George Redivo Pinto

Equipamento para teste de transceptores ethernet

Porto Alegre, Rio Grande do Sul, Brasil

Novembro de 2013

George Redivo Pinto

Equipamento para teste de transceptores ethernet

Equipamento para automatização de testes de equipamentos de rede dotados de transceptores SFP.

Orientador:
Prof. Marcos Augusto Stemmer

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

Porto Alegre, Rio Grande do Sul, Brasil

Novembro de 2013

Resumo

Durante o desenvolvimento de um sistema, várias etapas de teste são executadas a fim de garantir o correto funcionamento do mesmo. Em equipamentos destinados a aplicações críticas, esses testes são massivos, na tentativa de garantir que o equipamento, em hipótese alguma, fuja do comportamento especificado e esperado, quando em campo.

Equipamentos *ethernet switch*, além de interagir com outros equipamentos de rede, como *hosts*, roteadores e outros *switches*, podem possuir portas do tipo *Small Form-factor Pluggable transceiver* (SFP) e interagirem diretamente com eles. Idealmente, o processo de teste de equipamentos com portas SFP deve incluir testes de inserção e remoção, o que, por sua vez, necessita de intervenção física. Essa intervenção é geralmente manual, o que acaba provocando um alto índice de erros.

Nesse contexto, este Trabalho de Conclusão de Curso propõe um equipamento capaz de viabilizar a automatização dos testes de SFP. Essa plataforma inclui um módulo de injeção de falhas. Em mais detalhes, a plataforma será capaz de fazer a inserção e remoção de dois SFPs diferentes em uma mesma porta do *ethernet switch*, além da injeção de falhas no barramento I^2C, tudo controlado por comunicação serial, possibilitando assim a automatização do processo como um todo.

A solução proposta foi validada em um equipamento de rede fornecido pela empresa Datacom Telemática. Neste sentido, foi realizado uma série de experimentos visando a validação da solução proposta.

Abstract

During a product design, several steps are performed to guarantee the correct product operation. In equipment designed for critical applications, these tests are massively stressed, trying to guarantee that the equipment works according to specifications.

Ethernet switches, also may interact with other network equipment, like hosts, routers and another switches, could have Small Form-factor Pluggable transceiver (SFP) ports and may interact directly with these equipment. Ideally the test process on SFP ports must include SFP insertion and removing tests, which requires a physical intervention. This intervention, in general, is done manually, inserting a high human error probability.

This paper proposes an infrastructure to turn automatic SFP tests feasible, including a fault injection module. Detailing it, this equipment is capable to perform the insertion and removing of two different SFPs in a same ethernet switch's port. It is also capable to do fault injection on I^2C bus, all of it controlled by a serial communication, turning possible the automation of this test process.

This solution was validated in a network equipment provided by Datacom Telemática. To validate it, several tests was performed to guarantee the functionality of the features proposed in this paper.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Siglas

TCC Trabalho de Conclusão de Curso

SFP *Small Form-factor Pluggable*

XFP 10 *Gigabit Small Form-factor Pluggable*

ATE *Automatic Test Equipment*, equipamento de teste automático

MDIX *Medium Dependent Interface crossover*, interface cruzada dependente de mídia

MSA *Multi-Source Agreement*, acordo multi-fonte

OSI *Open Systems Interconnection*, sistema aberto de interconexões

CI Circuito Integrado

DD *Digital Diagnostics*, diagnosticos digitais

PCB *Printed Circuit Board*, placa de circuito impresso

HS-PCB *High Speed* PCB, PCB de alta velocidade

LS-PCB *Low Speed* PCB, PCB de baixa velocidade

BIST *Built-In Self Tests*

ACK *acknowledgement*

SoC *System on Chip*

ARM-DEV-KIT *ARM Devetopment Kit*, Kit de desenvolvimento ARM

SerDes Serializador/Desserializador

SGMII *Serial Gigabit Media Independent Interface*, Interface serial gigabit independente de mídia

1 Introdução

Com o aumento da complexidade dos sistemas atuais, a tarefa de teste é algo de extrema importância. Um teste, em linhas gerais, é uma atividade na qual uma ou mais entradas são injetadas em um dado sistema e suas saídas são analisadas, a fim de encontrar discrepâncias entre o comportamento especificado e o comportamento real do sistema.

Dependendo das características e dimensões de um sistema, a tarefa de desenvolvimento pode se tornar bastante complexa (DELAMARO; MALDONADO; JINO, 2007), o que torna a atividade de teste de um projeto algo crucial para garantir o bom funcionamento de um produto.

O teste pode ser feito em vários níveis e estágios de produção de um sistema. Desde testes dos circuitos integrados (CIs) que compõe o sistema até testes em campo, utilizando *Built-In Self Tests*, BISTs. É desejável que um problema seja descoberto no estágio mais básico possível, pois o custo da resolução de um problema sobe conforme o nível em qual ele foi encontrado (WANG; WU; WEN, 2006).

O teste de um projeto pode ser dividido em várias etapas. Entre elas estão: (1) teste funcional, onde o projeto é testado a fim de garantir a conformidade com sua especificação comportamental e (2) teste comportamental, onde o alvo do teste é excitado de forma a simular o uso real do equipamento ou sistema, tentando encontrar falhas, para que elas sejam corrigidas antes do projeto ser dado como pronto. Os estímulos do teste podem ser introduzidos manualmente, com uma pessoa interagindo ativamente com o alvo do teste. Outra forma, é utilizando componentes automatizadores de teste, onde um elemento pré programado interage com o alvo do teste, inserindo os estímulos (HSUEH; TSAI; IYER, 1997). Esses componentes são chamados equipamento de teste automático, ATE, do inglês *Automatic Test Equipment*.

Existem também testes envolvendo injeção de falhas, no qual o equipamento ou sistema é

testado em condições anormais. Os testes com injeção de falhas são utilizados para observar o comportamento do alvo do teste em uma situação de falha tanto do próprio projeto, quanto de outros subsistemas envolvidos. Com os dados obtidos nos testes com injeção de falhas pode-se medir, por exemplo, a robustez do projeto e se ele responde da forma esperada quando encontra um estado de falha. As falhas podem ser introduzidas em diversos níveis, desde níveis mais baixos, como curto circuitos, até o nível de usuário, como inserção de um valor inválido em um campo de entrada. Esse tipo de teste é muito útil em aplicações críticas, pois essas tem de se manter sempre ativas, muitas vezes implementando módulos de tolerância a falhas para evitar a indisponibilidade do sistema. Aplicações críticas são, por exemplo, sistemas de navegação de aeronaves, ou sistemas automatizados de controle de metrô, os quais nunca podem parar, pois causariam um prejuízo muito grande.

Os testes podem ser manuais ou automáticos, dependendo dos requerimentos do teste. Testes manuais tem diversas desvantagens quando comparados à testes automáticos, como menor velocidade de execução do teste, tempo gasto com a aprendizagem de um novo procedimento, falta de consistência e possibilidade de erro humano na execução das tarefas. Embora muitas das desvantagens sejam relacionadas apenas a requisitos temporais, a possibilidade de erro durante a execução do roteiro de teste está relacionada à credibilidade do teste. Se, durante a execução de um teste manual, algum passo do roteiro for pulado, adicionado, ou executado de forma errada, o teste pode apresentar resultados inválidos, tanto falsos positivos quanto falsos negativos. Por todos esses motivos, o esforço gasto em automatização de testes vem sendo cada vez maior, assim, os recursos humanos podem ser investidos em outras atividades mais produtivas, como o aprimoramento dos casos de teste, ao invés de serem gastos na execução dos mesmos.

A atividade de teste, além se ser a base um sistema livre de falhas, ajuda a desenvolver estatísticas de rendimento do processo (WANG; WU; WEN, 2006). Assim, dependendo do nível de detalhamento é possível identificar falhas de processo do projeto como um todo, desde a fase de especificação até a fase de produção.

Com o objetivo de flexibilizar o uso, alguns equipamentos de rede possuem portas de comunicação com conector para a inserção de um transceptor (*transceiver*) externo. Um *transceiver*, do ponto de vista de redes de dados, é um dispositivo da camada física, capaz de transmitir e receber dados, fazendo a conversão dos sinais (sinais elétricos em ópticos, por exemplo) e/ou a conversão da interface de comunicação. Com diferentes *transceivers*, é possível, por exemplo, uma mesma

porta de comunicação ser utilizada com fibra óptica ou conetor RJ45.

A configuração aplicada na porta deve ser compatível com o transceptor inserido na mesma, para evitar comportamentos inesperados. No entanto, devido à possibilidade de variadas configurações e modelos de transceptores, a configuração se torna algo extremamente delicado e suscetível a erros.

Certos equipamentos são capazes de configurar automaticamente a porta, baseando-se nas informações obtidas do transceptor. A dificuldade da autoconfiguração é agravada pelo fato de existir uma grande gama de combinações entre configuração e modelo de transceptor. Além disso, existem casos onde equipamentos são logicamente empilhados (*stacking*[1]).

Em um ambiente de *staking*, podem existir diversos equipamentos com características diferentes, o que dificulta a configuração destes.

Durante o processo de teste dos *trasnceivers*, se faz necessário a conexão e/ou desconexão física dos mesmos, geralmente manual, trazendo os riscos e desvantagens já citadas anteriormente. A automatização desses procedimentos resulta em um grande acréscimo na produtividade e confiabilidade do teste, porém isso introduz um elemento a mais no ambiente de testes, um ATE.

Este TCC propõe um solução baseada em *hardware* e *software* que permite a automatização de testes de equipamentos *ethernet* dotados de transceptores SPFs. Como estudo de caso foram utilizados transceptores SPFs em *switches ethernet*.

