que se recuperem sozinhos, de forma que, se um ator do sistema quebrar, o sistema tem como detectar o problema e colocar o sistema em um estado consistente novamente.

A política de deixar quebrar é enxuta, já que dispensa a necessidade de criar infinitos tratadores para muitos problemas, que talvez nunca ocorram, e a implementação de outras políticas de Programação Defensiva.

2.2.2 Implementação em LabVIEW

O modelo de atores possui implementações em inúmeras linguagens, tanto no paradigma orientado a objetos, como também no paradigma funcional. Em LabVIEW, sua implementação foi realizada por meio de um *framework*, ou seja, uma estrutura básica de programa na qual uma aplicação pode ser construída.

No texto de Smith and Mercer (2011), constata-se que esta implementação é mais restrita do que em outras linguagens. Primeiro que, no *framework* de atores, a comunicação de um ator é limitada a mensagens para o ator que o criou, atores que ele criou ou para si mesmo (ver figura 13). Essa hierarquia, por um lado, obriga a criar atores supervisórios, mas também restringe a comunicação entre a rede de atores.

A implementação dos atores não são estruturas leves e *atômicas* como proposto por Hewitt et al. (1973), mas objetos sobrecarregados por diversas camadas de sub-rotinas². Isto impede a construção e destruição irrestrita de atores. Todavia, o *framework* ainda detém grande flexibilidade de arranjos de atores e hierarquias de programa.

Um ponto curioso do *framework* é o suporte às mensagens síncronas entre atores (Smith and Mercer, 2011), funcionalidade que cria um compartilhamento de estado e rompe com a proposta do modelo.

² Uma sub-rotina em LabVIEW é também referida como *subVirtual Instrument* ou *subVI*.

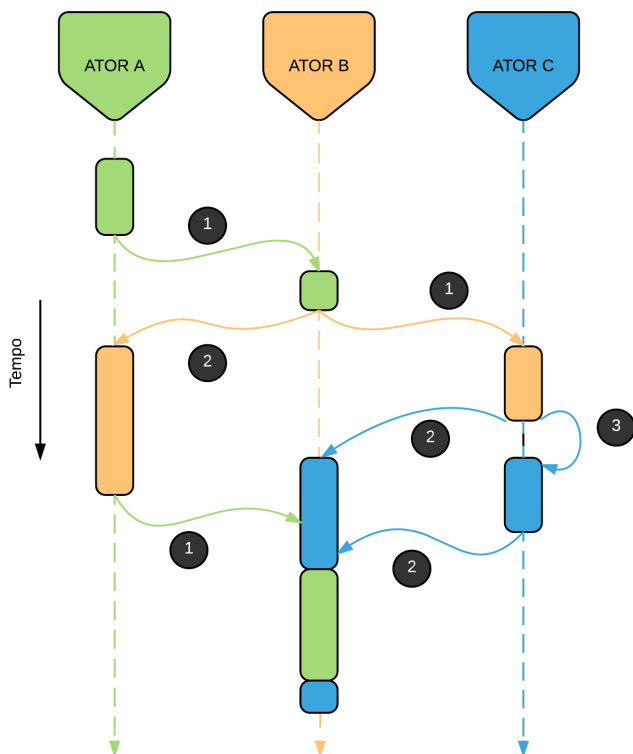


Figura 13 – Mensagens no *framework* de atores em LabVIEW: (1) atores podem mandar mensagens para seus filhos; (2) atores podem mandar mensagem para os seus pais; (3) atores podem mandar mensagem para si mesmos

3 ESPECIFICAÇÃO DO PROGRAMA

A partir do estado da arte em teste e diagnóstico estudado, nota-se que a configuração de teste de manufatura estudada pode ser aprimorado em diversos aspectos, principalmente em hardware. Entretanto, este trabalho se limita ao desenvolvimento de um novo programa de teste de manufatura e decidiu-se limitar o escopo do trabalho a um único produto e manter a mesma configuração de teste: bateria de testes, roteiro e a mesma jiga. No início deste capítulo são detalhados o objeto de teste, o acesso ao teste que ele suporta, a jiga e instrumentos de medição utilizados, enquanto o final se reserva para discutir as especificações de software.

3.1 O DISPOSITIVO SOB TESTE

O dispositivo sob teste é um módulo de telemetria para sistemas de infraestrutura de água e energia, mais especificamente para atender o setor de distribuição de energia. A funcionalidade principal do produto é o módulo M2M de comunicação móvel (High Speed Packet Access - HSPA) com interpretador JavaTM. Possui como funcionalidades secundárias: sensoramento de tensões trifásicas, atualização remota de *firmware*¹, sistema de alimentação emergencial (bateria ou supercapacitor), portas digitais para detectar a abertura de gabinete, porta RS-232 para comunicação com religadores e outras funcionalidades que podem ser inseridos à pedido do cliente. Sua PCIM pode ser vista na figura 14.

¹ *Over-the-air provisioning* é um método remoto de atualização de software. Isso é normalmente implementado no próprio *bootloader* do sistema (Beningo, 2013).

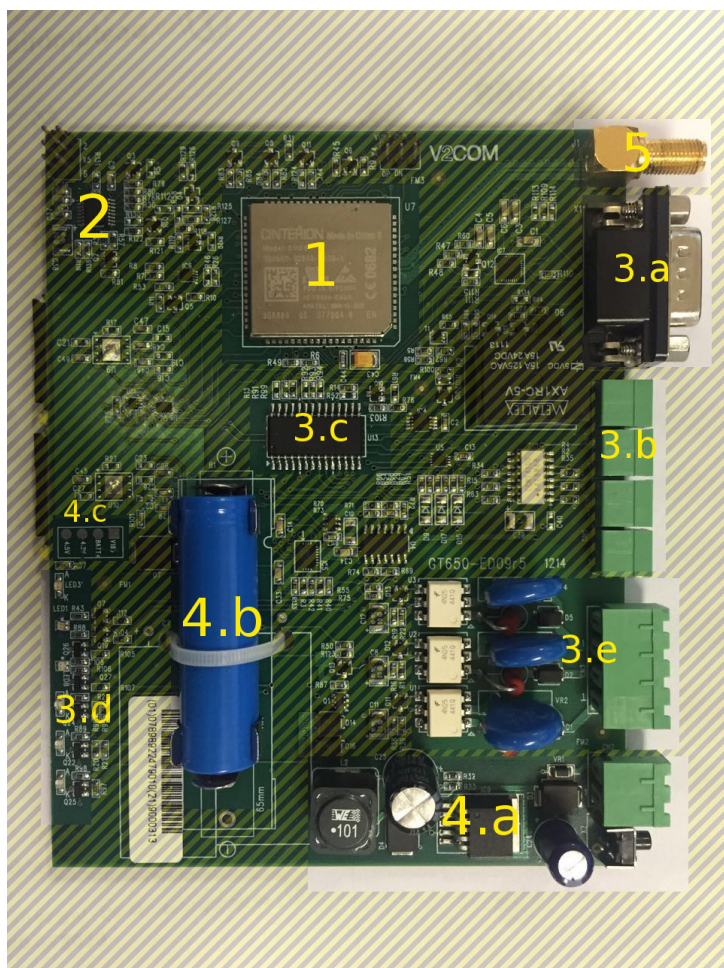


Figura 14 – Foto do GT650 com destaque aos componentes da placa: o modem EHS6 (1); o microcontrolador monitor watchdog (2); o conector DB9 com duas portas RS232 (3.a); as E/S isoladas (3.b); o expensor de E/S por I²C (3.c); o painel de LEDs indicativos (3.d); os detectores de tensão trifásica (3.e); o conversor buck (4.a); a bateria e o seu sistema de gerenciamento (4.b); os pontos de teste de alimentação (4.c); e o conector da antena (5) (foto do autor).

