UNIVERSIDADE DO SUL DE SANTA CATARINA

ANDERSON ANTONIO DE CAMARGO

SISTEMA DE TESTES AUTOMÁTICOS PARA TERMINAL DE LINHA ÓPTICA

Palhoça

2019

ANDERSON ANTONIO DE CAMARGO

SISTEMA DE TESTES AUTOMÁTICOS PARA TERMINAL DE LINHA ÓPTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade do Sul de Santa Catarina como requisito parcial à obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Thiago Felski Pereira Ms.

Palhoça

2019

ANDERSON ANTONIO DE CAMARGO

SISTEMA DE TESTES AUTOMÁTICOS PARA TERMINAL DE LINHA ÓPTICA

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do grau de Engenheiro Eletricista e aprovado em sua forma final pelo Curso de Graduação em Engenharia Elétrica Telemática da Universidade do Sul de Santa Catarina.

Palhoça, 13 junho 2019.

_____

Professor e orientador: Prof. Thiago Felski Pereira, Ms.
Universidade do Sul de Santa Catarina

_____

Prof. André Tonon, Esp.
Universidade do Sul de Santa Catarina

_____

João Marcos Zilio, Eng.
Engenheiro de desenvolvimento da Intelbras

AGRADECIMENTOS

Agradeço a meus pais, Jandir e Teonise, por me apoiarem em todos os momentos da faculdade. A minha esposa Mainara por incentivar a continuar mesmo em momentos difíceis. A minha filha Nicole que apesar de ter apenas 9 meses já é uma fonte de inspiração.

Um obrigado aos professores Thiago F., André T., Djan R. e Sheila T. pela participação no desenvolvimento deste trabalho.

E um agradecimento especial aos meus amigos, João, Thiago, Allyson, Adriano, Johannes e Diego por participarem diretamente e/ou indiretamente na minha formação acadêmica e profissional.

RESUMO

O presente trabalho expõe, conceitos sobre redes ópticas passivas, abrangendo a construção da rede e cada um dos periféricos presentes na mesma, itens essenciais para compreensão dos testes aplicados na manufatura dos equipamentos, assim como a sua automatização. O sistema de teste automatizado de um produto, é popularmente conhecido como Jiga de testes, tem como principal objetivo reduzir tempo de manufatura, e garantir maior robustez aos produtos, viabilizando sua fabricação e comercio. Abordando conceitos e aplicações sobre diferentes tipos e métodos de testes. Também sendo exposto as diferenças entre defeitos, falhas e erros presentes em equipamentos ou em processos. A Jiga para a automatização dos testes, que permitiu ganhos de 78% em comparação com o teste manual, foi desenvolvida utilizando como premissa, a simulação de um cenário real. Os resultados alcançados são apresentados por meio de diagramas e trechos específicos de software.

Palavras chave: Testes, automação, hardware.

ABSTRACT

This paper presents concepts about passive optical networks, covering the construction of the network's infrastructure and each of the peripherals present in it, essential items for understanding the tests applied in the manufacture of the equipment, as well as its automation. The automated test system of a product, popularly known as test giga, has as main objective to reduce manufacturing time, ensuring greater robustness to the products, making feasible its manufacture and trade. Concepts and applications about different types and methods of testing has been shown. Also being exposed the differences between defects, failures and errors present in equipment or processes. The Giga for the automation of the tests, which allowed gains of 78% in comparison to the manual test, was developed using as a premise, the simulation of a real scenario. The results achieved are shown through diagrams and specific patches of software.

Keywords: Tests, automation, hardware.

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

BGA – *Ball Grid Array*

BOSA – *Bidirectional Optical Sub-Assemblies*

CI – Circuito Integrado

CSS – *Cascading Style Sheets*

FTTH – *Fiber to the home*

GPON – *Gigabit Passive Optical Network*

HTML – *Hypertext Markup Language*

IP – *Internet protocol*

JTAG – *Joint Test Action Group*

LED – *Light Emissor Diode*

NFS – *Network file system*

OLT – *Optical Line Terminal*

ONU – *Optical network units*

OS – *Operational system*

PON – *Passive Optical Network*

SCI – *system call interface*

SCP – *Secury Copy*

SFP – *Small Form Factor Pluggable*

SSH – *Secury Shell*

TDM – *Time Division Multiplexer*

SUMÁRIO

1 INTRODUÇÃO

1.1 TEMA:

O tráfego de dados no mundo está em constante crescimento. As resoluções de vídeo em alta definição, o armazenamento de dados na nuvem e o advento da internet das coisas, são alguns dos fatores que demandam uma infraestrutura cada vez melhor para suportar a numerosa quantidade de dados que esses serviços requerem. A fibra óptica teve grande impacto no avanço da qualidade dos serviços que exigem alta taxa de dados, com um meio de transmissão onde as perdas são mínimas e o limite da taxa de dados é definido pela eletrônica e não pelo meio de transmissão. Dessa forma, tornou-se possível a criação de uma infraestrutura altamente tecnológica e moderna para suportar essa demanda de dados, as redes PON (*passive optical network*). (MULLER, 2016)

As redes PON possuem diversas características que a tornam a melhor forma de cobertura para a última milha (conexão com usuário final) das conexões de internet. Entre essas características, podemos citar a alta largura de banda, facilidade no gerenciamento e instalação, ademais da longa distância máxima do ponto inicial até o cliente, fazendo com que as redes PON sejam o ponto facilitador para o que conhecemos como FTTH (*Fiber to the home*), ou seja, a fibra óptica até a casa do cliente. (OLIVEIRA, 2014)

Os maiores esforços para melhorar a qualidade das redes PON, hoje estão concentrados na eletrônica presente na infraestrutura da rede, ou seja, o hardware de transmissão e recepção do sinal, os quais serão detalhados no decorrer deste trabalho. A Broadcom por exemplo, é uma das empresas que está sempre envolvida em pesquisas e padronizações das redes ópticas passivas, sendo uma das maiores fabricantes de circuitos integrados para esse mercado. As próximas gerações das redes PON devem alcançar taxas de dados de 400Gbps. (AVAGO, 2010)

Uma parte essencial do processo de fabricação desses eletrônicos são os testes realizados para garantir a sua qualidade. Razão pela qual este trabalho tem por objetivo apresentar uma proposta de viabilidade técnica e econômica, visando o melhor custo benefício para testar o hardware presente no núcleo das redes GPON (*Gigabit Passive Optical Network*), as OLTs (*Optical Line Terminal*) e que possa ser utilizado como base para novas pesquisas relacionadas ao assunto.