[1]Do ponto de vista de equipamentos de rede, quando monta-se um ambiente onde vários equipamentos são ligados uns aos outros, de forma a se tornarem virtualmente um equipamento só, com capacidade somada e controle centralizado, denomina-se um ambiente *stacking*.

2 *Fundamentação Teórica*

Dentro do universo de redes de dados existem inúmeras distinções entre tipos de equipamentos, funcionalidades e configurações. O cenário escolhido para o estudo é o de equipamentos *ethernet* dotados de portas SFPs.

2.1 I^2C

Desenvolvido pela Philips (atual NXP), nos anos 80, a comunicação I^2C é uma comunicação em barramento, que utiliza dois fios para tráfego de dados serial. Usando uma estrutura mestre-escravo, é possível a comunicação entre vários dispositivos utilizando apenas dois fios, em dreno aberto: um fio de *clock* e um de dados, chegando à taxa de transferência de 3,4Mbps (I2C-BUS.ORG, 2013). Devido à simplicidade e versatilidade de implementação, o I^2C é largamente usado em comunicações simples que não exigem alta taxa de transferência de dados.

A comunicação I^2C se dá através de um barramento onde ficam conectados um dispositivo mestre e até 127 dispositivos escravos (figura 2.1). O dispositivo mestre não possui endereço. Os dispositivos escravos possuem um endereçamento de 7 bits, podendo endereçar 127 dispositivos. O endereço 0 é utilizado como endereço geral, uma especie de endereço de *broadcast*. Existe também um modo de endereçamento de 10 bits, o que aumenta a quantidade de endereços disponíveis.

Em linhas gerais, a comunicação se dá da seguinte forma: (1) mestre informa ao barramento que deseja se comunicar com um dispositivo informando seu endereço; (2) o dispositivo escravo responde um *acknowledgement* (ACK); (3) mestre informa registrador interno com o qual deseja se conectar; (4) escravo responde dado referente ao registrador interno, em caso de leitura, ou

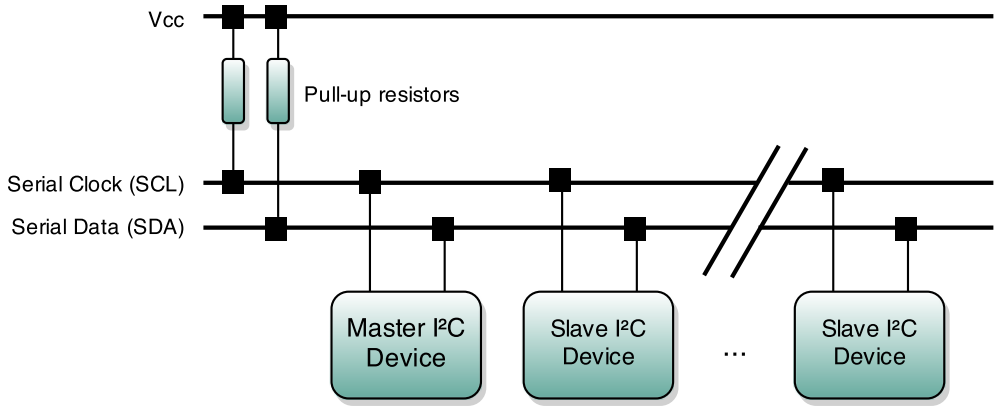

Figura 2.1: Barramento I^2C

faz a atribuição em caso de escrita.

2.2 Portas Ethernet

A *ethernet* é uma arquitetura de rede de dados largamente utilizada. Ela é normatizada pela IEEE 802.3, que descreve todos os detalhes da arquitetura. As portas de um equipamento *ethernet* podem ser de tipos variados e ter configurações distintas. Esta sessão destina-se a detalhar as principais características das portas *ethernet*.

2.2.1 Modelo OSI

O Modelo OSI (*Open Systems Interconnection*, sistema aberto de interconexões) é uma das mais antigas e mais usadas arquiteturas de comunicação entre máquinas. Este modelo divide a comunicação em 7 camadas hierárquicas. Cada camada tem uma função bem definida dentro da comunicação e consegue se comunicar apenas com as camadas adjacentes. Essa propriedade modulariza a comunicação de forma a diminuir a complexidade da comunicação como um todo.

As camadas são divididas da seguinte maneira:

Aplicação

A camada de aplicação é a camada de mais alto nível. É nela que ficam os *softwares* como navegadores de internet e clientes de e-mail.

Apresentação

Esta camada é responsável pela formatação e conversão dos dados.

Sessão

A camada de sessão gerencia o fluxo de dados, sendo responsável por tratamento de erros dessa natureza.

Transporte

É responsável pela entrega e recebimento dos dados, tentando garantir a confiabilidade de transmissão.

Rede

Esta camada faz o roteamento dos pacotes, escolhendo a melhor rota, baseando-se em métricas como congestionamento e custo do caminho.

Enlace

A camada de enlace é o elo entre a camada física e a camada de rede. Ela é responsável pela configuração de fluxo de dados e pela topologia de rede.

Física

Esta é a camada de mais baixo nível, onde os sinais lógicos e elétricos são convertidos e enviados.

Este trabalho atua nas duas camadas de nível mais baixo: camada física e de enlace.

2.2.2 Portas físicas

As portas de um equipamento *ethernet* servem para fazer a ligação entre dois dispositivos em uma rede *ethernet*. Diversos conectores já foram usados em aplicações *ethernet* e o mais comum é o conector RJ45, o qual é usado para fazer a ligação com um par trançado[1]. Com o desenvolvimento da tecnologia óptica, os cabos de fibra óptica vem se popularizando, principalmente em aplicações de alta frequência e aplicações de longa distância. A fibra óptica, por usar a luz,

[1]Par trançado é um tipo de cabeamento onde dois fios, contendo sinais diferenciais, são enrolados um ao redor do outro, a fim de diminuir a suscetibilidade à interferências eletromagnéticas. Cabos *ethernet* do padrão CAT 6, por exemplo, contem 4 pares trançados.

pode trabalhar em frequências muito altas, tem baixo nível de perda de sinal e suscetibilidade a interferência. Por outro lado, o cabo elétrico, apesar de não possuir tantas vantagens, chega a 1Gbps e distâncias de até 100 metros por um custo bem inferior à fibra óptica.

Como foi citado no capítulo 1, existem equipamentos com portas as quais seu conector não é nem para fibra óptica, nem para RJ45, mas sim conector para algum *transceiver*. Desta forma, uma mesma porta pode ser usada com cabeamento de fibra óptica ou par trançado, dependendo do transceptor conectado. Em aplicações *ethernet*, os transceptores mais comumente usados são os SFPs e XFPs. SFPs são geralmente usados para aplicações de rede *Gigabit Ethernet*, enquanto XFPs são usando para aplicações de 10 *Gigabit Ethernet*.

2.2.3 Configuração

Cada porta *ethernet* deve receber uma série de configurações para que tenha uma operação estável. Essas configurações ditam desde o *bit rate*, até configurações de inversão de TX e RX. Os parâmetros mais comuns de configuração de uma porta *ethernet* estão listados na tabela 2.1.

Tabela 2.1: Parâmetros de configuração de portas *ethernet*

Parâmetro	**Descrição**
Velocidade	Informa a velocidade do tráfego na porta.[1]
Speed Capabilities	Informa as configurações de velocidade possíveis para a porta.[2]
Duplex	Informa se a conexão é *Full Duplex* ou *Half Duplex*.[1]
Duplex Capabilities	Informa as configurações de *duplex* possíveis para a porta.[2]
Negociação	Informa se os parâmetros de *duplex* e velocidade serão forçados ou negociados, utilizando as *capabilities*.
MDIX	Informa se a ligação de TX e RX está normal, invertida ou se será configurada automaticamente.[3]
Habilitação	Informa se a porta está habilitada ou desabilitada.

[1]Configuração válida apenas quando auto negociação está desabilitada.
[2]Configuração válida apenas quando auto negociação está habilitada.
[3]MDIX automático funciona apenas quando auto negociação está habilitada, caso contrário o MDIX permanece com o *status* anterior.

Existe ainda uma configuração referente ao protocolo de comunicação usado entre a camada física e a camada de enlace. Essa configuração geralmente é transparente para o usuário da porta, ao contrário das configurações listadas na tabela 2.1. A escolha de qual protocolo utilizar

depende do protocolo utilizado pelo dispositivo parceiro ligado à porta. Dois protocolos normalmente são utilizados: (1) SerDes (Serializador/Desserializador) e (2) SGMII (*Serial Gigabit Media Independent Interface*)[2]. Os dois são protocolos de comunicação serial. O protocolo SerDes é geralmente utilizado em conexões diretas, onde não existe um PHY intermediário (ver sub sessão 2.3.2), caso contrário utiliza-se o protocolo SGMII.

As configurações a serem feitas dependem de diversos fatores como aplicação, tipo de cabo, tipo de conexão e suporte às configurações por parte das duas pontas da conexão, da porta e, em caso de uso de *transceivers*, suporte às configurações por parte dos *transceivers*.

Devido a essa variabilidade, a parte do sistema que executa as configurações é extremamente sensível. Em casos onde existe um transceptor na porta, essa sensibilidade é acentuada, pois além do suporte de configuração das portas dos dois lados da conexão, a configuração ainda precisa ser coerente com o transceptor. Durante o funcionamento do equipamento, é possível trocar o *transceiver* de uma determinada porta por outro com características diferentes, o que faz com que configurações diferentes devam ser escolhidas e aplicadas, dependendo das especificações do transceptor.

Exitem ainda equipamentos com portas chamadas portas combo. Essas portas são replicadas. Para flexibilizar as aplicações do equipamento, cada porta combo é dotada de um conector SFP e um conector RJ45, porém os dois conectores não podem ser usados simultaneamente. Essas portas são configuradas de maneira completamente diferente em cada meio (meio óptico, SFP, e meio elétrico, RJ45), o que torna a configuração de portas combo ainda mais sensível a erros.