3.1.1 Módulo M2M

O *modem* EHS6 da Cinterion (ponto 1 da figura 14) é o processador principal da placa, com comunicação TCP/IP via GSM/3GPP/GPRS, Interpretador Java e outros recursos comuns, como interface serial e I²C. Segundo o *datasheet* do fabricante (Cinterion, 2016), o circuito integrado oferece:

- Comunicação de dados 3G (HSPA): 800/850/900/1900/2100 MHz
- GPRS/EDGE Class 12: 850/900/1800/1900 MHz
- Suporte a Java™ ME² 3.2, com suporte a *multithreading* e execução de múltiplas aplicações, além de 6 MB de memória RAM e 10 MB de memória Flash;
- Pilha de protocolos TCP/IP embarcada e acessível por comandos AT, incluindo conexão segura via TLS/SSL; serviços DNS e Ping; Cliente FTP e HTTP;
- Interface USB 2.0 HS e duas portas de interface modem serial;
- 16 GPIOs compartilhadas com as portas de comunicação, PWM e outras.
- ADC e interface I²C;
- Atualização de *Firmware Over-the-air* (FOTA).

O fabricante não fornece informações internas sobre a arquitetura, como também nenhum meio de acesso à estrutura interna do módulo,

² Java Micro Edition é uma plataforma Java para sistemas embarcados

como uma porta JTAG. Como contrapartida, a empresa oferece garantia de funcionamento, assim como suporte técnico completo. As consequências dessa política não são necessariamente positivas, conforme segue:

Primeiro, impede a realização de testes estruturais e verificações mais profundas nos módulos, tanto na produção, quanto na manutenção, e torna todos os processos dependentes de agentes externos, de forma que o teste se limita à verificação de interconexões do *modem* com seus periféricos, em um esquema similar aos testes centralizados por processador, vistos em 2.1.6. Outra dificuldade do EHS6 é seu encapsulamento LGA, que impede a completa verificação de solda por inspeção visual. Isso pode ser contornado, caso a empresa responsável pela montagem possua equipamentos de inspeção radiográfica.

Segundo, que para todo e qualquer problema físico ou de *firmware* que ocorrer com estes módulos, é necessário o envio ao fabricante original para solução, muitas vezes dependendo do suporte deste fornecedor em outro país. Por um lado, isto retira da empresa solicitante a responsabilidade de resolver problemas de baixo nível, assim como a necessidade de manter pessoal e equipamento especializado para analisar este tipo de falha. Por outro lado, trabalhar com componente e *firmware* fechados deixa a produção sujeita a problemas invisíveis e cria uma situação de dependência com o OEM.

Por estas limitações no teste estrutural, o escopo de teste elaborado é restrito às verificações funcionais administradas pelo próprio produto.

3.1.2 Monitor Watchdog

O MC9S08QG8 é um microcontrolador simples de 8 bits (figura 14, no ponto 2) e é responsável por desligar o *modem* por duas maneiras: a

primeira pelo temporizador *watchdog* e a segunda, pelo botão liga-desliga.

Também é armazenado nele o número de série da placa. Dessa forma, a empresa consegue separar o rastreamento dos *modems* e das *placas*, podendo intercambiá-las conforme necessário.

Por estar conectado ao EHS6 como escravo I²C, este componente é acessado e testado pelo mesmo. No teste funcional são realizadas as seguintes rotinas: captura do número de série, teste do botão liga-desliga e teste do temporizador *watchdog*.

3.1.3 Interfaces de usuário e interconexões externas

Por ser uma solução de comunicação para outros equipamentos, o GT650 não possui interfaces digitais e analógicas muito complexas, exceto pela comunicação de dados via GSM, que será descrita à parte. Como pode ser visto na figura 14, a placa possui:

- Duas portas RS232³;
- Três portas de entrada digitais opto isoladas;
- Um relé de saída;
- Seis LEDs de sinalização;
- Uma porta para conferência de tensão trifásica.

Vale mencionar que as entradas e saídas digitais, assim como os LEDs, são controlados pelo *modem* através do expansor de GPIO por endereçamento I²C. Assim como no caso do microcontrolador de *watchdog*, as verificações funcionais são acionadas pelo EHS6.

³ Uma é usada para comunicação por comandos AT e a outra para acessar os periféricos na porta I²C

3.1.4 Conector e transmissor de radiofrequência (RF)

O módulo não possui antena integrada e possui um conector SMA para a conexão de antenas externas. Neste caso, testa-se a potência do transmissor de radiofrequência através de um instrumento específico mencionado na seção 3.3.2.

3.1.5 Alimentação, bateria e BMS

O sistema de alimentação consiste em um conversor CC-CC de 12 Volts com saídas de 3,3 V e 5 V (figura 14-4.a) e uma bateria de lítio de 4,8 V, em caso de faltas na rede (figura 14-4.b).

Para testar este sistema, foram separados quatro pontos de teste na placa, como visto no ponto 4.c da figura 14: 5V, BATT+, 3V3, e 1.8V. Atualmente, o teste é realizado manualmente com o multímetro, ponto por ponto, mas enviando os dados de leitura pelas portas seriais do computador.

3.2 COBERTURA E ROTEIRO DE TESTE

O roteiro de teste consiste numa verificação predominantemente funcional da placa do produto. Por não possuir um volume de produção que justifique a aquisição de um ICT, não possuir suporte ao IEEE 1149.1, como também por não conter nenhuma lógica programável no produto, o teste estrutural se limita às verificações realizadas na empresa de montagem, por verificações de solda e interconexões por AOI ou AXI.

Essa estratégia predominantemente funcional consegue atingir um bom controle de qualidade dos lotes, mas fracassa no diagnóstico de falhas e detecção de causas raiz.

A bateria de testes do produto é representada pelo fluxograma da figura 15. Nota-se que este roteiro é, em sua totalidade, sequencial, sendo o teste funcional das bandejas do cartão SIM a única etapa executada de forma concorrente.

3.3 A JIGA E OS INSTRUMENTOS DE TESTE

A jiga utilizada foi desenvolvida para o teste funcional e provê interconexão para as interfaces externas do produto, exceto para o transmissor de radiofrequência. Fornece botões de interface para o operador controlar estímulos simples de sinais lógicos nas entradas e emular tensões trifásicas na porta respectiva. Ela também oferece um ponto de conexão do terra da placa para utilização do multímetro digital.

3.3.1 Multímetro

O multímetro utilizado é um ICEL MD-6400, com suporte de comunicação por uma porta serial RS232. A figura 16 exhibe o protocolo que, a cada 250 ms, envia pacotes de 14 bytes de dados a uma *baudrate* de 9600.

3.3.2 Medidor de potência do transmissor

O instrumento de medição de potência RF utilizado foi o NI 5680, da National Instruments, que mede potência RMS de sinais até 6 GHz, em larguras de banda bem estreitas (10-100 Hz de banda), e um limites de potência de entrada de -40 dBm a +23 dBm.

3.4 REQUISITOS DE SOFTWARE

Requisitos de software são objetivos ou restrições estabelecidos pelos usuários e são divididos em requisitos funcionais e não funcionais (Bourque et al., 2014).

Requisitos funcionais definem as funcionalidades específicas que o programa deve oferecer, não necessariamente tratando das interações com o usuário, mas também como funcionalidades ocultas como, por exemplo, uma API ou interações com hardware.

Requisitos não-funcionais são qualidades gerais de um programa como custo, usabilidade, modularidade e vários outros. Normalmente são requisitos difíceis de validar por sua natureza qualitativa.

A partir do contexto apresentado, foram levantados os seguintes requisitos não-funcionais do software:

- O programa deverá ser implementado em LabVIEW;
- Modularidade do programa e reusabilidade de código;
- Permitir o uso de concorrência e paralelismo na bateria de testes;
- Escalabilidade e flexibilidade de software para acompanhar alterações de hardware e da jiga de teste;
- Interface de usuário ergonômica e intuitiva.