1.2 PROBLEMA DE PESQUISA

Concentradores de interface de dados, como também são conhecidas as OLTs normalmente possuem muitas portas de comunicação, necessitando tempo elevado de gravação de software e testes de fábrica. A OLT que será utilizada para o estudo e desenvolvimento desde trabalho, por exemplo, dispõe de 8 portas preparadas para a comunicação GPON.

Esse equipamento é utilizado normalmente como núcleo de uma rede, o que significa que um defeito deixaria centenas de usuários desconectados, gerando transtornos tanto aos usuários quanto aos provedores de internet. Dessa forma, é necessário testar à exaustão cada equipamento antes de comercializá-lo. O processo manual de gravação e testes de cada OLT prejudica o tempo de fabricação do produto. Além dos testes funcionais, cada placa passa por um teste mais severo denominado *Burn-in*, que consiste em deixar o equipamento em uma condição de estresse térmico, enquanto são realizados testes em suas interfaces por pelo menos 24h, podendo variar até 76h em etapas iniciais de projeto. Esses testes visam encontrar possíveis intermitências em circuitos críticos da placa.

As placas são submetidas a diversos testes manuais, a fim de alcançar a sua aprovação, contudo, o tempo para realização desses testes podem inviabilizar a sua produção em massa. Assim sendo, é inevitável o desenvolvimento de uma Jiga (sistema automatizado de testes), que tem por objetivo automatizar e paralelizar todos ou grande parte dos processos para diminuir o tempo de teste por produto.

1.3 JUSTIFICATIVA:

A Intelbras em conjunto com parceiros, desenvolveu uma OLT 100% nacional. Esse concentrador de interfaces GPON, possui a capacidade de prover internet para até 1024 usuários em um equipamento compacto.

O tempo gasto com processos manuais de teste para equipamentos que exigem um alto grau de qualidade é grande, podendo inviabilizar projetos caso não sejam criadas melhorias nas etapas produtivas, para isso há duas formas de se trabalhar o problema, a primeira é comprar um equipamento específico para realizar as tarefas de produção em geral (incluindo testes), e a segunda é desenvolver equipamentos similares para realizar essas tarefas. Os equipamentos específicos para realizar testes em OLTs podem custar centenas de milhares de reais e para

justificar a sua compra, o volume de produção deve ser alto. A previsão inicial de venda da Intelbras para esse produto não condiz com um investimento desse porte.

Além do ponto de vista econômico, o desenvolvimento de uma Jiga de testes é um desafio para a formação acadêmica, pois, é como o desenvolvimento de um produto, havendo diversas etapas a se considerar para seu desenvolvimento. Compreendendo o projeto da Jiga à construção do hardware e software, agregando ao universitário diversos novos conhecimentos.

1.4 OBJETIVOS

1.4.1 Geral

O objetivo deste trabalho é desenvolver uma Jiga de testes capaz de automatizar e paralelizar os processos de gravação e testes de uma OLT, visando testar mais de 80% dos circuitos presentes na placa, levantando as métricas envolvidas no processo antes e após a utilização de uma Jiga, com auxílio de tecnologia de código livre e equipamentos de fácil acesso e baixo custo.

1.4.2 Específicos

a) Realizar um levantamento dos circuitos presentes na OLT, a taxa de cobertura atual de testes e o tempo individual por processo realizado na fabricação do produto
b) Diminuir o tempo do processo de gravação da placa e de teste funcional;
c) Aumentar a abrangência dos testes realizados, visando melhorar a cobertura de teste dos circuitos presentes na placa;
d) Viabilizar a utilização do teste de *burn-in* para a produção em massa do produto.
e) Estratificar a aplicação de melhorias futuras aos métodos de teste.

1.5 DELIMITAÇÃO

A projeto proposto para a elaboração da Jiga de teste tem como premissa o desenvolvimento e automatização do cenário externo de testes para validar o hardware do

produto OLT8820i da Intelbras. Várias melhorias no sistema de testes podem ser embarcadas no software do produto por possuir um auto teste, o qual é utilizado pelo sistema externo de testes para potencializar os resultados, o auto teste é um software proprietário, não podendo ser exposto em totalidade no trabalho, mas será contabilizado para algumas métricas vide que o desenvolvimento do mesmo é posterior ao início do projeto da Jiga de teste.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, serão expostos conceitos e teorias sobre redes ópticas e sistemas automatizados de testes, comumente chamado de Jiga de testes, expondo de forma detalhada o propósito de cada item utilizado em redes PON e para o desenvolvimento da Jiga.

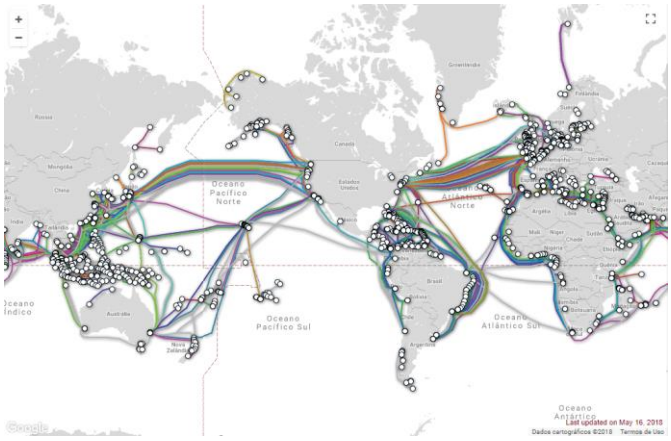
2.1 REDES ÓPTICAS

As redes ópticas estão em grande expansão, a evolução e disseminação delas andam a passos largos, links de 100 Mbps chegando na casa de um cliente comum eram uma utopia no Brasil antes dos anos 2000, atualmente esses links são realidade, e a tendência é a taxa de dados aumentar. Segundo estudo Índice de Rede Visual realizado pela Cisco (2017), a previsão é que globalmente o crescimento na taxa de dados IP (*Internet protocol*) deve triplicar entre 2016 e 2021, passando de 96 EB (Exa Bytes) para 278 EB, significando um crescimento de 24% ano.