Com essa dinâmica, a configuração de portas pode se tornar algo complexo, em especial em casos onde o equipamento é provido de um módulo de auto configuração. Essas características tornam o teste de configuração e troca de transceptores uma tarefa crítica, pois existem inúmeras combinações de configuração possíveis, tornando o teste uma tarefa longa e um bom alvo para ser automatizado.

[2]SGMII é um protocolo de comunicação criado pela Cisco com o objetivo de viabilizar a utilização entre a camada MAC e PHYs 10/100/1000 com o mínimo possível de perda de sinal (CISCO SYSTEMS, 2005).

2.3 SFPs

SFP (*Small Form-factor Pluggable*) é um tipo de transceptor, um modelo de tamanho físico pequeno, utilizado para conexões de baixa e média velocidade (abaixo de 10Gbps). Existem diversos tipos de SFPs. Os SFPs dividem-se basicamente em SFPs elétricos e SFPs ópticos, os quais mais comumente usam conectores RJ45 e LC, respectivamente.

Os SFPs são descritos por dois MSA (*Multi-Source Agreement*), INF-8074 e SFF-8472. Esses documentos definem como devem ser as características mecânicas, elétricas e lógicas, como dimensões, impedâncias e endereçamento das informações. Como esses documentos não são uma norma oficial, apenas um acordo entre alguns dos grandes fabricantes desses equipamentos, existe a possibilidade de alguns modelos não seguirem as recomendações. Isso torna a configuração das portas envolvendo SFPs mais difícil e perigosa. SFPs que não seguem as especificações dos MSAs podem apresentar comportamentos indefinidos. Por esse motivo alguns fabricantes de equipamentos que usam SFPs tem uma política de homologação de SFPs e bloqueio de SFPs não homologados.

Cada SFP possui pelo menos um dispositivo interno acessado por I^2C.

O dispositivo básico, acessado pelo endereço 0x50 (SFF COMMITTEE, 2013), comum à todos os SFPs, é um bloco de memória onde ficam gravadas as informações sobre o SFP. Esse bloco de memória guarda informações como nome do fabricante, número de série, tipo conector, taxa máxima de transmissão e outros dados referentes ao SFP, bem como códigos de checagem.

Além do dispositivo básico, o SFP pode conter outros dispositivos adicionais acessados por I^2C, tais como módulo de diagnóstico digital (DD, *Diagital Diagnostics*), e PHY interno do SFP. Cada dispositivo é acessado através de um endereço I^2C diferente.

2.3.1 Tipos de SFPs

Entre todos os modelos de SFPs existem dois grandes grupos: os SFPs elétricos e os SFPs ópticos. Dentro desses grupos existem outras subdivisões. As divisões são feitas a fim de separar os SFPs por taxa de transferência, possibilidade de negociação, *capabilities* e, especificamente para SFPs ópticos, existem divisões de comprimento de onda e distância máxima (potência) do

laser.

Os variados tipos de SFPs necessitam de configurações diferentes, alguns suportam mais configurações que outros. Isso faz com que, em caso de auto configuração, de alguma forma, o equipamento que está hospedando o SFP deva conseguir descobrir informações pertinentes às configurações e fazer a correta aplicação dessas configurações. Essas informações estão dispostas em um memória dentro do SFP, acessada por I^2C, cujo mapeamento é descrito pelos documentos INF-8074 e SFF-8472 (SFF COMMITTEE, 2001) (SFF COMMITTEE, 2013).

A maioria dos SFPs ópticos com taxa de transferência na casa de 1Gbps fazem uma passagem direta (*pass-through*) na comunicação com o *switch*, através do modo SerDes (figura 2.2), utilizando o PHY (melhor detalhado em 2.3.2) do próprio *switch* para fazer as configurações, apenas transformando o sinal óptico em sinal elétrico e vice-versa. Já os SFPs elétricos e alguns 100Mbps ópticos possuem um PHY interno, o qual deve ser configurado com os parâmetros vistos na tabela 2.1, além da configuração do próprio *switch*.

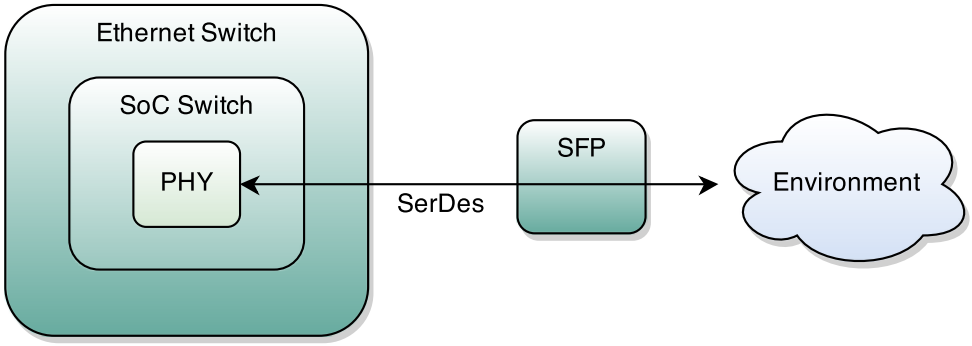

Figura 2.2: Comunicação *pass-through* entre *switch* e SFP.

2.3.2 PHY

PHY, do inglês *physical*, do ponto de vista de redes *ethernet*, é um dispositivo que trabalha na camada física do modelo OSI, capaz de fazer a conversão de sinais analógicos em digitais, e vice-versa, enviando e recebendo *frames ethernet* (PHY, 2013). Além das configurações citadas na tabela 2.1, alguns PHYs dispões de configurações e informações mais específicas, como tipo de mídia, protocolo de comunicação e inversão de trilhas.

Esses dispositivos podem ser um CI separado, ou estar integrado em um CI com mais funci-

onalidades. O tráfego de uma porta *ethernet* pode passar por mais de um PHY, mesmo antes de sair do equipamento origem. Isso acontece em casos de portas combo, por exemplo, onde além de o tráfego passar pelo PHY integrado no SoC (*System on Chip*) do *switch*, pode passar por mais um PHY, que é responsável de escolher qual meio (porta elétrica ou porta SFP) será enviado ou, em caso de recebimento, qual porta será escutada, como mostra a figua 2.3. Outro caso onde há mais de um PHY envolvido, é em ocasiões em que usamos SFPs com PHY embarcado (figura 2.4), como explicado na sessão 2.3.1, onde esse PHY é acessado através de uma comunicação I^2C com o SFP. Nesses casos existe um PHY para fazer a comunicação do *switch* até o SFP e outro PHY, embarcado no SFP, que faz a comunicação até o outro lado da conexão.

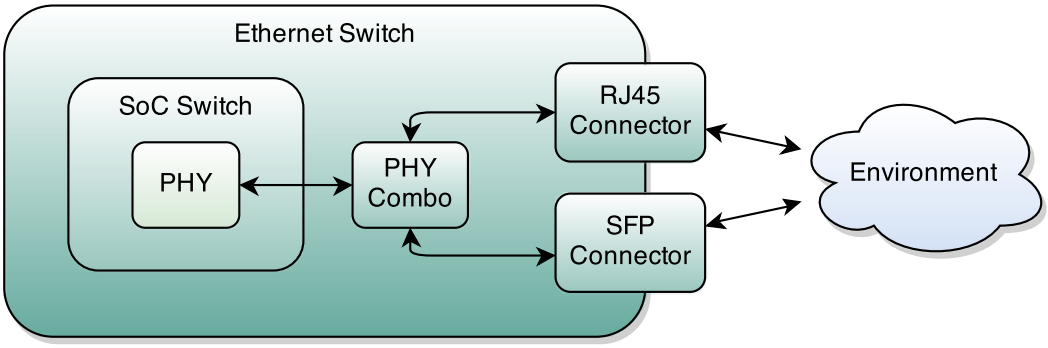

Figura 2.3: Comunicação através de um PHY combo.

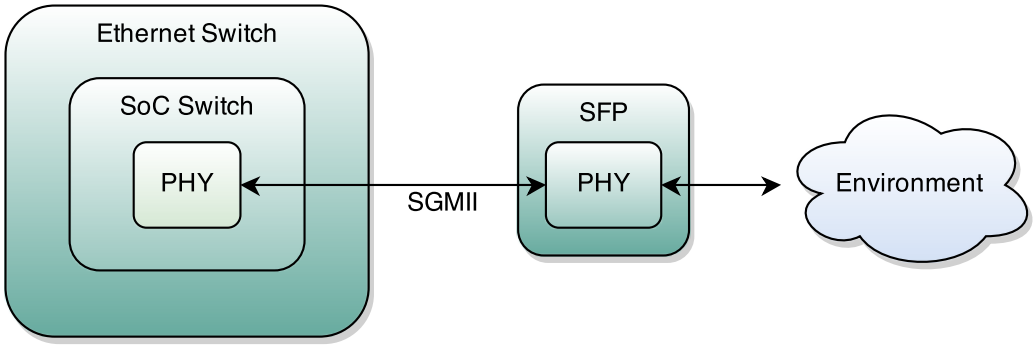

Figura 2.4: Comunicação através de um PHY interno ao SFP elétrico.

Em todos os casos onde existe mais de um PHY em um mesmo lado da comunicação é necessário fazer a correta configuração de todos os PHYs. Essas configurações podem variar de caso para caso, dependendo de características estáticas, como da arquitetura do equipamento, e características dinâmicas, como a topologia de rede e aplicação no qual o equipamento em

questão está sendo utilizado.

2.3.3 Dispositivos Internos

Os SFPs são providos basicamente de uma memória interna. Essa memória guarda informações sobre o SFP, como nome do fabricante, código do produto e número de série, além de dados pertinentes à auto configuração, como taxa máxima de dados e *Ethernet Standard*.