Já os requisitos funcionais de software acompanham o que é testado pelo roteiro de testes anterior a este trabalho:

- Leitura e armazenamento do número de série colado na placa por um leitor de código de barras;

- Comunicação com o DUT via serial;
- Teste de comunicação das duas portas seriais;
- Verificação do modelo de *modem*;
- Verificação da versão da *firmware* do *modem*;
- Armazenamento o IMEI⁴ do *modem*;
- Configuração das GPIOs do *modem* conforme o perfil da aplicação;
- Abertura, seleção e teste das duas bandejas de cartão SIM⁵;
- Configuração e teste de comunicação I^2C ;
- Configuração da operação do expansor de IO, MCP23018: interrupção espelhada, não incremento de ponteiro, interrupção push-pull;
- Configuração e teste do BQ24070, carregador de bateria e gerenciador de energia do sistema;
- Teste de leitura da bateria;
- Teste do MC9S08QG8, microcontrolador *watchdog* e teste do botão liga e desliga;
- Leitura do número de série armazenado no MC9S08QG8, e validação deste com o número de série colado na placa;
- Teste de comunicação e leitura do sensor de temperatura LM75;

⁴ *International Mobile Station Equipment Identity* ou Identificação Internacional de Equipamento Móvel é um número de identificação global e único para cada telefone ou *modem* celular.

⁵ *Subscriber Identity Module*

- Teste de finalização da sessão do terminal na porta serial ;
- Testes dos barramentos de tensão da placa: 4,5V, 4,2V, BATT++ e 1,8V;
- Teste do painel de LEDs;
- Teste de detecção de retirada e inserção da fonte de alimentação;
- Teste das 3 portas digitais;
- Teste de acionamento e desarme de relé;
- Teste da porta de tensão trifásica - detecção de presença e ausência;
- Teste de interrupção de desligamento e reinicialização do sistema;
- Teste de potência de sinal do transmissor de radiofrequência;
- Registro de execução de teste em arquivo de texto e resumo de teste em formato estruturado;
- Cliente WEB para envio automático à base de dados de produtos;
- Notificação para o operador sobre sucesso ou falha do teste, assim como possibilidade de reteste.

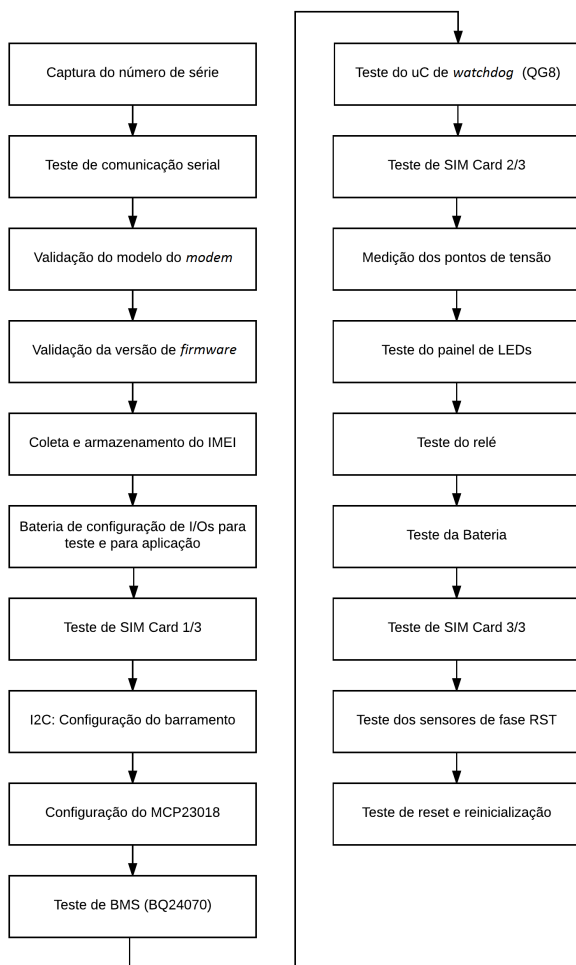


Figura 15 – Fluxograma da bateria de testes

Pacote de 14 bytes enviados a cada 250 ms:

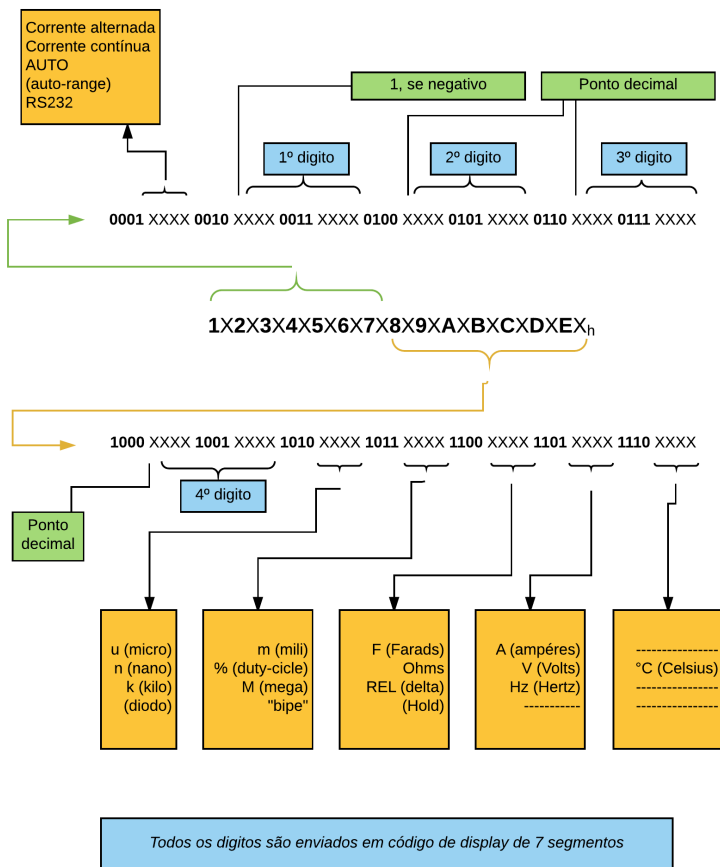


Figura 16 – Estrutura de pacote de comunicação do multímetro



Figura 17 – NI 5680 - instrumento utilizado para medição de potência RF

4 METODOLOGIA

Ao considerar os requisitos de escalabilidade, concorrência, paralelismo e modularidade, nota-se que dentre os padrões de projeto revisados, o *framework* de atores é o que melhor se ajusta como base para o programa, principalmente porque permite a criação de componentes isolados, naturalmente concorrentes e aplicáveis em diversas configurações de software e hardware. Além disso, é uma estrutura escalável, tanto na criação de mais atores na mesma máquina, como também para sistemas distribuídos.

Para a elaboração do modelo do sistema, utilizou-se de técnicas em Linguagem de Modelagem Unificada (do inglês *Unified Modeling Language - UML*) para a sua representação e documentação (Rumbaugh et al., 2004). Na etapa de implementação dos atores foram utilizados diversos padrões de projeto: máquinas de estado, tratadores de eventos, produtor-consumidores, e outros.

4.1 MATERIAL UTILIZADO

- Computador com uma instalação LabVIEW e com 3 portas RS-232;
- Multímetro ICEL MD-6400;
- Um NI 5680 (*powermeter* USB);
- Um GT650 e uma jiga de teste;
- Bibliotecas do Labview do modelo de atores, de comunicação serial e driver do *powermeter*.