Segundo Amazonas (2005), as redes ópticas podem ser divididas em dois principais métodos de transmissão, sendo eles redes ponto a ponto, e redes ponto multiponto, que serão explicadas a seguir.

Redes ponto a ponto: formam os enlaces conhecidos como *Backbones* (Espinha dorsal) das redes IP, é por onde trafega a grande massa de dados por distâncias continentais, esses enlaces interligam por exemplo o continente americano ao europeu, possibilitando o acesso a informações do velho continente em milissegundos, a Figura 01 demonstra os links ponto a ponto ligados por fibra óptica submarina no mundo.

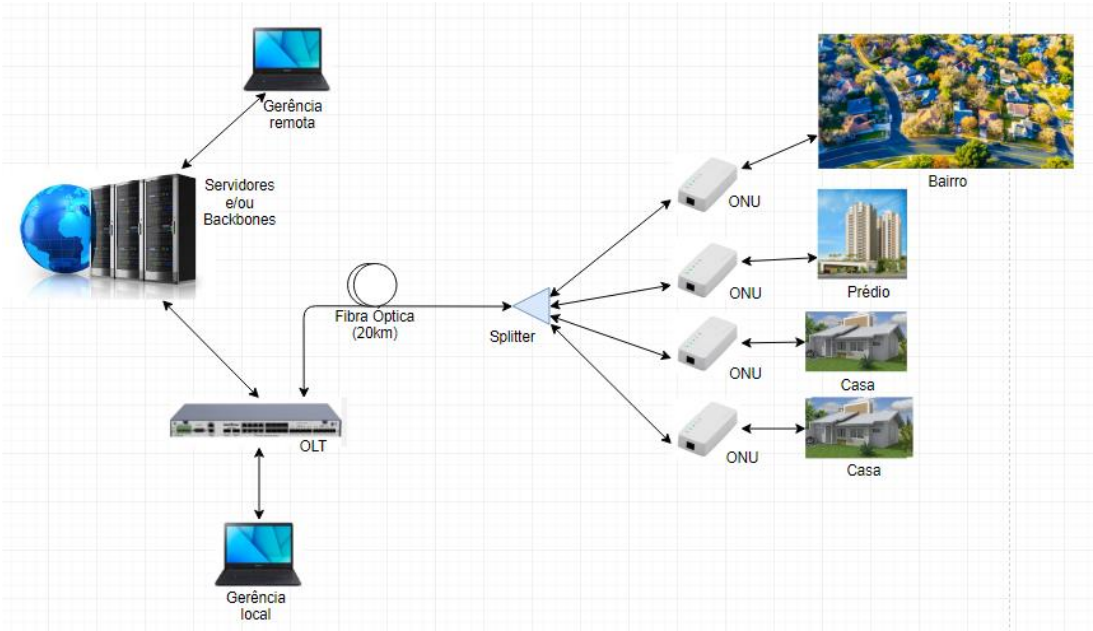
Figura 01 - Mapa das conexões de dados por fibra submarina



Fonte: (SUBMARINECABLEMAP, 2018)

Redes ponto multiponto: Utilizadas para interligar um terminal do *backbone* aos usuários finais, essa conexão é conhecida como *last-mile* (última milha), a conexão aos usuários finais via fibra óptica é conhecida como FTTx, ou seja, a fibra até o x, onde podemos considerar o x como sendo uma casa, um edifício, um bairro, etc. É nessa rede que se encontram os equipamentos PON que são os objetos de estudo deste trabalho. A principal característica dessa rede é a divisão do sinal de luz de forma passiva através de *Splitters* (divisores ópticos), possibilitando com que através de um sinal luminoso seja possível distribuir internet para até 128 ONUs (*Optical network units*), em cada porta de saída da OLT, como observa-se na Figura 02.

Figura 02 - Diagrama de conexão de redes PON



Fonte: O Autor, (2018)

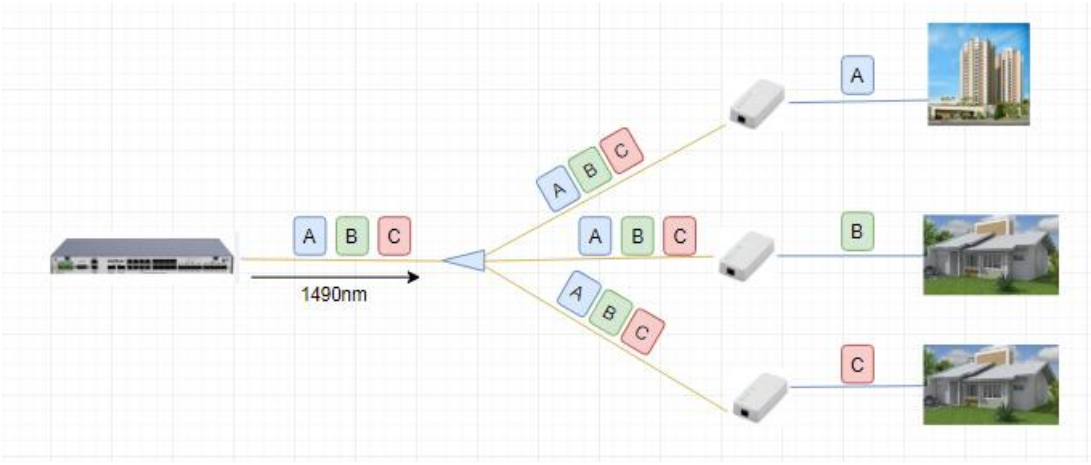
Ao longo da história das redes de internet tiveram alguns saltos de tecnologia que permitiram conexão de banda larga, e a criação das redes PON é o último avanço realizado pelo homem para expandir esse universo. As redes PON são escalonáveis, os seja, a partir do mesmo meio de transmissão é possível aumentar o tráfego de dados com a alteração dos elementos ativos da rede por outros com tecnologia da próxima geração. As Redes PON iniciaram com links de 622Mbps, no momento as tecnologias mais difundidas no Brasil são a EPON (*Ethernet Passive Optical Network*) e a GPON (*Gigabit Passive Optical Network*) com capacidades de taxas de transferência de respectivamente 1,25Gbps e 2,5Gbps no sentido *downstream*, e já existem equipamentos PON no mercado brasileiro com tecnologia que possibilitam links de 10Gpbs. (OLIVEIRA 2014)