As informações disponibilizadas na memória do SFP são a base para decisões de autoconfiguração dos equipamentos hospedeiros de SFPs. É na memória que está toda a informação estática acessível. A disposição das informações da memória dos SFPs são descritas pelos MSAs citados no capítulo 2.3, mas como foi dito anteriormente, visto que os MSAs não são normas, os SFPs podem ou não seguir essas recomendações.

Ainda existem áreas de memória específicas para o vendedor do SFP. Essas áreas possibilitam o vendedor gravar informações como nome do vendedor, *part number* específico usado pelo vendedor e quaisquer outras informações, como código de homologação, ou informações adicionais para a identificação do módulo SFP.

Alguns SFPs são providos de um módulo DD, especificado pela SFF-8472, que é um sistema de monitoramento do status do SFP. O DD guarda informações como temperatura do SFP, potência do *laser* (TX e RX) e tensão de alimentação. Esses dados são lidos em tempo real e acessados através de um dispositivo I^2C específico. Lendo informações de DD é possível ter um relatório da saúde do SFP, o que pode ser usado para executar ações de prevenção ou simplesmente como um informativo ao usuário.

2.4 Modelos de Falha

Em um sistema eletrônico problemas podem ocorrer em várias áreas e em vários níveis de abstração. Quando se deseja trabalhar na descoberta, ou injeção de algum tipo de problema, é necessário entender em que nível está e como o problema ocorre.

Primeiramente deve-se entender a diferença entre três termos que podem facilmente ser confundidos: defeito, erro e falha. Michael L. Bushnell e Vishwani D. Agrawal (BUSHNELL;

AGRAWAL, 2000) fazem a seguinte definição acerca desses termos:

- Defeito é uma discrepância, em *hardware*, entre o sistema implementado e seu respectivo projeto.
- Erro é uma saída errada produzida por um sistema defeituoso.
- Falha é a representação de um err, em um nível de abstração funcional.

A definição do autor é voltada para *hardware*, mas também podemos adapta-la para um sistema híbrido, onde *hardware* e *software* coexistem. Nesse contexto, o defeito seria um problema na construção (física, ou lógica), um erro seria a saída errada, produzida pelo defeito, e a falha seria a manifestação do erro, porém em um nível observável pelo usuário.

Assim como o projeto de um sistema, as falhas são modeladas em níveis hierárquicos. Os níveis mais baixos apresentam maiores detalhes sobre o as falhas, porém com a modularização pode-se ter uma compreensão maior.

O nível físico é o nível básico, conhecido também como nível de componente. Nesse nível ficam os problemas de curtos circuitos e circuitos abertos.

Intermediariamente existe o nível lógico, no qual as falhas são modeladas a nível de portas lógicas. Aqui temos falhas de *stuck-ats*, falhas de temporização, entre outras.

O nível mais alto é o nível comportamental. Esse nível representa falhas funcionais do sistema. Falhas de comunicação podem ser representadas nesse nível.

A injeção de falhas abordada nesse estudo é no nível comportamental, por ser falha de comunicação I^2C. Falhas nesse nível não necessariamente são a nível elétrico. Elas podem ser modeladas de acordo com a aplicação em questão (BUSHNELL; AGRAWAL, 2000).

As falhas podem estar associadas à um barramento, à um pino de um CI ou até mesmo a uma interferência oriunda do ambiente. Além disso, as falhas podem ser permanentes ou transiente. Falhas permanentes são causadas por um dano físico em algum componente do sistema. Esse dano pode ser proveniente do processo de fabricação ou do desgaste do uso do sistema. Falhas transientes acontecem por algum tempo e desaparecem, podendo voltar a ocorrer.

Neste estudo, a injeção de falhas feita foi falha de comunicação, porém o ATE desenvolvido tem capacidade de implementar outras falhas como falhas de temporização na comunicação, além de *stuck-ats* dos fios do barramento I^2C.

2.5 Tipos de Teste

A atividade de desenvolvimento de um sistema é uma tarefa complexa e está sujeita a uma série de problemas. Esses problemas são inerentes ao desenvolvimento de um projeto, e eles ocorrerão mesmo com a utilização de métodos e ferramentas de engenharia (DELAMARO; MALDONADO; JINO, 2007). No desenvolvimento de um produto de alta qualidade, várias etapas de testes são executadas, a fim de garantir a qualidade do projeto. Quanto mais crítica é a aplicação do projeto a ser desenvolvido, mais se faz necessário boas técnicas de testes, visando o maior coeficiente de cobertura de falhas.

Ao contrário do que muitos pensam, o objetivo do teste não é provar que um sistema não possui falhas, mas sim encontrar as falhas que tal sistema possui, antes desse ser dado como pronto. Quanto mais cedo um problema for detectado, menos custosa será a correção deste. Suponhamos que foi detectado um sério problema arquitetural em um dado CI. Caso este CI já tenha sido vendido, dependendo da criticidade de sua tarefa, será necessário trocar as unidades que foram vendidas. Por outro lado, se o problema for descoberto na fase de projeto, o CI passará por um processo de correção, sem necessidade de um *recall*.

Com o aumento da complexidade dos sistemas, podem ser necessários testes em várias etapas do desenvolvimento de um produto. Os teste vão desde a etapa de especificação, que verifica se as especificações são coerentes com o que se espera do projeto, até a nível de aplicação, onde são verificadas funcionalidades a nível de usuário.

Mesmo após um produto ser criado, produzido e vendido, pode ser necessário a avaliação constante das condições do produto. Nesses casos, durante a etapa de projeto, são previstos mecanismos de detecção de falhas, que são embarcados no sistema (BISTs). Esses testes podem ser do modo *online* ou *offline*. Os testes *online* são executados durante o funcionamento do sistema, enquanto os *offline* param a execução do sistema, ou são rodados em momentos em que o sistema não está ativo, como na inicialização ou encerramento de um ciclo de atividades.

Um exemplo clássico desse tipo de teste é o teste de memória, que faz um auto teste durante a inicialização do sistema.

Entre os testes executados antes da venda de um produto, está o teste funcional, o qual é abordado neste estudo. O teste funcional tem por objetivo encontrar falhas em requisitos funcionais do projeto, verificando inconsistências entre a especificação e a execução real de um sistema. Para que esse tipo de teste seja realizado com a maior eficiência possível, é necessário conhecer a arquitetura do sistema. Com esse conhecimento é possível gerar casos de teste que excitem partes sensíveis do projeto, ou seja, casos de teste que possuam a maior probabilidade de revelar erros possível.

Por vezes, é necessário mais do que apenas entradas de usuário para que os casos de teste tenham uma cobertura de falhas aceitável. Dependendo da área que se quer excitar, é necessário fazer variações de temperatura, incidência de ruídos e até mesmo injeção artificial de falhas, o que será visto no capítulo 2.6.

2.6 Mecanismos de Injeção de Falhas

Equipamentos tolerantes a falhas são equipamentos capazes de contornar ou se recuperar de um determinado conjunto de falhas. Esses equipamentos devem conseguir, de alguma forma, detectar falhas e se recuperar delas.

Alguns equipamentos tolerantes a falhas conseguem se recuperar e partir do estado imediatamente anterior à falha (*forward recovery*). Outros, ao se recuperar, voltam a um ponto pré determinado (*backward recovery*), como um *reset*. *Forward recovery* é um método mais custoso, pois é mais complexo que o *backward recovery*, porém proporciona um menor tempo de indisponibilidade.

Muitas vezes o teste de equipamentos tolerantes a falhas não é trivial. Isso se dá pelo simples fato de que as falhas não ocorrem frequentemente. Em vista disso, são desenvolvidos mecanismos de injeção de falhas, que tem por objetivo injetar artificialmente falhas, a fim de avaliar a efetividade dos mecanismos de tolerância a falhas de um dado equipamento ou sistema (HSUEH; TSAI; IYER, 1997).

Com injeção de falha, o sistema sobre teste se comporta como se tivesse um problema de fabricação ou comportamental (HSUEH; TSAI; IYER, 1997). Em caso de problema comportamental, o comportamento falho injetado é definido através de estatísticas e conhecimento prévio sobre o funcionamento falho do sistema. Quando a injeção se trata de um problema no processo de fabricação, como curtos circuitos, circuitos abertos ou *stuck-ats*, a injeção feita é baseada em uma análise estrutural dependente do nível de abstração da injeção a ser feita.

Os testes envolvendo injeção de falhas podem ser feitos basicamente em dois níveis: injeção de falhas baseado em simulação e baseado em protótipo. Injeção de falhas baseada em simulações é utilizada na etapa de projeto de um sistema, onde o sistema ainda não existe de fato. Já a injeção de falhas baseada em protótipo é executada com o sistema real, na sua fase de teste. (HSUEH; TSAI; IYER, 1997)

Os mecanismos de injeção de falhas podem ser divididos em três tipos: baseados em *hardware*, baseados em *software* ou híbridos.

Os mecanismos baseados em *hardware* são sistemas físicos adicionais ou alterações do *hardware* do projeto alvo de testes. Eles visam injetar artificialmente um determinado conjunto de falhas. Essas falhas podem ser fenômenos como curtos circuitos, circuitos abertos, flutuação de tensão de alimentação, entre outros.

Quando baseada em *software*, a injeção de falhas pode ser feita alterando o código a nível de compilação, com *#ifdefs*, da linguagem C, por exemplo, o que faz com que a injeção de falhas seja estática. Outra forma, é adotar uma abordagem, ainda em *software*, onde as falhas podem ser controladas em tempo de execução. Essa solução implica em uma maior intrusão no *software* "real", o que pode introduzir problemas não existentes ou mascarar falhas existentes no projeto original, podendo causar falsos positivos e/ou falsos negativos.

A solução híbrida adota as duas abordagens. *hardware* e *software*, em conjunto.