5 MODELO E IMPLEMENTAÇÃO

5.1 MODELO DO SISTEMA NO *FRAMEWORK* DE ATORES

Foram determinados cinco módulos, conforme as competências de teste necessárias: comunicação com o módulo; leitura e validação do multímetro; teste de potência de RF; registro da execução de testes; e supervisão; Para cada um destes módulos, foi criada uma classe de ator.

Dessa forma, permitem-se diversos arranjos de software e flexibilidade de adaptação frente à configuração física de teste utilizada. Para o caso da configuração de teste atualmente utilizada e selecionada para este trabalho, foi elaborado um arranjo em topologia em estrela, conforme visto na figura 18. A escolha desta topologia foi consequente às próprias limitações do *Actor Framework* em termos de comunicação entre atores, que não permite a comunicação de atores que não sejam *pais* ou *filhos*. A principal vantagem desta topologia é o Controlador como entidade supervisora natural do sistema, para detecção e tratamento de atores em falha de execução.

O Ator ModCOM é responsável pela comunicação com o DUT através de uma porta RS-232. Cumpre funções como enviar comandos de teste e verificar as respostas recebidas do dispositivo sob teste. Idealmente, este ator interpretaria trechos do roteiro da bateria de testes (em um arquivo xml), retornando o status da execução para o controlador. Todavia, devido ao curto prazo e simplificação do protótipo, a função de interpretação de roteiro foi deixada de lado. Dessa forma, o roteiro foi codificado diretamente no código fonte do programa, impossibilitando alterações de roteiro após a compilação.

O ator DMM é incumbido pela leitura de dados do multímetro atra-

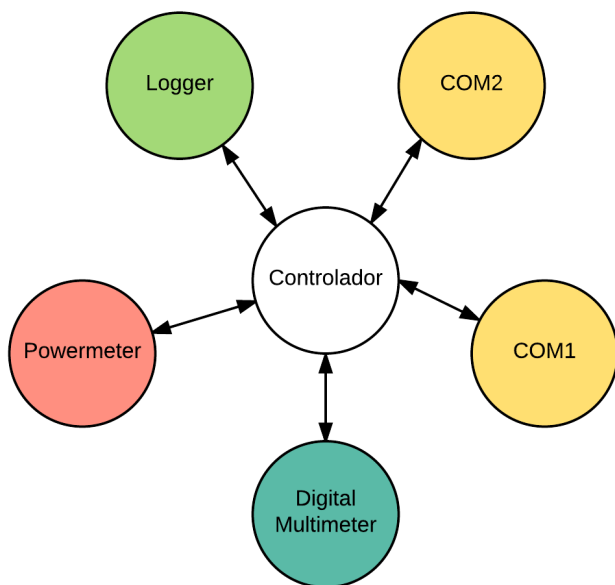


Figura 18 – Modelo do programa de teste reimplementado à partir do *framework* de atores, e adequado para a jiga de teste legada

vés da porta RS-232. A leitura dos pacotes do instrumento (figura 16), sincronização e conversão de dados é realizada por ele, como também a interface de usuário para o operador do multímetro e a validação de medidas comparadas com a tabela de referências.

O registrador, ou *Logger*, é o ator responsável pelo registro da execução e resultados das baterias de testes. Dentre as suas incumbências estão: gerar um arquivo de registro completo do teste para fins de depuração

e diagnóstico; gerar um arquivo de registro resumido para o controle de qualidade da produção e o envio direto dos registros para a base de dados de produção.

O Ator *Powermeter* detém o controle do medidor de potência RF, também referenciado como *Powermeter*, responsável por toda a interface e configuração, e coleta de medições do NI 5680.

O *Controller* ou controlador, como o próprio nome define, detém a responsabilidade sobre o fluxo de execução do programa, criando e destruindo os outros atores-módulos e fazendo a comunicação entre seus atores filhos. Além disso, detém a interface de usuário principal do programa.

A relação entre as classes criadas com as classes base do *framework* podem ser vistas no diagrama de classes UML da figura 19. Nota-se que as mensagens entre atores possuem uma superclasse em comum e que, para cada ator, existem diversas classes de mensagens associadas a ele. Cada classe de mensagem se relaciona com um método interno do ator, funcionando como a interface pública deste ator. Para melhor legibilidade, neste diagrama foram suprimidas as classes de mensagens, assim como muitas das VIs (*Virtual Instruments*) das classes criadas.

5.2 IMPLEMENTAÇÃO

Análoga à programação orientada a objetos, a implementação de um ator envolve a criação de métodos públicos e privados, sendo que todas as suas *subVI* são privadas por padrão e podem ser acessíveis de maneira pública, se for criada uma classe de mensagem invocando-a. Dessa forma, o conjunto de classes de mensagens relacionadas àquele ator funciona como sua interface pública.

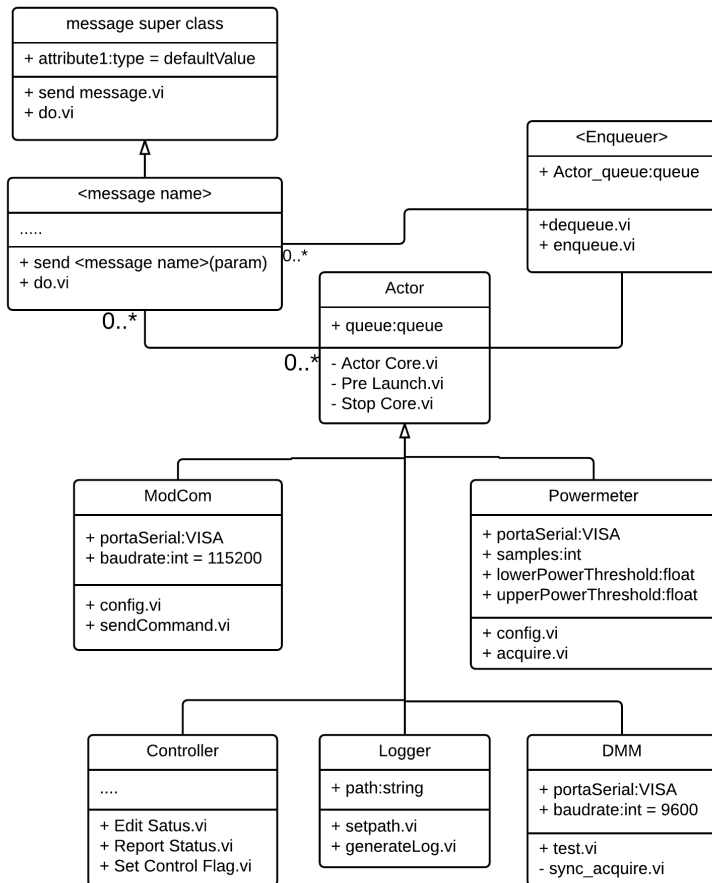


Figura 19 – Diagrama UML de classes simplificado retratando as classes do *framework*, em conjunto com as classes de atores propostas e implementadas neste trabalho

Todas as implementações dos módulos herdam das classes `actor` e `actor message` do *framework* de atores. Todo ator tem que implementar a função `actor core.vi` para poder alterar o comportamento do ator. Como o próprio nome descreve, esta é a função núcleo do ator, responsável por sua inicialização e de seu tratador de mensagens. Também é possível e recomendável inserir nesta `vi` outras sub-rotinas, como, por exemplo, tratadores de eventos.

A interação dos atores filho com o `Controller` é realizada através de duas de suas classes de mensagens: `Set Control Flag`, que informa o status final de execução do teste e `Report Status`, que envia dados brutos do teste.

5.2.1 Ator de comunicação serial

A comunicação serial em LabVIEW é realizada através de portas VISA (*Virtual Instrument Software Architecture*) e são o padrão da National Instruments para configuração, programação, comunicação e depuração de sistemas de instrumentação. Dentro da VISA existe uma biblioteca própria para comunicação serial, que foi usada para criar as *SubVI* base deste ator.