2.2 PADRONIZAÇÃO DAS REDES GPON

As redes PON em geral possuem um padrão discutido e estabelecido pela ITU (*International Telecomunication Union* – União Internacional de Telecomunicações), onde o padrão desenvolvido para a comunicação GPON recebeu o nome de ITU-T G984.1. Esse documento traz diversas informações de como as redes GPON devem ser instaladas, bem como, os equipamentos presentes na rede devem se comportar para que tenham compatibilidade entre si. Todas as informações presentes nesse item fazem referência a essa norma.

A tecnologia GPON faz uso de fibras monomodo com comprimentos de onda nas janelas de 1310nm, 1490nm para transmissão e recepção de dados e 1550nm para transmissão de vídeo, a primeira janela em 850nm normalmente é utilizada em fibras multimodo. Conforme a Figura 03, no sentido *downstream* os pacotes de dados são enviados a todas as ONUs conectadas na porta PON da OLT, na ONU é realizado o trabalho de filtrar qual pacote deve ser entregue ao usuário final, há várias formas de realizar esse filtro, o mais comum, e utilizado é a criação de redes virtuais (VLAN).

Figura 03 - Transmissão de dados no sentido downstream



Fonte: O Autor, (2018)

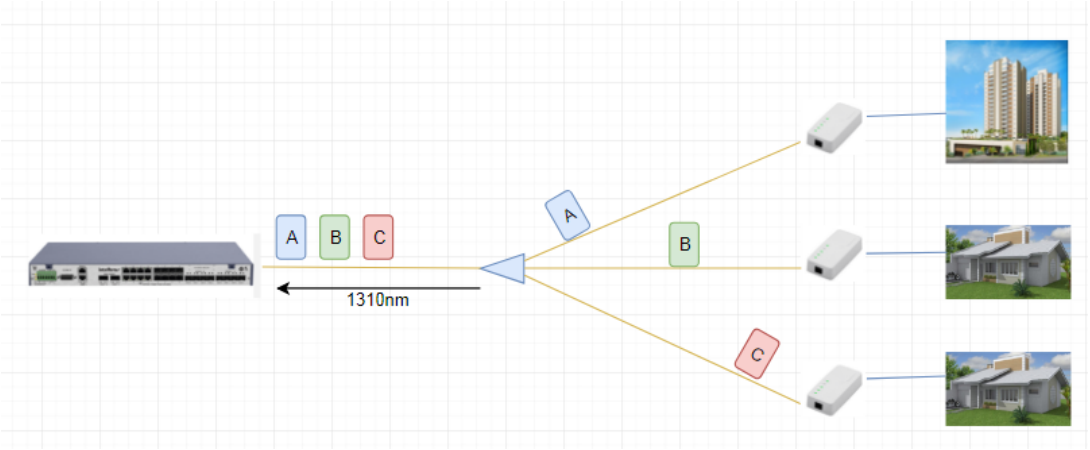
Cada porta PON de uma OLT deve garantir uma transmissão de 2,5Gbps no sentido *downstream*, podendo ser conectada a até 128 ONUs, a uma distância de até no máximo 60km da última ONU do enlace, essa distância é denominada máxima distância lógica entre OLT e ONU. Além disso as ONUs mais próximas e a mais longe da OLT devem estar no máximo 20km de distância, essa distância é denominada máxima distância física do enlace como podemos ver na Figura 04.

Figura 04 - Máximas distâncias dos enlaces PON

Fonte: O Autor, (2018)

Há duas taxas de dados possíveis para o sentido *upstream* de dados, podendo ser elas de 1,25 Gbps ou 2,5Gbps, em geral as OLTs utilizam como padrão 1,25Gbps para *uplink*. No sentido *upstream* o sinal é multiplexado por TDM (Time Division Multiplexer – multiplexador por divisão de tempo) demonstrado na Figura 05. A distância física máxima deve ser respeitada para o TDM não perder o sincronismo, caso isso ocorra a taxa de dados pode cair consideravelmente devido à perda de pacotes. As OLTs trazem uma configuração que pode ser alterada para até 60km de distância física sem perder o sincronismo.

Figura 05 - Transmissão de dados no sentido *upstream*



Fonte: O Autor, (2018)

As redes GPON ainda preveem a transmissão de vídeo no comprimento de onda de 1550nm, a qual é criada por outro gerador de sinal e ligada a fibra do enlace através de um acoplador de multiplexação por divisão de comprimento de onda (*WDM coupler*). Esse sistema é pouco utilizado uma vez que necessita de mais um equipamento na rede, tornando-a mais cara. Usualmente a transmissão de vídeo das redes GPON é realizada no protocolo IP junto ao

comprimento de onda de 1490nm, a vantagem de utilizar o comprimento de 1550nm seria não congestionar o tráfego do enlace com transmissão de vídeo.

2.3 EQUIPAMENTOS GPON

Conforme descrito, as redes PON mesclam a utilização de equipamentos ativos e passivos. Na sequência descreve-se sobre cada um desses equipamentos, como a Jiga será desenvolvida para um produto específico da empresa Intelbras S/A, serão utilizados seus equipamentos para exemplificar cada item presente nas redes.