Enquanto mecanismos em *hardware* podem ter um alto custo de tempo e de recursos financeiros, devido ao fato de ele necessariamente precisar ser construído, os mecanismos baseados em *software* geralmente exigem apenas custo de tempo. Porém, os mecanismos baseados em *hardware* geralmente oferecem um risco de distorção de resultado muito menor que os baseados em *software*. Isso ocorre pelo fato de que quando um módulo extra é introduzido no *soft-*

ware, o comportamento do sistema como um todo sofre uma perturbação ativa, proveniente dessa modificação, que pode alterar sua resposta.

Além das questões como custo e perturbação no sistema, antes da escolha de qual abordagem seguir é necessário conhecer bem qual o tipo de falhas se deseja injetar. Algumas falhas simplesmente não podem ser injetadas por uma das abordagens. Por exemplo, uma falha de flutuação de tensão não pode ser injetada em *software*, assim como valores inválidos em um campo de entrada não podem ser injetados através de *hardware*.

Existem também ocasiões onde o mais adequado é a solução híbrida, usando as duas metodologias para implementar diferentes partes do mecanismo de injeção de falhas.

3 *Solução Proposta*

Como solução para a dificuldade de automatização de testes com SFPs, este trabalho propõe um ATE, baseado em protótipo (ler sessão 2.6), utilizando metodologia híbrida (*hardware* e *softwate*), que funciona como um mediador entre os SFPs e o equipamento hospedeiro de SFPs, implementando um módulo de injeção de falhas e um multiplexador de SFPs, que serão detalhados neste capítulo.

Em uma ligação normal, o SFP é ligado diretamente ao *switch* (figura 3.1), e então ligado à outro dispositivo de rede, o que obriga intervenção física em caso de remoção, inserção ou troca de módulos SFP. Com a solução proposta, o SFP será ligado em um equipamento intermediário que, por sua vez será ligado ao *switch* (figura 3.2). Assim é possível atrelar mais de 1 SFP à mesma porta do *switch*, sendo escolhido qual estará realmente ligado ao *switch* através do controle de uma cadeia de multiplexação.

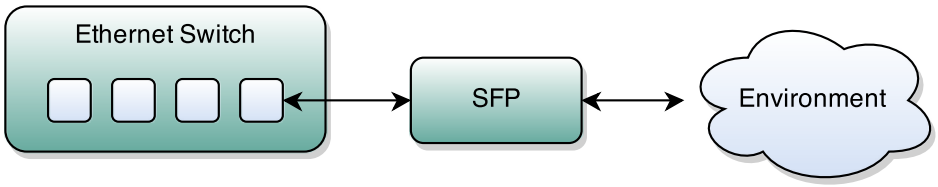

Figura 3.1: Operação normal entre *switch* e SFP.

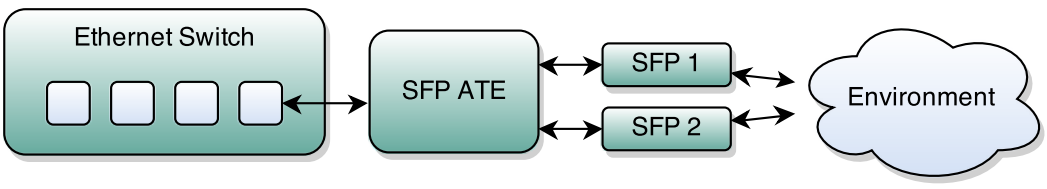

Figura 3.2: Operação com intervenção do ATE entre *switch* e SFP.

Além de possibilitar a inserção, remoção e troca de SFPs sem intervenção física, o ATE é dotado de um módulo de injeção de falhas, o qual pode facilitar o teste de tolerância a falhas.

A abordagem utilizada pode ser aplicada não somente nos equipamentos usados neste caso de estudo, mas sim em qualquer equipamento principal que trabalha em conjunto com um equipamento secundário. Essa generalização é muito poderosa, pois possibilita a aplicação dos conceitos um uma larga gama de aplicações.

3.1 Multiplexação de SFPs

Em funcionamento normal, os SFPs são ligados diretamente ao *switch*, como mostra a figura 3.1. A multiplexação dos SFPs é feita com uma cadeia de multiplexadores que faz a escolha de qual SFP está ligado ao *switch* (figura 3.3). Os SFPs tem diversos pinos, os quais carregam diferentes sinais, desde alimentação até dados e alerta de erros. Esses pinos passam pelos multiplexadores que, dependendo da configuração feita no ATE, ligaram o *switch* em um dos SFPs ou nenhum deles.

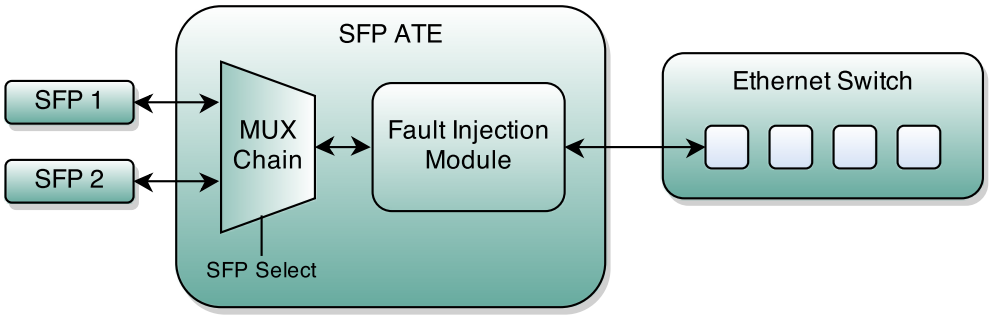

Figura 3.3: Visão geral do ATE.

Com multiplexação dos SFPs é possível fazer a inserção e remoção de SFPs de forma muito semelhante ao procedimento manual. Desta forma, o processo automatizado é transparente para o equipamento sobre teste, que, por sua vez, produz as mesmas respostas de quando o processo de inserção e remoção é feito manualmente.

Idealmente, a multiplexação deveria ocorrer em todos os sinais, porém a multiplexação dos sinais de TX e RX não fazem parte do escopo deste estudo. Por serem sinais de alta frequência (na ordem de GHz), os sinais TX e RX exigem roteamento e terminações especiais, o que não faz

parte do objetivo deste estudo. Contudo, com caráter experimental, a multiplexação dos sinais de TX e RX foi feita. Em testes básicos, o *link* foi estabelecido, porém não foi feito testes de *stress* de tráfego de dados. Com o controle dos sinais de TX e RX, é possível também simular a conexão e desconexão dos cabos, a fim de testar protocolos de comunicação em situações de falhas de *link*.

O processo ocorre com o mínimo de intervenção possível. Pinos do SFP que trazem os mesmo sinais (em especial pinos de alimentação) são unidos no *layout* da PCB.

3.2 Módulo de Injeção de Falhas

Os SFPs se comunicam através de um barramento I^2C, o qual dá acesso aos dispositivos internos do SFP. Em casos de falha de um dos dispositivos internos, ou até mesmo problemas na comunicação I^2C, o sistema pode entrar em estado de falha. Esse tipo de falha é difícil de ser testada, pois raramente se dispõe de SFPs defeituosos, além do que não existem maneiras de configurar o SFP para não responder à comunicação I^2C. Essa dificuldade de teste é a maior motivação para a integração de um módulo de injeção de falhas ao ATE.

O módulo de injeção de falhas atua na fase seguinte à multiplexação dos SFPs (figura 3.3), na comunicação I^2C, podendo injetar falhas em diversas áreas, como PHY, DD e informações sobre o SFP. A injeção de falhas é implementada parte em *hardware* e parte em *software*. A parte do *hardware* é responsável pela separação dos sinais I^2C, enquanto o *software* faz o tratamento das informações, implementa as máquinas de estados I^2C e a injeção de falhas propriamente dita.

O ATE funciona como um escravo I^2C do ponto de vista do *switch* e como um mestre I^2C do ponto de vista do SFP, como mostra a figura 3.4. Essa arquitetura permite que o ATE decida se a informação recebida por um dos lados deve ir até o outro, até mesmo devolvendo respostas erradas ao requisitante.

3.3 Implementação

Como já dito, a implementação do projeto é baseada em uma metodologia híbrida de *harwdare* e *software*. O *harwdare* é composto pelas PCBs, multiplexadores e conectores do projeto,

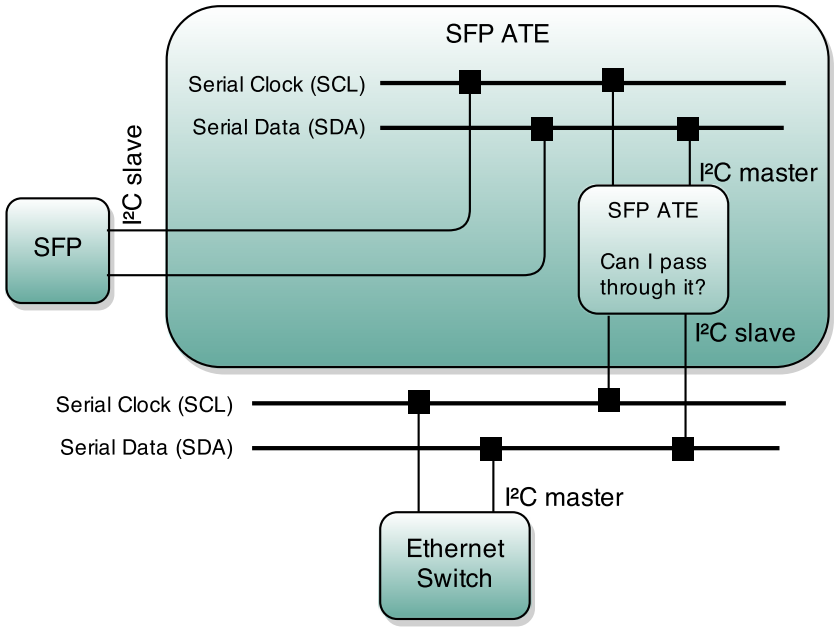

Figura 3.4: Monitoramento e injeção de falhas nos acessos I^2C.

enquanto o *software* contem o controle do *hardware*, implementa a injeção de falhas e provê a interface com o usuário.