O funcionamento deste ator é viabilizado pela `serialcore.vi`, a *subVI* que envia comandos e captura respostas do dispositivo sob teste, como também confere respostas por padrões de *strings*. É invocada diversas vezes por todos os outros métodos do ator. Esse funcionamento interno, assim como a relação do ator com agentes externos, estão ilustrados no diagrama da figura 20.

Todo o roteiro está escrito diretamente no código fonte do programa e as mensagens do `Controller` para o `ModCOM` consistem somente

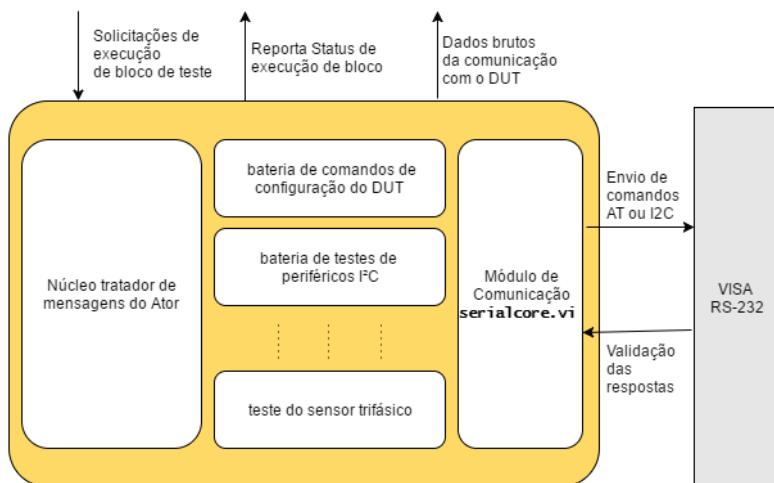


Figura 20 – Diagrama de blocos do ModCOM, ator de comunicação serial

do pedido de execução de determinado bloco de teste interno. Devido à falta de generalização de sua interface, este ator foi de longe aquele com o maior número de classes de mensagens. Para retornar dados brutos da comunicação com o módulo, o ator usa a mensagem *Report Status do Controlador*, que decide sobre a reexecução do bloco ou o término do teste.

Por trabalhar com duas portas seriais, foram instanciados dois atores: para comunicar com o módulo por comandos AT e outro para acessar periféricos conectados no barramento I²C.

Apesar de não possuir uma interface gráfica de usuário própria, alguns métodos deste ator invocam as suas próprias. Destaca-se entre elas o teste de painel de LEDs, aonde foi empregado um painel esquemático. Isso será mencionado na sessão 5.2.6, sobre ergonomia.

5.2.2 Ator Multímetro

A implementação desta classe de ator, denominado DMM, começou pela função de aquisição e sincronia de *frames*, acondicionados dentro da VI `sync_acquire.vi`. Após a criação de uma interface de leitura serial do multímetro, foi criada uma máquina de estados para a sincronização dos *frames*, já descritos pela figura 16.

Com a sincronização funcionando, desenvolveu-se um conjunto de SubVI para a conversão dos *frames* em um *cluster* com medidas e todas as informações adicionais que o multímetro fornece. Isso envolveu a decomposição dos *frames*, conversão de código de sete segmentos em inteiros e associação de unidades de medida.

`test.vi` é a segunda VI fundamental deste ator, que checa as medidas recebidas de `sync_acquire.vi` e provê uma interface gráfica de usuário para o operador de teste. A comunicação entre as duas VIs é feita através de uma fila. Sua interação dentro do ator é ilustrada pela figura 21.

A checagem de medidas de `test.vi` é análoga a um algoritmo de *debounce* de botão analógico e funciona em três estados:

1. Aguardar o valor da medida ficar dentro dos limites de tolerância para o valor de referência daquele ponto de medida. Exceção esperada: *timeout*.
2. Certificar-se de que a medida se mantém dentro da faixa de tolerância por um tempo especificado. Em caso de sucesso, ir para o estado 3 ou lançar uma exceção de erro de medida.
3. Em caso de exceção, terminar o teste como reprovado ou permitir o reteste. Do contrário, passar para o próximo ponto de teste ou, se for o último ponto, aprovar a bateria de testes.

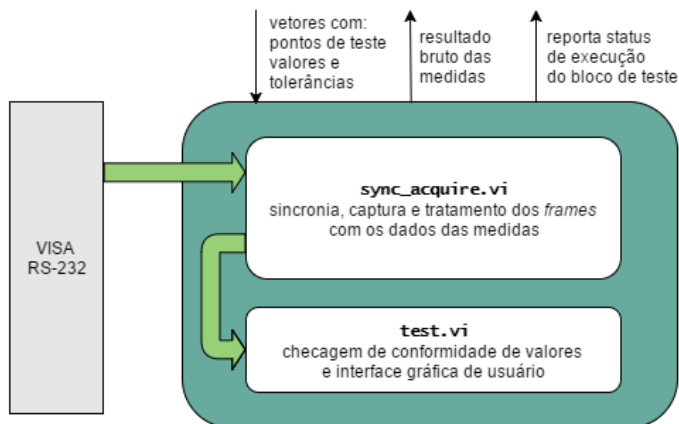


Figura 21 – Diagrama de blocos do DMM, ator de aquisição de medidas do multímetro digital

Em caso do multímetro estar em um modo de operação incorreto, `test.vi` lança uma exceção e notifica o operador.

A interface gráfica de usuário (GUI) é exibida na figura 22, onde o operador pode acompanhar as medições e qual ponto de teste deve deixar a ponta de prova. Também é notificado por sinais sonoros quando o ponto de teste é aprovado ou reprovado.

Em termos de configuração, este ator precisa receber as seguintes informações: porta serial utilizada; tempo de estabilização de medida; número de tentativas; e o *array* de pontos de teste com os seguintes atributos para cada ponto: *limiar máximo*; *limiar mínimo*; *unidade de medida*.

Assim como o ator `ModCOM`, o DMM utiliza-se do conjunto de mensagens do `Controller` para reportar o parecer final das medições, assim como os dados brutos de teste.



Figura 22 – Painel da VI de aquisição de medidas do multímetro digital

5.2.3 Registrador de testes

O gerenciador de registros, por sua baixa complexidade, foi implementado utilizando-se poucas VIs (figura 23). Este ator recebe do controlador de teste dois conjuntos de dados diferentes: os dados brutos da execução de teste e o cluster com os dados de teste resumidos. A VI principal é `generateLog.vi`, que aglutina as conversões dos dados resumidos para JSON (*JavaScript Object Notation*) e as dos dados brutos em documento de texto.

O registro com os dados brutos de execução contém o despejo da troca de mensagens pela serial, das medidas de tensão e potência, as exceções lançadas durante o teste e as métricas de desempenho. São dados necessários para a depuração e diagnóstico de problemas na placa e para

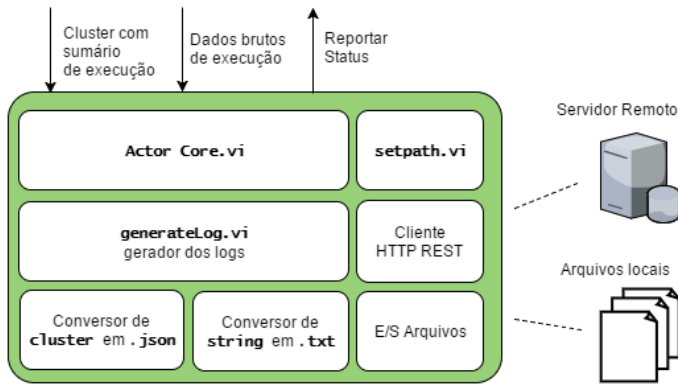


Figura 23 – Diagrama de blocos do logger, ator atribuído de gerar arquivos de registro das baterias de testes

melhor efetividade no retrabalho, sendo usados tanto em produção quanto em manutenção.