2.3.1 OLT

A OLT vista na Figura 06 é o centro das redes GPON, é o equipamento responsável por receber os dados ethernet dos *backbones* e retransmiti-los para as ONUs conectadas. Este é o equipamento a ser validado pela Jiga de teste proposta neste trabalho.

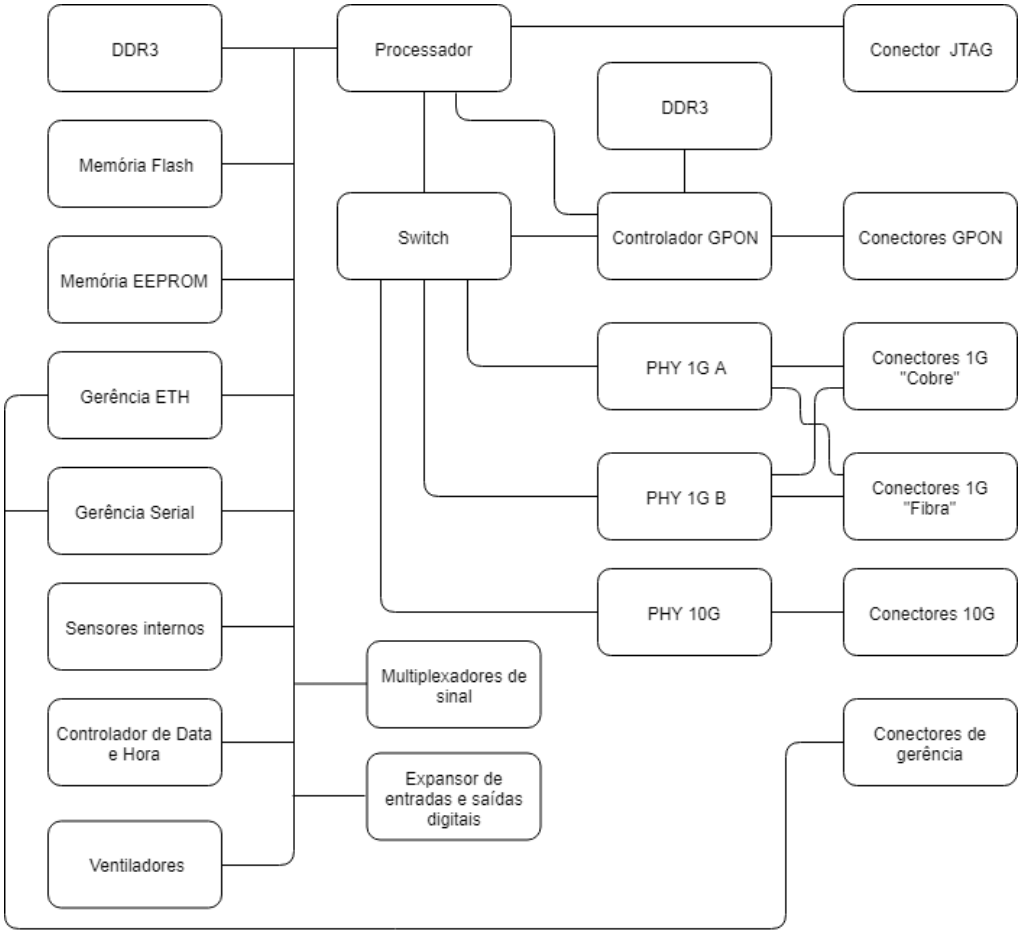
Figura 06 - OLT 8820 i

Fonte: (INTELBRAS, 2018)

A OLT8820i, possui 6 diferentes interfaces de dados, sendo elas as interfaces de gerência ethernet e gerência serial, interface de dados ethernet de 10Gbps e 1Gbps (via cabo e via fibra), e as interfaces GPON. (INTELBRAS S/A, 2018)

O hardware da OLT 8820 i da Intelbras segue diagrama de blocos da Figura 07.

Figura 07 - Diagrama de blocos da OLT 8820 i



Fonte: O Autor, 2018

Com o auxílio do diagrama é mais fácil entender como funciona o tráfego de dados e a sinalização presente na rede GPON. O processador é o concentrador de sinais e informações de funcionamento do sistema, nele estão conectadas as interfaces de gerência, e os canais de gerenciamento do switch e do controlador GPON, sendo responsável por gerenciar todos os circuitos da OLT. O *Switch* é responsável por distribuir o tronco de dados das portas de *uplink* que chegam pelos transceptores (PHYs), para o controlador GPON que tem o trabalho de sincronizar e transmitir os dados para suas respectivas conexões. (INTELBRAS S/A, 2018)

2.3.2 ONU

A Figura 08 mostra uma ONU (*Optical Network Unit*), equipamento que fica na ponta da rede GPON, normalmente colocada o mais próximo possível do usuário final, é responsável por transformar os dados recebidos no pacote GPON para ethernet novamente.

Figura 08 - ONU110

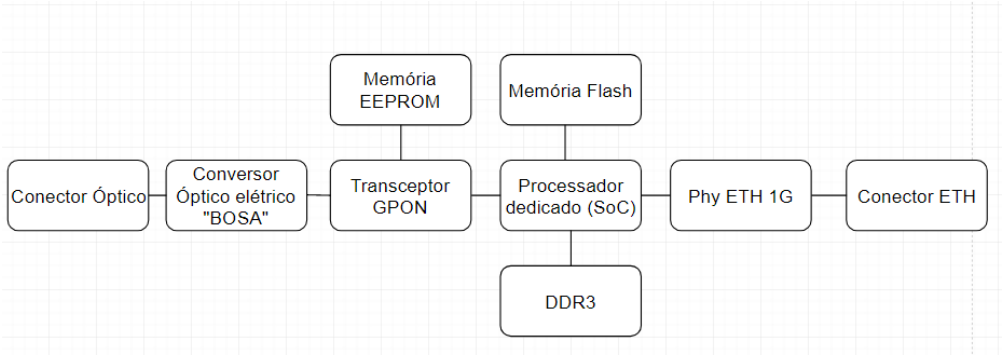
Fonte: (INTELBRAS S/A, 2018)

A ONU110 da Intelbras S/A (2018), dispõe de uma interface ethernet para transmissão de dados, e uma interface óptica para recepção do sinal com as seguintes características:

- Comprimento de onda TX: 1310 nm;
- Comprimento de onda RX: 1490 nm;
- Potência do sinal +0,5 a +5 dBm;
- Sensibilidade de recepção máxima -8 dBm;
- Sensibilidade de recepção mínima -27 dBm.