3.3.1 Arquitetura de Hardware

O projeto de *hardware* visa criar uma infraestrutura física capaz de fazer a multiplexação de todos os sinais do SFPs, além de disponibilizar os sinais I^2Cs para que o processador manipule-os. A multiplexação deve ser o mais transparente possível para o *ethernet siwtch*, alvo de testes. Essa transparência garante que as características do SFP sejam observadas pelo *ethernet switch* com a mínima interferência do equipamento de teste. A figura 3.5 mostra como está dividido o *hardware* e será detalhada ainda nesta sub sessão.

Dentre os sinais do conector SFP, existem 4 que se destacam. Esses sinais são os pares diferenciais TX e RX. Os pares diferenciais TX e RX são os sinais que carregam os dados úteis ao usuário recebidos (RX) e enviados (TX) pelo SFP. A frequência de operação desses sinais está diretamente relacionada com o *bit rate* do SFP. Por exemplo, se tivermos um SFP de 1Gbps, os dados que trafegam nos pares diferenciais TX e RX, trafegam em uma frequência em torno de 1GHz, o que torna esses sinais extremamente sensíveis.

Em virtude disso, o *hardware* foi dividido em duas PCBs: (1) *High Speed* PCB (HS-PCB), que faz a conexão com o *ethernet switch* e a multiplexação dos pares diferenciais, e (2) a *Low Speed* PCB (LS-PCB), que faz a multiplexação dos demais sinais e a conexão com o processador. Essa divisão foi feita a fim de diminuir o tamanho físico da PCB que fica conectada ao *ethernet switch* e, principalmente, isolar os sinais dos pares diferenciais TX e RX a fim de obter a menor distância de trilha possível, reduzindo assim as chances de interferência e degradação dos sinais. A HS-PCB, além de fazer a conexão com o *ethernet switch*, faz a conexão com os SFPs e envia os sinais lentos para a LS-PCB, como mostra a figura 3.5.

Além das PCBs que serão desenvolvidas no presente TCC, é utilizado um kit de desenvolvimento ARM (ARM-DEV-KIT). Este kit de desenvolvimento possui um microcontrolador ARM, o qual é conectado à LS-PCB.

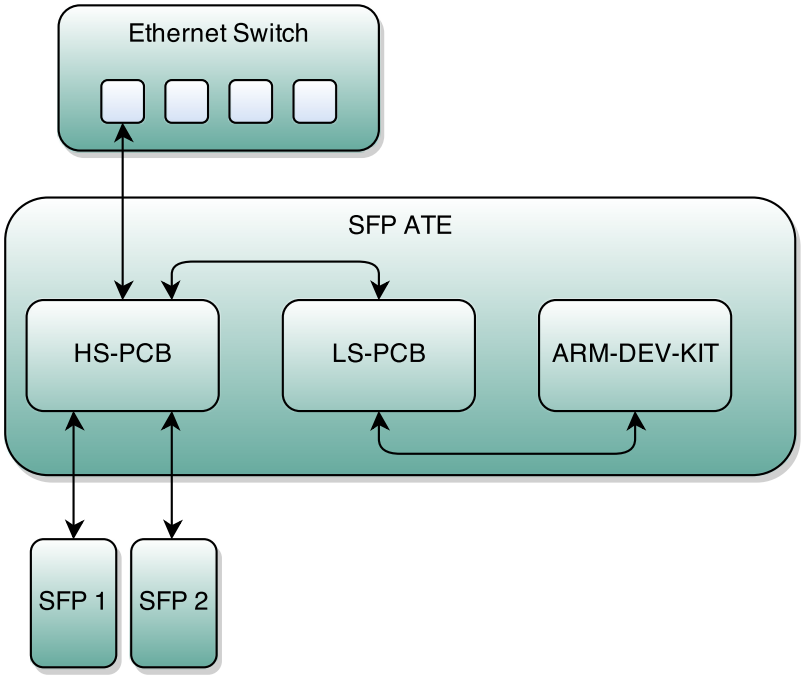

Figura 3.5: Projeto de *hardware*.

HS-PCB

A HS-PCB (figura 3.6) é basicamente uma PCB conector. Além de fazer a conexão com o *switch ethernet*, com os SFPs e com a LS-PCB, a HS-PCB faz a multiplexação dos sinais de

dados, os pares diferencias TX e RX.

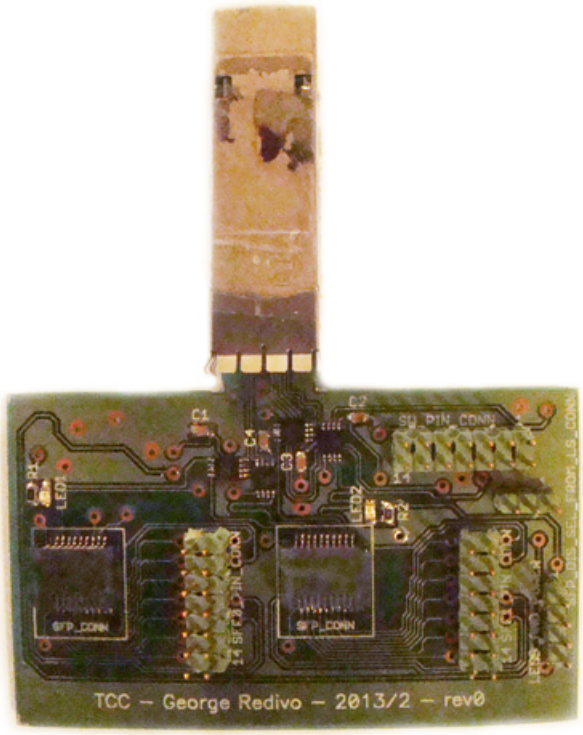

Figura 3.6: Imagem da HS-PCB.

A multiplexação dos pares diferenciais é, como todas as outras, controlada pelo ARM-DEV-KIT através de dois sinais de controle independentes. Esses sinais de controle passam pela LS-PCB antes de chegar na HS-PCB, porém isso acontece puramente por questões mecânicas de montagem do ambiente. Para fazer a multiplexação dos pares diferenciais foi utilizado o CI ADG918, da Analog Devices. Segundo o fabricante, este CI implementa um multiplexador CMOS e trabalha em alta frequência, na casa de GHz, o que atende as necessidades do sinal trafegado nos pinos de TX e RX do SFPs.

LS-PCB

A LS-PCB (figura 3.7) faz a multiplexação dos sinais lentos dos SFPs e conexão com o ARM-DEV-KIT, além de um rebaixamento de tensão que é enviado para a HS-PCB para ser usado nos CIs ADG918.

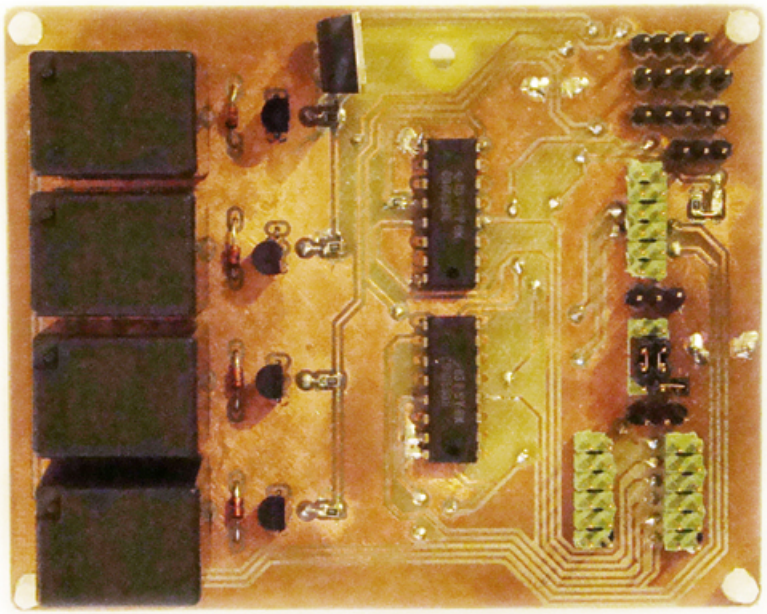

Figura 3.7: Imagem da LS-PCB.

A figura 3.8 mostra o *layout* em blocos da PCB. A multiplexação dos SFPs é feita através de relés e multiplexadores CMOS CD4053. Os CIs CD4053 fazem a multiplexação dos sinais de dados, exceto os sinais I^2C, que são conectados em forma de barramento. Apesar das vantagens, como tamanho reduzido e simplicidade de montagem dos CD4053s, a multiplexação da alimentação dos SFPs foi feita através de uma cadeia de relés. Segundo o MSA descrito em (SFF COMMITTEE, 2001), a corrente de alimentação dos SFPs, pode chegar a 300mA. Com um limite nominal de 15mA por porta, não foi possível utilizar o CI CD4053 para a multiplexação dos sinais de alimentação. Em virtude disso, a opção de uma cadeia de relés se mostrou atrativa devido à facilidade de montagem do circuito e máxima corrente suficientemente grande para a aplicação. Além das vantagens iniciais, a cadeia de relés proporciona um ruído na troca de posição muito semelhante ao ruído da conexão e desconexão manual dos SFPs, devido à característica de contato seco, ou seja, a troca de posição ocorre mecanicamente.