A segunda classe de registro é um resumo estruturado da bateria de testes, o diagnóstico geral da placa e outras informações importantes para o controle e rastreamento dos produtos, requisito necessário para otimizações do processo produtivo, como também do próprio produto. Recebe o *cluster* com os dados do DUT e resultados da bateria de teste e transforma em um objeto JSON.

Quanto ao armazenamento, o ator suporta que seja feito tanto em servidor remoto, através de uma API REST, como também em pastas locais, através da biblioteca padrão de E/S de arquivos. O uso de arquivos locais é importante para o diagnóstico e depuração no local de fabricação, como também em situações sem conexão com a internet.

5.2.4 Sensoriamento de potência do *front-end* RF

O desenvolvimento deste ator foi simplificado com uso das bibliotecas e *drivers* fornecidas pelo próprio fabricante. Estas bibliotecas possuem VIs de fácil utilização para configuração e aquisição de medidas e foram utilizadas na criação de `acquire.vi`, a VI principal deste ator. Esta VI recebe como entrada: frequência de medição, largura de banda, tamanho da janela de aquisição, número de amostras e limites máximo e mínimo de potência (em dBm). Seu funcionamento interno consiste em um laço simples de aquisição e comparação com a referência especificada. `config.vi` configura os dados internos do ator, como a porta USB utilizada. O ator pode ser visto na figura 24.

5.2.5 Controlador

O `controller`, módulo de supervisão e controle de execução, é composto pelo módulo de inicialização de atores filhos e de dois laços acionados por eventos (figura 25): a interface gráfica de usuário e o controle de fluxo de execução.

A escolha de uma estrutura orientada a eventos para o controle de fluxo de execução foi devido à sua sinergia com um sistema de atores e as trocas de mensagens entre eles. Dessa maneira, sempre que ele receber resposta de um ator filho, ele pode disparar outras mensagens de execução.

O `Controller` possui o método `set_control_flag.vi` para que seus atores filhos relatem sucesso/falha de execução de subrotinas e baterias de testes. Para ser acessível às entidades externas ao ator, o método é exposto por uma classe de mensagem. Estas mensagens são usadas pelo bloco de controle de fluxo para alternar seu estado de execução e disparar outras mensagens para seus atores filhos.

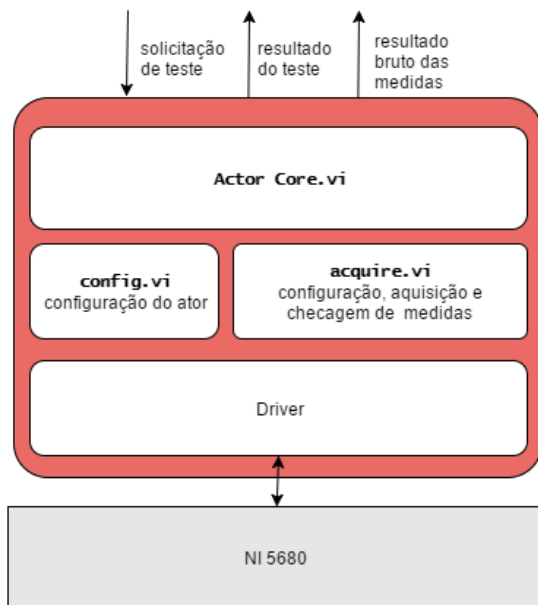


Figura 24 – Diagrama de blocos do PowerMeter, ator de interface com o NI 5680

Outro método exposto por uma classe de mensagem é `report status.vi`, que é usada para transmitir ao Controller os registros de execução de teste e saída da porta serial. Todos os dados recebidos por estas mensagens vão para a interface de usuário, para depuração em tempo de execução e posteriormente, enviadas ao ator Logger.

Seu tratador de exceção possui finalidades: parada, reinicialização e *watchdog* de atores filhos e, como próprio sugere, tratamento de falhas de execução. Parte dele foi implementado a partir das funções de tratamento

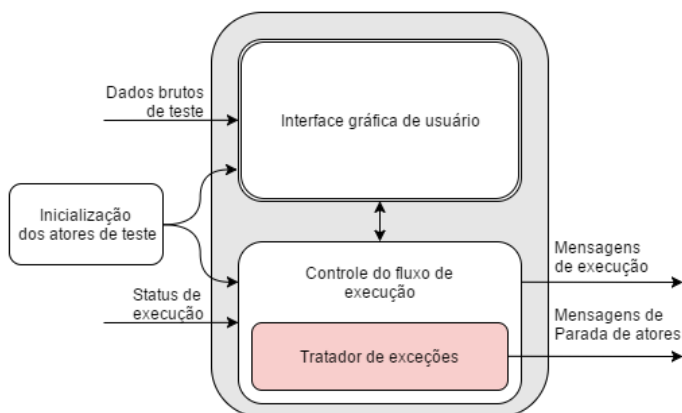


Figura 25 – Diagrama de blocos do Controller, ator supervisor

de erro do LabVIEW, do *Actor Framework* e outras funções customizadas, especialmente para casos de desconfiguração de instrumentos de medida.

A figura 26 expõe a interface de usuário do programa, que oferece ao usuário duas saídas de texto: despejo da comunicação serial, à esquerda, e o registro de execução das baterias de teste e medições, à direita.

5.2.6 Ergonomia

A ergonomia no trabalho envolvido foi considerada durante todo o projeto e desenvolvimento do programa. As etapas de teste que envolvem a interação do trabalhador foram pensadas de forma a minimizar o esforço repetitivo e o desgaste do operador. No que confere à parte de software, o interesse é minimizar o uso do teclado e mouse, bem como a necessidade de dividir atenção visual entre o DUT e o monitor. As partes principais

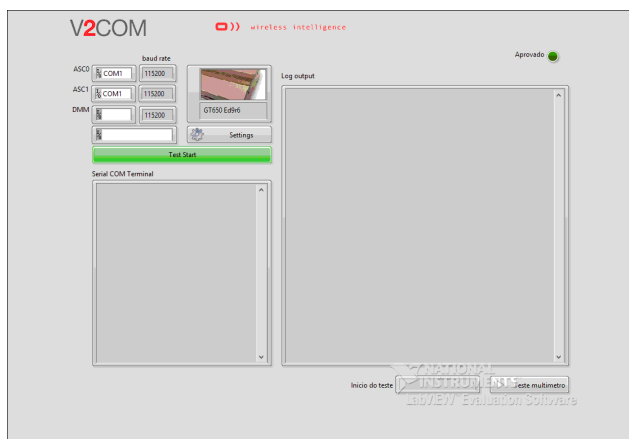


Figura 26 – Tela do painel principal de interface de usuário

onde o operador precisa interagir com o software são: teste dos LEDs do painel, medição de tensões com o multímetro e testes de alimentação e bateria.

Para o teste do painel de LEDs do DUT foi criada uma GUI esquemática (figura 27) para facilitar a comparação com o painel real, agilizando e diminuindo erros no processo de verificação visual.

Quanto às medições de tensões na placa, as mãos e atenção do operador se voltam para a DUT. Descuidos aqui podem causar erros ou, até mesmo, danos ao circuito. Em relação a isto, foi criado um *debouncer* para validação das medidas, permitindo que o operador passe a ponteira pelos pontos de medição sem a necessidade de digitar ou interagir com o teclado. Além disso, referências sonoras foram criadas para os casos de aprovação, repetição ou reprovação de medição, de forma que a atenção visual do operador não precise desviar do DUT para o computador.