O circuito da placa da ONU 110 da Intelbras segue diagrama de blocos da Figura 09:

Figura 09 - Diagrama de blocos da ONU



Fonte: O Autor, (2018)

O diagrama da ONU é mais simples ao compararmos com a OLT, conforme pode ser visto no diagrama, o sinal óptico que chega pelo conector, passa por um conversor bidirecional conhecido como BOSA (*Bidirectional Optical Sub-Assemblies*), então o sinal agora elétrico é tratado e retransmitido ao transceptor ethernet, o qual é conectado aos clientes finais. A

memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) é responsável por manter as configurações estáticas da ONU como por exemplo a configuração do módulo BOSA, enquanto a memória *Flash* armazena o software da ONU responsável pela interface homem máquina e as funções de roteamento de dados, já a memória DDR3 (*Double data rate type 3*), é responsável por alocar temporariamente os pacotes de dados, e eventos do processador. (INTELBRAS, 2018)

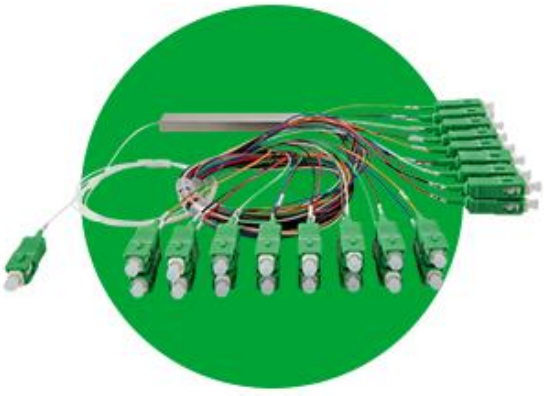
2.3.3 Divisor óptico

Divisores ópticos passivos, mais comumente conhecido como *Splitters*, são responsáveis por ramificar a rede, existem *splitters* de 1:2 até 1:128 divisões, porém os mais utilizados são os 1:32 pois possibilitam a entrega de uma boa taxa de dados. o *downstream* GPON como vimos chega a 2.488Gbps, isso dividido para 32 ONUs disponibiliza um link médio de aproximadamente 77Mbps por ONU. Possuem geralmente uma baixa perda de inserção, podendo ser obtida a partir da fórmula:

$$L = 2 + 10log(N)$$

Onde L é a perda e N o número de divisões da saída do *splitter*, essa atenuação pode variar com a diferentes comprimentos de onda sendo ideal para atuar nos comprimentos de onda de 1.260 a 1.650 nm, a Figura 10 é um exemplo de *splitter* 1:32. (INTELBRAS S/A, 2018)

Figura 10 - *Splitter* 1:32



Fonte: (INTELBRAS S/A, 2018)

2.3.4 Módulo SFP

Na Figura 11 expõe um módulo SFP (*Small Form Factor Pluggable*), esses são conectados às OLTs, responsáveis pela transmissão e recepção da luz utilizada na comunicação, a potência e sensibilidade de sinal dos mesmos define o alcance máximo do enlace, os módulos utilizados na comunicação GPON geralmente são da classe B garantindo um alcance de 20Km com a taxa de dados estabelecida pelo padrão ITU-T G.984. (INTELBRAS S/A, 2018)

Figura 11 - Módulo SFP

Fonte: (INTELBRAS S/A, 2018)

2.4 DEFEITOS, FALHAS E ERROS

Esse tópico se faz necessário para facilitar o entendimento de quais problemas uma Jiga de teste pode evidenciar no hardware de um equipamento. Os conceitos de defeito, erros e falhas, são distintos em diferentes áreas da engenharia, conforme veremos na sequência.

2.4.1 Defeitos

Na engenharia de software um defeito é tratado como um evento que ocorre quando a entrega de um serviço desvia da especificação do sistema de origem, o que causa um resultado incorreto. Podemos exemplificar de forma simples utilizando um programa de edição de imagens, um defeito seria a seleção incorreta de uma cor selecionada, resultando em um final incorreto. (IEEE, 1990)

Em hardware um defeito é a diferença não intencional entre o equipamento montado e o equipamento projetado. Defeitos podem ocorrer durante a produção, proveniente de um

processo mal executado, no transporte, devido as vibrações a que os equipamentos são submetidos ou ainda, ao envelhecimento dos componentes presentes na placa, como a oxidação de terminais e conexões. Um exemplo de defeito é o não funcionamento de um circuito devido à falta de algum componente. (PRADHAN, 1996)

2.4.2 Falhas

Em software falhas podem ser descritas como a inabilidade de um programa em executar suas funções requeridas em relação às funções especificadas dele. Falhas são comumente conhecidas como *bugs*. Ocorre quando defeitos ou erros causam resultados inesperados, podendo gerar a quebra da aplicação ou mau funcionamento de outras funções. Utilizando o exemplo em defeitos, podemos exemplificar uma falha como sendo a falta de uma cor selecionável em uma aplicação de edição de imagens. (IEEE, 1990)

Para hardware uma falha pode ser descrita como o resultado errôneo perceptível produzido por um defeito, ou seja, a presença de um defeito que faz com que uma função do equipamento tenha um resultado não condizente com o esperado. Podendo ser exemplificado pelo não ou mal funcionamento de uma saída de sinal, onde o defeito pode estar em vários pontos do circuito que trata aquele sinal. (PRADHAN, 1996)