A ligação I^2C, mostrada na figura 3.9, é feita de forma a isolar os SFPs do *ethernet switch*, fazendo com que o ATE seja a ligação entre os dois. Na LS-PCB existe um barramento I^2C que liga uma interface escrava do ARM-DEV-KIT com o *switch*. Esse barramento recebe as requisições I^2C do *switch*, que é o mestre I^2C do barramento. Os SFPs estão ligados em um outro barramento onde também está ligado uma interface mestre do ARM-DEV-KIT. Esse barramento

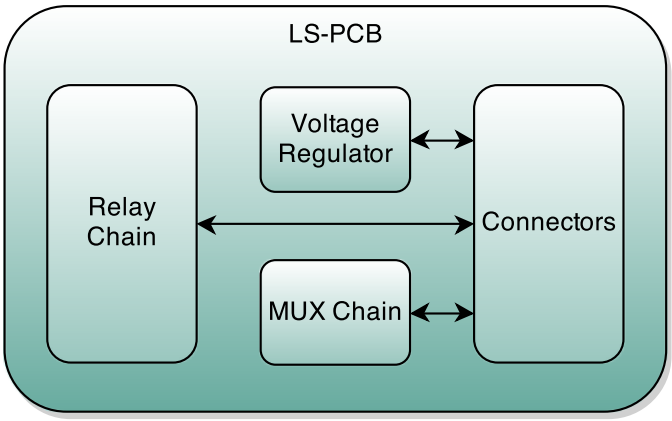

Figura 3.8: Imagem de blocos da LS-PCB

repassa as requisições I^{2}C permitidas pelo módulo de injeção de falhas.

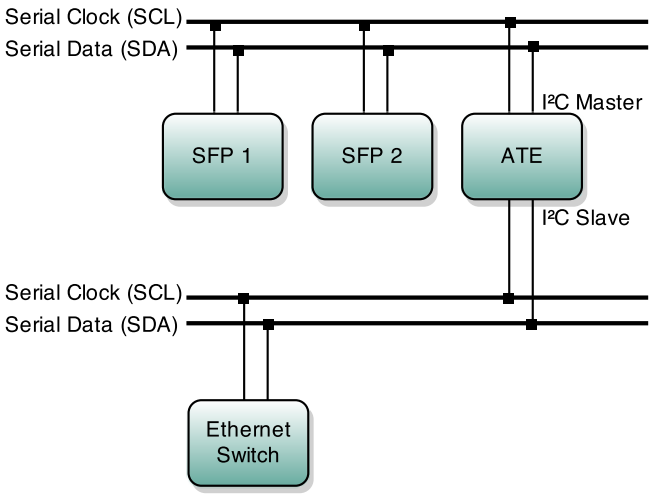

Figura 3.9: Ligação do I^{2}C entre ATE, SFP e *ethernet switch*

Os SFPs normalmente tem os mesmos endereçamentos em seus dispositivos I^{2}C internos. Para evitar colisão de informação é imprescindível que os dois SFPs ligados no ATE nunca fiquem ligados simultaneamente, ou seja, sempre que a alimentação de um dos SFPs é ligada, a alimentação do outro necessariamente deve estar desligada. Todavia, o *software* de controle faz testes de coerência para evitar esse tipo de problema.

O circuito rebaixador de tensão usa o CI LM317 em conjunto com um esquema de filtragem tensão proposto no *datasheet* do CI. Como citado anteriormente, ainda nesta sessão, a tensão

rebaixada é utilizada nos multiplexadores GHz localizados na HS-PCB. A tensão é rebaixada de 5V para 2,75V, para ficar dentro da faixa de tensão admitida pelo CI.

3.3.2 Arquitetura de Sofware

O controle da multiplexação dos sinais dos SFPs e a injeção de falhas da comunicação I^2C é feita através de um microcontrolador ARM. O *software* desenvolvido para o ATE está embarcado neste controlador, sem fazer qualquer alteração no *software* do sistema alvo de teste. Essa abordagem elimina qualquer perturbação decorrente de mutação do programa alvo do teste, uma vez que este não é, de forma alguma, alterado.

O *software* contém o programa que controla todo o projeto. Programa este, escrito na linguagem C.

A arquitetura do *software* (figura 3.10) foi modularizada a fim de facilitar o desenvolvimento e manutenção do código. Existem basicamente quatro grandes módulos dividido em três camadas de abstração. A modularização em camadas de abstração bem definidas permite que o projeto seja modificado com o mínimo de intrusão possível. Por exemplo, caso o processador fosse trocado, apenas a camada de abstração de *hardware* necessitaria ser alterada, ou se a interface com o usuário necessitasse de um novo modelo, as camadas mais baixas não seriam afetadas.

Inicialmente, a camada de interface faz a interface com o usuário. É responsável pelo tratamento das entradas e saídas do programa, para o alto nível.

A camada de configuração faz a análise dos dados inseridos pelo o usuário, coerências, salva as configurações e se comunica com a camada inferior para pedir acesso ao processador.

Por fim, a camada de acesso ao *hardware* possui toda a tradução de comandos dependente de processador. Essa camada é responsável por atender as requisições da camada de configuração e tratar interrupções de oriundas do *hardware*.

Existe um módulo intermediário, localizado na camada de configuração, chamado de módulo de *Debug*. O módulo de *Debug* não necessariamente necessita estar presente na imagem a ser carregada no processador. Isso ocorre pois a camada de *Debug* é utilizada para fazer verificações e testes durante o desenvolvimento. Essa camada dá à quem a usa a possibilidade de ler e escrever em qualquer registrador do processador, além de fazer leituras e escritas no barramento I^2C.

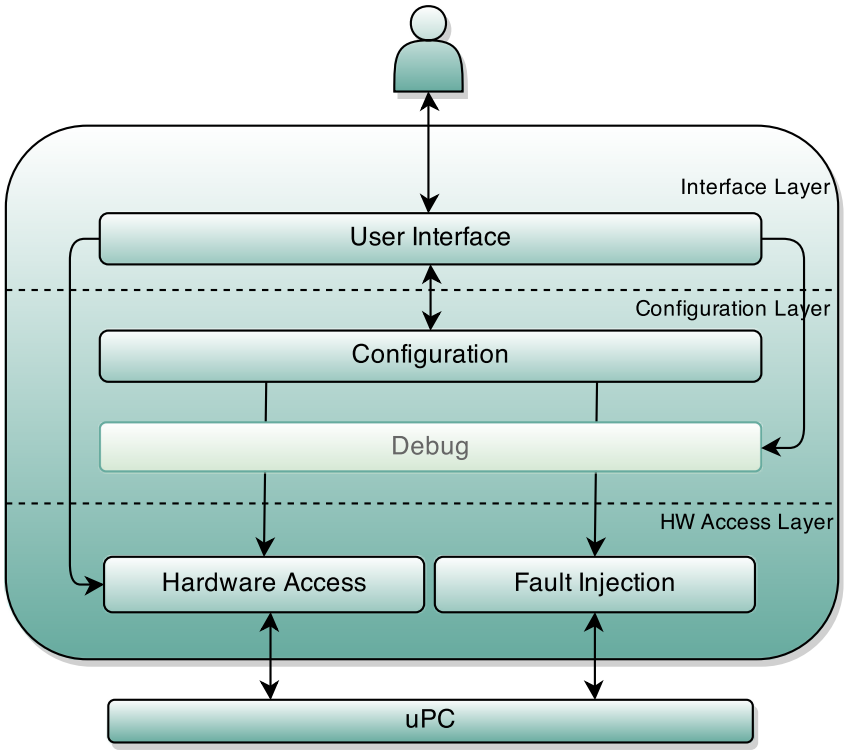

Figura 3.10: Arquitetura do *software*

Módulo de Interface com o Usuário

Este módulo, localizado na camada de Interface, provê uma interface com usuário através de um terminal por linha de comando. É possível inserir comandos de ação e comandos de visualização do estado atual.

É através deste módulo que se dá toda a comunicação do ATE desenvolvido neste trabalho com o mundo exterior. Através dele é feita a configuração de injeção de falhas e multiplexação dos SFPs.

O módulo possui entradas e saídas padrões, o que torna possível a execução através de *scripts* de teste e, consequentemente, a automatização de testes.

Além da óbvia comunicação com o usuário, este módulo também se comunica com os módulos de configuração, módulo de acesso ao *hardware* e módulo de *debug*. A comunicação com o módulo de configuração é destinada a alterar o estado da configuração e buscar informações sobre a configuração atual. A comunicação com o módulo de acesso ao *hardware* se dá simples-

mente para que o *hardware* faça a implementação da comunicação serial, que é utilizada para entra e saída de informações.

Módulo de Configuração

Para fazer a configuração do ATE existe um módulo de configuração. Este módulo, ao ser inicializado, parte de uma configuração padrão, onde os dois SFPs estão desconectados e a injeção de falhas está desabilitada. A partir da inicialização, é possível alterar o estado da configuração através da entrada de comandos pela interface com o usuário.

O módulo de configuração, além de salvar as configurações (em memória volátil), implementa funções que faz toda a verificação da configuração a ser aplicada, a fim de barrar inconsistências, como inserção simultânea de dois SFPs. Também é dever deste módulo solicitar a alteração do nível lógico dos pinos de controle dos multiplexadores, relés e LEDs de informação.

Além de ser ligado com a interface com o usuário, o módulo de configuração é, quando ativo, ligado ao módulo de *debug*, explicado anteriormente. Além disso, este módulo faz ligação com outros dois módulos: módulo de acesso ao *hardware* e módulo de injeção de falhas. O acesso ao *hardware* faz a tradução de atribuição dos pinos para os respectivos endereços, enquanto o módulo de injeção de falhas é apenas avisado das configurações das falhas a serem executadas.

Módulo de Acesso ao Hardware

Este módulo é muito simples. Ele basicamente faz a tradução das requisições de atribuição feitas pelo módulo de configuração para o endereço de memória relacionado ao registrador pretendido, além da implementação de terminal serial e máquina de estados I^2C.