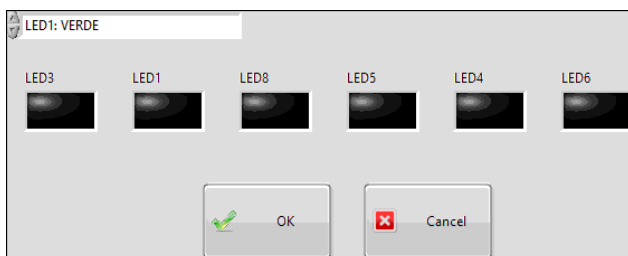


Figura 27 – Captura de tela do Painel de teste de led

Dentre as outras melhorias citam-se: a separação de uma tela dedicada somente ao registro da comunicação com o módulo para a melhor depuração; a redução de etapas de digitação por teclas simples ou mouse.

6 RESULTADOS E ANÁLISE

6.1 RESULTADOS E ANÁLISE DE DESEMPENHO

Com o programa pronto para ser validado, foi montado um esquema de teste para avaliação interna de desempenho do programa de testes, antes de entrar em produção na fábrica. A intenção era de compará-lo em relação à versão em Java e também detectar erros de software que possam ter passado despercebidos durante o desenvolvimento. Foram separadas cerca de 5 placas do GT650 para teste. Cada uma delas foi testada duas vezes por cada um dos dois programas, totalizando 20 execuções. O teste foi executado pelo autor em bancada de desenvolvimento.

A tabela 3 exibe as médias do tempo de execução de teste para os dois programas. Nota-se, pela tabela, uma redução de quase 7 segundos no tempo médio de teste do programa reescrito em relação ao programa legado. É possível observar que estes resultados são atribuídos somente pelas mudanças ergonômicas do programa, já que o roteiro é idêntico ao anterior. Questões de velocidade de processamento entre as linguagens não refletiram em muitas mudanças, se for considerado que nenhuma das atividades exige uso intenso de CPU, já que consiste de rotinas de comunicação e tratamento de dados em baixa velocidade. O gargalo do teste é a própria velocidade de resposta do dispositivo sob teste e das interações com o operador, como nos testes com o multímetro. Certamente que resultados melhores podem ser obtidos se aproveitados os recursos de concorrência de software que *framework* oferece, mas infelizmente não foi possível tratar os despesos de teste e comparar as diferenças de execução, bloco a bloco, do roteiro de teste.

Nota-se também que a distribuição dos tempos de teste do pro-

Tabela 3 – Média e desvio padrão das 20 amostras de teste de bancada

	Programa Legado	Programa Reescrito
Média (s)	33.65 ± 2.43	26.24 ± 3.37

grama novo é ligeiramente mais difusa do que o do programa antigo, o que pode ser atribuído à falta de hábito na operação do software.

6.2 ANÁLISE DA IMPLEMENTAÇÃO

A falta de um analisador de linguagem de marcação que interprete os roteiros de teste dificulta que este programa entre em operação em fábrica pois, no esquema atual, qualquer alteração no roteiro de teste requer uma nova compilação do programa.

Além da questão de flexibilidade do roteiro de teste, ainda é necessário que o programa passe por uma etapa maior de testes controlados e exposição a erros de produção controlados, a fim de depurar erros de programa como também melhorar os tratadores de exceção.

O ambiente LabVIEW permite ter uma visão clara do fluxo de sinais e seu *multithreading* natural permite a realização de sistemas concorrentes com facilidade. Entretanto, a implementação do modelo de atores ficou comprometida pelas limitações do seu *framework* em LabVIEW, principalmente pela impossibilidade de livre comunicação entre atores sem relação pai-filho, e também porque a criação de classes de mensagens depende de macros para serem geradas, o que enrijeceu o processo de desenvolvimento. O uso da linguagem gráfica do ambiente LabVIEW, apesar de simples para pequenas rotinas, torna-se complicado e difícil de refatorar, se aplicado a programas maiores, e sua implementação do paradigma orientado a objetos, assim como do modelo de atores, é demasiada-

mente complicada e problemática de usar. Uma maneira de contornar isto seria talvez a utilização do Labwindows/CVI, o ambiente C da National Instruments, ou o uso do LabVIEW somente em partes aonde ele se sobressalta como melhor opção. Isso permitiria melhor controle de versão, refatoração de código, revisão por pares e aplicação de boas práticas de programação textual.

7 CONCLUSÃO

Neste trabalho foi possível mostrar a aplicação do *framework* de atores para a solução de problemas de final de linha de montagem de placas eletrônicas. O trabalho conseguiu cumprir com os requisitos de modularidade e flexibilidade de implementação, porém foi limitado pela falta de um interpretador de arquivos de configuração que estendesse a flexibilidade do sistema à equipe de teste no local de produção. No que diz respeito à produtividade, uma atualização de jiga de teste pode melhorar a taxa de produção, e no caso particular das jigas com chaveamento de instrumentos de teste, o sistema de atores e o uso de concorrência pode ser melhor explorado.

7.1 TRABALHOS FUTUROS

Dentre as melhorias interessantes para o processos de teste de placas eletrônicas, destacam-se:

- Um interpretador de roteiro para flexibilização da execução de testes e para poder testar outros produtos com as mesmas ferramentas de software;
- Um escalonador e gerenciador de recursos para o uso com uma jiga multiplexada para teste de múltiplas unidades simultâneas;
- Criar uma linguagem de marcação para os roteiros de teste que dê suporte ao sistema de atores e otimize o tempo de execução;
- Construção de uma jiga de testes que suporte múltiplas placas de circuito impresso testadas simultaneamente. No aspecto de software,

isso demandaria a reescrita do controlador ou o desenvolvimento de um ator de hierarquia superior.

É possível melhorar consideravelmente a qualidade do teste ao aplicar técnicas de *projeto orientado à testabilidade* em uma nova revisão do produto. Essa opção, apesar de custosa à equipe de projeto, possui um bom retorno de investimento. Outra possibilidade de melhoria seria a partir de uma análise de cobertura de faltas, transformando isto em uma nova especificação de baterias de teste.

Ambas as propostas só se tornam viáveis conforme a relevância do volume de perdas durante a produção e dos retornos para manutenção, especialmente aqueles que se enquadram nos casos de *causa desconhecida* no controle de qualidade.

7.2 PALAVRAS FINAIS

Feitas as críticas, ressalta-se que o modelo de atores possui potencial para a implementação de rotinas de teste concorrente e escalona bem para aplicações maiores. O arranjo de módulos desacoplados foram implementados com sucesso e sua interação em teste desempenhou conforme esperado. Os avanços na ergonomia da interface mostraram-se também relevantes à produtividade e nas melhorias de condição de trabalho.

O trabalho foi proveitoso no aprendizado de técnicas de teste e diagnóstico de sistemas eletrônicos e suas aplicações durante os processos de manufatura, operação e manutenção. Constatou-se também que o uso de técnicas e boas práticas de desenvolvimento de software facilitam a manutenção e melhorias ao longo dos anos, prevenindo retrabalho e otimizando o tempo de desenvolvimento.

REFERÊNCIAS

- Beningo, J. (2013). Bootloader design for microcontrollers in embedded systems.
- Blume, P. A. (2007). *The LabVIEW style book*. Pearson Education.
- Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.
- Cinterion (2016). Ehs6 wireless module global 3g with java embedded. Disponível em: <<http://www.gemalto.com/m2m/modules-terminals>>. Acesso em 10 de junho de 2016.
- Cook, A., Ull, D., Elm, M., Wunderlich, H.-J., Randoll, H., and Dohren, S. (2012). Reuse of structural volume test methods for in-system testing of automotive asics. In *2012 IEEE 21st Asian Test Symposium*, pages 214–219. IEEE.
- Crouch, A. (2011). Fpga-controlled test (fct): What it is and why is it needed? Whitepaper, ASSET InterTech, Inc.
- de Carvalho Felgueiras, M. C. M. (2008). *Apoio à depuração e teste de circuitos mistos compatíveis com a norma IEEE1149. 4*. PhD thesis, Universidade do Porto.
- de Melo, A. R. (2015). Sistema de inspeção visual de placas de circuito impresso para linhas de produção em pequenas séries em um contexto multiagentes. Master's thesis, Universidade Federal de Santa Catarina.
- Erb, B. (2012). Concurrent programming for scalable web architectures. Diploma thesis, Institute of Distributed Systems, Ulm University.

- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*. Pearson Education.
- Hewitt, C. (2010). Actor model for discretionary, adaptive concurrency. *CoRR*, abs/1008.1459.
- Hewitt, C., Bishop, P., and Steiger, R. (1973). Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence. In *Advance Papers of the Conference*, volume 3, page 235. Stanford Research Institute.
- Hewitt, C. and Zenil, H. (2013). What is computation? actor model versus turing’s model. *A Computable Universe: Understanding and Exploring Nature as Computation*, pages 159–85.
- Hird, K., Parker, K. P., and Follis, B. (2002). Test coverage: what does it mean when a board test passes? In *Test Conference, 2002. Proceedings. International*, pages 1066–1074. IEEE.
- Huang, S.-H. and Pan, Y.-C. (2015). Automated visual inspection in the semiconductor industry: A survey. *Computers in industry*, 66:1–10.
- Huang, X., Zhu, S., Huang, X., Su, B., Ou, C., and Zhou, W. (2015). Detection of plated through hole defects in printed circuit board with x-ray. In *2015 16th International Conference on Electronic Packaging Technology (ICEPT)*, pages 1296–1301.
- IEEE (1990). Ieee standard test access port and boundary - scan architecture. *IEEE Std 1149.1-1990*.

- IEEE (1995). Supplement to ieee std 1149.1-1990, ieee standard test access port and boundary-scan architecture. *IEEE Std 1149.1b-1994*.
- IEEE (2003a). Ieee standard for boundary-scan testing of advanced digital networks. *IEEE Std 1149.6-2003*, pages 0–132.
- IEEE (2003b). Ieee standard for in-system configuration of programmable devices. *IEEE Std 1532-2002 (Revision of IEEE Std 1532-2001)*, pages 0–141.
- IEEE (2005). Ieee standard testability method for embedded core-based integrated circuits. *IEEE Std 1500-2005*, pages 0–117.
- IEEE (2010). Ieee standard for reduced-pin and enhanced-functionality test access port and boundary-scan architecture. *IEEE Std 1149.7-2009*, pages 1–985.
- IEEE (2011). Ieee standard for a mixed-signal test bus. *IEEE Std 1149.4-2010 (Revision of IEEE Std 1149.4-1999)*, pages 1–116.
- IEEE (2012). Ieee standard for boundary-scan-based stimulus of interconnections to passive and/or active components. *IEEE Std 1149.8.1-2012*, pages 1–95.
- IEEE (2013). Ieee standard for test access port and boundary-scan architecture. *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pages 1–444.
- IEEE (2014). Ieee standard for access and control of instrumentation embedded within a semiconductor device. *IEEE Std 1687-2014*, pages 1–283.

- Jutman, A., Reorda, M. S., and Wunderlich, H.-J. (2014). High quality system level test and diagnosis. In *2014 IEEE 23rd Asian Test Symposium*, pages 298–305. IEEE.
- Keysight (2003). Agilent 5dx automated x-ray inspection test system. Disponível em: <http://www.keysight.com/upload/cmc_upload/A11/59890268EN.pdf>. Acesso em 10 de fevereiro de 2017.
- Leinbach, G. and Oresjo, S. (2001). The why, where, what, how, and when of automated x-ray inspection. *Agilent Technologies*.
- Ley, A. W. and InterTech, A. (2009). Defect coverage for non-intrusive board tests. *ASSET InterTech*.
- Liu, A., Zou, C., Lin, T., Li, J., Tan, C. K., Feng, Z. J., Geiger, D., Liu, S., Wen, J. P., Xiao, J., Liu, L., and Krastev, E. (2016). X-ray inspection methods for controlling pcba potting process #8212; 2dx and partial angle computer tomography. In *2016 Pan Pacific Microelectronics Symposium (Pan Pacific)*, pages 1–5.
- Lotz, C., Collins, P., and Wiatrowski, D. (2006). Functional board test-coverage analysis what does it mean when a functional test passes? In *European Board Test Workshop*.
- Luo, B. and Zhang, L. Y. (2010). Smt solder paste deposit inspection based on 3d pmp and 2d image features fusion. In *2010 International Conference on Wavelet Analysis and Pattern Recognition*, pages 190–194.
- McClure, D. (2000). X-rays spot circuit-board flaws. *Machine Design*. Disponível em: <<http://machinedesign.com/archive/>>

- x-rays-spot-circuit-board-flaws>. Acesso em 10 de janeiro de 2017.
- Mitra, S., Seshia, S. A., and Nicolici, N. (2010). Post-silicon validation opportunities, challenges and recent advances. pages 12–17.
- Oresjo, S. (2002). When to use aoi, when to use axi, and when to use both. In *Proceedings of Nepcon West*.
- Owen, M. and Hawthorne, J. (2000). Process control for solder paste deposition. *SMT Surface Mount Technology Magazine*, 14(1):3.
- P1149.10, I. (2016). High speed test access port and on-chip distribution architecture working group.
- P1838, I. (2016). Ieee 3d-test working group (3dt-wg).
- Pang, G. K. H. and Chu, M. H. (2009). Automated optical inspection of solder paste based on 2.5d visual images. In *2009 International Conference on Mechatronics and Automation*, pages 982–987.
- Poole, I. (2016). In circuit test tutorial. Disponível em: <http://www.radio-electronics.com/info/t_and_m/ate/ict-in-circuit-test-tutorial.php>. Acesso em 10 de dezembro de 2016.
- Riefert, A., Ciganda, L., Sauer, M., Bernardi, P., Reorda, M. S., and Becker, B. (2014). An effective approach to automatic functional processor test generation for small-delay faults. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE.
- Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified modeling language reference manual, the*. Pearson Higher Education.

- Smith, A. C. and Mercer, S. R. (2011). Using the actor framework in labview. Whitepaper, National Instruments.
- Stollon, N. (2011). *IEEE P1687 – IJTAG*, pages 137–144. Springer US, Boston, MA.
- Suzuki, D., Noguchi, K., Murakoshi, T., and Teramoto, A. (2013). Development of high-speed x-ray ct inspection system using x-ray line sensor. In *2013 3rd IEEE CPMT Symposium Japan*, pages 1–4.
- Thibeault, C., Tremblay, D., and Hariri, Y. (2006). Redefining the role of functional testing. In *2006 IEEE North-East Workshop on Circuits and Systems*, pages 133–136.
- Tumin, K., Vargas, C., Patterson, R., and Nappi, C. (2001). Scan vs. functional testing - a comparative effectiveness study on motorola's mmc2107tm. In *Proceedings International Test Conference 2001 (Cat. No.01CH37260)*, pages 443–450.
- Wenzel, T. (2013). Test & measurement technology goes embedded. Whitepaper, GÖPEL electronic GmbH.
- Wenzel, T. and Zimmermann, E. (2016). Boundary scan meets functional test. *SMT Magazine*, 31(12):74–84.
- Xu, G., Khor, E., Su, L., Liu, W., Feng, Z. J., Geiger, D., Ngor, L. L., Yun, W., Hwa, C. H., and ZiYang, S. (2016). X-ray inspection study with pcb cavity press-fit connectors. In *2016 Pan Pacific Microelectronics Symposium (Pan Pacific)*, pages 1–4.