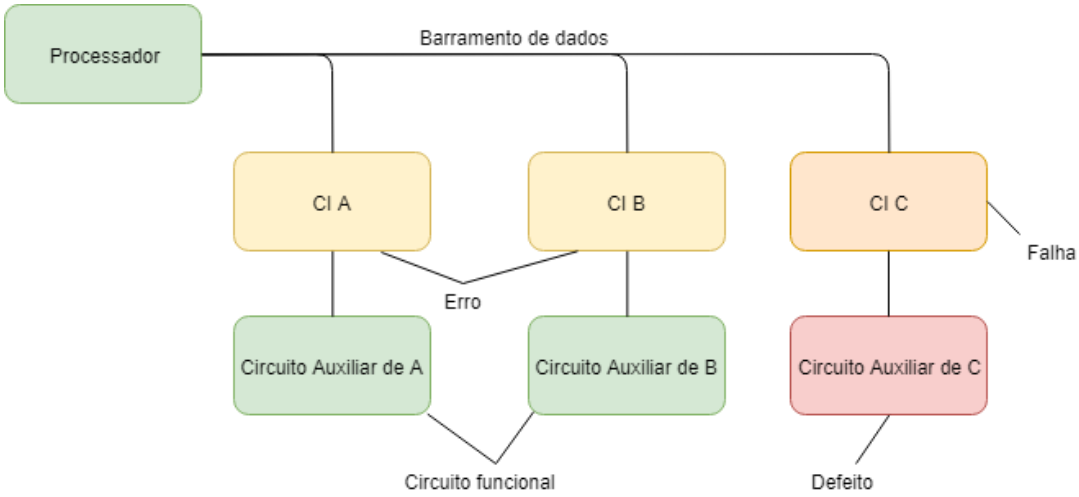
2.4.3 Erros

Para software um erro é a diferença entre o valor recebido e o valor esperado, ou em teoria, o valor correto, resultando em uma resposta incorreta na execução do software. Um erro pode também corresponder a uma ação humana incorreta que produza um resultado incorreto. Um erro de software pode ser exemplificado ao recebermos um arredondamento incorreto em uma aplicação de cálculo, ou ainda a seleção incorreta de casas após a virgula pelo operador da aplicação. (IEEE, 1990)

Em hardware um erro é quando um valor de saída errôneo é produzido devido a um defeito no circuito (circuito com falha). Uma única falha pode ocasionar vários erros, desta forma podemos considerar que ao tratar uma falha diversos erros podem ser corrigidos, e portanto, vários erros podem apontar para a mesma falha. (PRADHAN, 1996)

A Figura 12 apresenta um diagrama onde exemplificando os itens acima explicados.

Figura 12 - Diagrama de Erros, Defeitos e Falhas



Fonte: Adaptado de PRADHAN, (1996)

Pode-se observar como cada tipo de problema se apresenta. Suponha que o CI C tenha interação homem-máquina, porém, está apresentando informações incorretas, o defeito que gerou esta falha está no circuito auxiliar de C, e como o CI C é interligado em um barramento, tudo o que está presente neste esta plausível a erros de sistema, pois, o processador pode perder sincronismo ou dados no barramento (PRADHAN, 1996).

2.5 SISTEMAS DE TESTE

Este tópico abordará alguns conceitos técnicos sobre tipos de testes, modelos de coleta de informação e execução de tarefas (SILVA, 2007).

2.5.1 Black-box, grey-box e white-box test

Os modelos de testes denominados *Black-box*, *grey-box* e *white-box*, foram criados para demonstrar o grau de interatividade que o testador, seja ele humano ou máquina, possui com o equipamento em teste (SILVA, 2007).

- *Black-box:* conforme o nome sugere o teste é uma caixa preta, o testador não tem conhecimento algum sobre o que está acontecendo, não interfere nas etapas do mesmo, apenas inicia a execução e verifica o resultado final. As vantagens desse método são a rapidez, simplicidade e imparcialidade. O testador não precisa estar atento ao teste, podendo executar várias tarefas ao mesmo tempo, o resultado apresentará exclusivamente uma resposta, “Aprovado” ou “Reprovado”. Esse teste possui algumas deficiências, como a superficialidade e a redundância. Caso mal projetado, pode não obter uma boa cobertura ou ainda testar diversas vezes um mesmo circuito (SAUNOIS, 2016).
- *white-box:* De acordo com Saunois (2016) na caixa branca, o testador pode fazer basicamente o que desejar, rodar o teste de traz pra frente, alterar o teste de forma a apresentar o mesmo resultado final ou ainda automatizar processos repetitivos. Dentre seus benefícios, pode-se citar, a antecipação de problemas, a otimização da cobertura e a volatilidade. Nesse método o testador consegue resolver problemas do equipamento durante a execução do teste.
- *grey-box:* na caixa cinza, o testador dispõe de certa interatividade com o teste, por exemplo, alteração no cenário de teste durante a execução, podendo interferir em etapas controladas com opções restritas. Esse modelo é a automação de partes separadas do w*hite box*. Suas vantagens são: a rapidez em testar o que já está evoluído e a volatilidade em testar o que há de novo ou sem cobertura automatizada (SAUNOIS, 2016).

Durante o processo de automatização de testes de hardware, a tendência é a caixa “escurecer” com a evolução do mesmo, até o ponto onde a interação é mínima e a automação máxima (SILVA, 2007).

2.5.2 Tipos de testes

Segundo Crandall (1997) existem diversos testes que podem ser aplicados em placas eletrônicas, sendo os mais conhecidos e aplicados:

- Teste Funcional: A construção de uma Jiga de teste, consiste na montagem de um cenário próximo ao real em que o equipamento será utilizado. O teste funcional é capaz de demostrar se o equipamento dispõe de falhas graves, mas pode ser insuficiente para validação de desempenho. Suas principais características são o baixo custo e o fácil

acesso aos equipamentos necessário para confecção da Jiga de testes (CRANDALL, 1997).