Apesar de simples, esse módulo é de extrema importância da manutenibilidade do projeto. É ele que faz a abstração do processador para o resto da aplicação. Sem ele as atribuições e leituras de registradores estariam espalhadas por todo o código, o que tornaria extremamente custosa uma refatoração de código motivada por uma troca de processador.

Módulo de Injeção de Falhas

O ATE proposto implementa um módulo de injeção de falhas que trabalha na comunicação I^2C entre o SFP e o *ethernet switch*. O módulo de injeção de falhas implementa em *software* uma máquina de estados I^2C com falhas controladas. Este controle é enviado pelo módulo de configuração.

A máquina de estados I^2C de injeção de falhas faz diversas verificações onde testam a habilitação da injeção de algumas falhas propostas neste estudo. Quando uma ou mais falhas são habilitadas, a máquina de estados se comporta de forma não padrão, ou não esperada pela interface I^2C mestre, o que gera a falha.

As falhas escolhidas são falhas na comunicação I^2C. O projeto implementa as seguintes falhas:

- Falha na leitura e/ou escrita geral de um dispositivo (controlado por endereço de dispositivo);
- Falha na leitura e/ou escrita registrador específico;
- Retorno errado (pré estabelecido) na leitura de um registrador específico.

As falhas podem ser injetadas de forma geral em um endereço de dispositivo I^2C ou em um registrador específico de um endereço específico de dispositivo I^2C. Essa granularidade permite a injeção de vetores de falhas mais precisos, testando diversos aspectos da comunicação I^2C.

Com o controle da comunicação I^2C é possível a injeção não somente das falhas implementadas nesse estudo. Falhas mais específicas podem sem implementadas, como um maior atraso no tempo de resposta a uma requisição, ou falhas esporádicas. Uma vez que se tem o controle da máquina I^2C, as falhas passíveis de injeção, dado este projeto como base, são inúmeras e implementáveis com facilidade, devido à modularização da arquitetura de *software*.

4 Validação e Avaliação

As verificações e validações foram feitas continuamente durante o processo de desenvolvimento do projeto. Essa abordagem evita a propagação de erros e diminui o custo associado ao re-trabalho em casos de defeitos encontrados.

As verificações feitas podem ser divididas em três etapas básicas: (1) teste de montagem, (2) teste funcional e (3) teste de integração.

Teste de Montagem

Nessa etapa a PCB é testada a fim de validar a montagem da placa, buscando defeitos de curtos e circuitos abertos decorrente de possíveis danos à placa inseridos no processo de montagem.

Teste de Funcional

O teste funcional destina-se a validar o esquemático da placa de circuito impresso. Nesse teste a PCB é submetida a estímulos a fim de garantir que a PCB está de comportando de maneira esperada aos estímulos injetados.

Teste de Integração

Por fim, o sistema une as duas PCBs e o ARM-DEV-KIT, rodando o *software* desenvolvido para o ATE, e são feitos testes a fim de validar funcionalmente o sistema como um todo, analisando suas funcionalidades finais.

4.1 Teste das PCBs

Durante este estudo, as PCBs desenvolvidas foram testadas a fim de garantir se correto funcionamento.

4.1.1 Teste de Montagem

Durante o processo de teste de montagem, apenas a LS-PCB teve problemas. Enquanto a HS-PCB foi prototipada utilizando um processo de melhor acabamento, o processo utilizado na LS-PCB foi um processo mais simples. Ao contrário da HS-PCB, a LS-PCB não possui verniz, o que torna o processo de soldagem mais perigoso. Além disso, os furos de passagem, que ligam uma camada com a outra, foram feitos manualmente. Devido a esse problemas, alguns curtos e circuitos abertos foram inseridos no processo de montagem, porém após o teste de montagem ter falhado, foi feito uma revisão na placa e os problemas foram encontrados e corrigidos.

4.1.2 Teste Funcional

O teste funcional foi executado separadamente em cada PCB e nos dois casos foi obtido sucesso. As duas PCBs responderam da forma esperada aos estímulos inseridos.

Foi identificado, na LS-PCB, uma melhoria que poderia ser feita. Os sinais I^2C provenientes dos SFPs eram multiplexados da mesma forma que os outros sinais. Com uma análise foi identificado que, como o I^2C é uma arquitetura em barramento, ao invés da multiplexação os SFPs poderiam ser ligados simultaneamente no mesmo barramento, desde os dois SFPs não ficassem ligado simultaneamente. Essa alteração ajuda a diminuir a complexidade do layout da PCB, porém é apenas um melhoramento. Com o intuito de reduzir a complexidade da PCB, foi gerada uma nova revisão de placa contendo essa alteração e outras, detalhadas na sessão 4.2.

4.2 Teste de Integração

Após a validação das PCBs separadamente, deve-se fazer um teste de integração, unindo *hardware* e *software*, testando o projeto como um todo.

4.2.1 Multiplexação

Nos testes de multiplexação dos SFPs foi observado que, em casos de SFPs ópticos, quando o *laser* era ativado, o consumo de corrente era maior que a corrente máxima suportada pela porta do multiplexador, o que causava desligamento do SFP. Em virtude disso o sistema de multiplexação dos sinais de alimentação dos SFPs foi reestruturado a fim de contornar o problema da corrente. Como solução a multiplexação da alimentação dos SFPs foi migrada para uma cadeia de relés.

Após novos testes com a solução de contorno foi identificado que o problema foi resolvido. Em virtude disso, foi gerada uma nova revisão de placa contendo o novo sistema de multiplexação.

Além do novo sistema de multiplexação, a nova revisão da LS-PCB conta com um rebaixador de tensão, que mostrou necessário para a ligação dos multiplexadores utilizados na HS-PCB, além da modificação citada na sessão 4.1.

4.2.2 Injeção de Falhas

O módulo de injeção de falhas foi a última etapa a ser desenvolvida. Depois de todo *hardware* validado e todo o módulo de multiplexação funcionando perfeitamente, iniciou-se o desenvolvimento do *software* e controle de injeção de falhas.

Esta etapa foi desenvolvida em constante processo de teste. Após cada pequena etapa do desenvolvimento o módulo era testado e, quando necessário, ajustes eram feitos.

Vários problemas foram encontrados inicialmente. Problemas estes relacionados com atraso de resposta da interface I^2C, a qual inicialmente não conseguia fazer comunicação com o *switch*. Depois do problema de comunicação ter sido resolvido, pequenos problemas foram encontrados e corrigidos.

Após testes finais, o módulo de injeção de falhas mostrou-se estável e funcional.

4.3 Análise dos Resultados

O projeto de um ATE para equipamentos dotados de SFPs foi testado e validado em um *switch ethernet* fornecido pela empresa Datacom.

Neste equipamento foram inseridos diversos modelos de SFPs e testados os módulos de multiplexação e injeção de falhas. O projeto, que ao final teve todas suas funcionalidades validadas, mostrou-se uma boa alternativa para os testes de SFPs.

Os resultados finais dos testes revelaram este um sistema estável, com facilidade para a ampliação de suas funcionalidades, e que faz o que se propõe: implementa um equipamento que possibilita a automatização de testes de SFPs, provendo injeção de falhas na comunicação I^2C com o SFP em teste.

Apesar de não estar no escopo inicial do trabalho, foi feito também, de forma experimental, a multiplexação dos sinais de TX e RX. O escopo deste trabalho era um ATE que testasse a configuração, porém com a multiplexação dos sinais de TX e RX é possível fazer o *link* entre duas porta. Essa funcionalidade, por não estar no escopo, foi testada superficialmente. Nos testes foi possível observar *link up* entre duas portas, utilizando o ATE, inclusive tráfego de dados.

5 *Considerações Finais*

A complexidade de alguns equipamentos e sistemas impõe um alto nível de dificuldade durante o processo de verificação e validação. A automatização de testes pode ajudar muito em termos de determinismo, tempo de execução e confiabilidade do teste, porém ela pode encontrar, em alguns casos, entraves quanto à implementação da automatização.

Este estudo revela uma alternativa para a automatização de testes de *ethernet switches* que usam SFPs. A automatização desses testes traz mais confiabilidade à estes, hoje feitos manualmente, além de dar mais agilidade e velocidade.

Apesar de o projeto usar um cenário específico como estudo de caso, os conceitos e técnicas utilizados, como multiplexação independente de sinais, e ATEs híbridos (baseados em *hardware* e *software*), podem ser aplicados em diversas áreas onde um equipamento principal trabalha em conjunto com equipamento secundário, ou periférico. Essa generalização torna o projeto uma arma poderosa no teste funcional e validação de novos equipamentos.

Referências Bibliográficas

BUSHNELL, M. L.; AGRAWAL, V. D. *Essentials of Electronic Testing*. [S.l.]: Kluwer, 2000.

CISCO SYSTEMS. *SGMII Specification*. Rev 1.8. [S.l.], 2005. `ftp://ftp-eng.cisco.com/smii/sgmii.pdf`.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao Teste de Software*. [S.l.]: Elsevier, 2007.

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. Abril 1997.

I2C-BUS.ORG. *I2C Bus*. 11 2013. Website. `http://www.i2c-bus.org`.

PHY. 9 2013. Website. `http://pt.wikipedia.org/wiki/PHY`.

SFF COMMITTEE. *INF-8074i*. Rev 1.0. [S.l.], 2001. `ftp://ftp.seagate.com/sff/INF-8074.PDF`.

SFF COMMITTEE. *SFF-8472*. Rev 11.3. [S.l.], 2013. `ftp://ftp.seagate.com/sff/SFF-8472.PDF`.

WANG, L.-T.; WU, C.-W.; WEN, X. *VLSI Test Principles and Architectures*. [S.l.]: Elsevier, 2006.