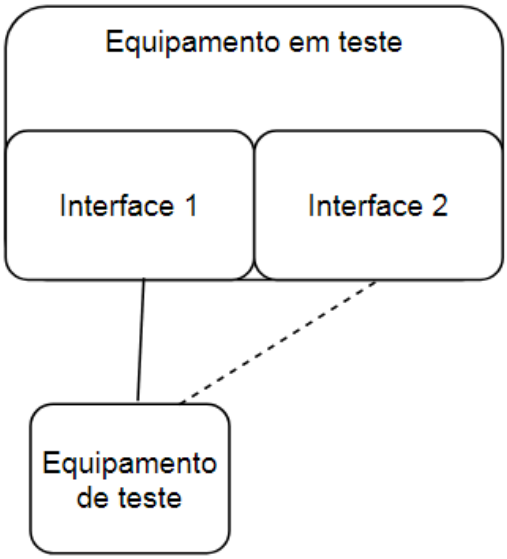
- Teste Analógico: São utilizados normalmente em circuitos simples, sem inteligência embarcada, empregados sobretudo para validar fontes de tensão internas e externas, avaliando se o nível de tensão do ponto de teste está dentro da tolerância especificada (CRANDALL, 1997).
- Teste Digital: Sua principal característica é avaliar se os circuitos integrados de uma placa estão se comunicando conforme o projetado. São testes sofisticados que visam analisar se o produto pode ou não ter queda de desempenho devido a pequenas alterações em barramentos de alta velocidade (NATIONAL INSTRUMENTS, 2018).

Os testes funcionais, analógicos e digitais se completam. Visando-se a melhor cobertura de testes, é necessária sua utilização em conjunto.

2.5.2.1 Teste funcional

Crandall (1997) afirma que esse teste tem complexidade escalável com a quantidade e diversidade de conexões presentes no equipamento. Para cada interface presente, deve-se realizar um teste funcional, conforme pode-se observar na Figura 13.

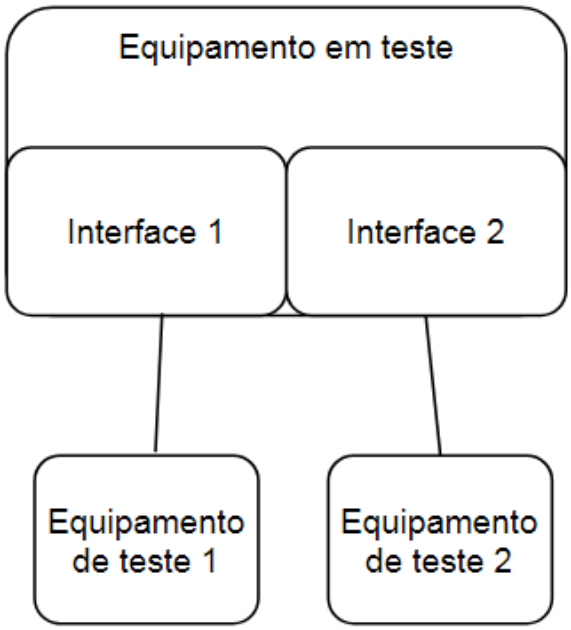
Figura 13 - Diagrama de teste funcional



Fonte: Adaptado de Crandall, (1997)

Quando um equipamento possui diversas interfaces similares, existe a possibilidade de paralelização de processos, ou seja, realizá-los em diversas interfaces ao mesmo tempo, replicando o cenário externo presente no teste conforme Figura 14. (BERNARDO, 2011)

Figura 14 - Teste funcional com paralelização



Fonte: Adaptado de Bernardo, (2011)

Conforme Bernardo (2011) quando paralelizado um teste tem por característica a redução do tempo gasto até obter o resultado, porém, algumas paralelizações, como a da Figura 14, podem acarretar um acréscimo considerável no custo do teste. Nesse caso é pertinente a realização de cálculos para avaliar se a redução do tempo de teste é viável economicamente com o método da paralelização.

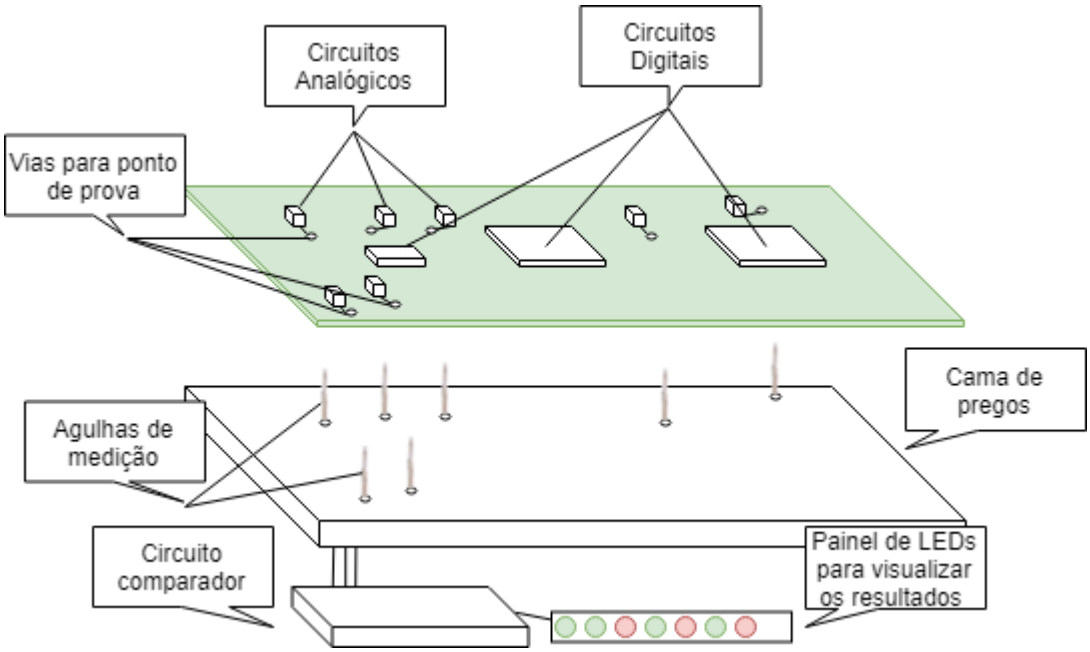
Crandall (1997) também expõe as vantagens na utilização de testes de estresse para produtos, conhecido como teste em *burn-in*. Esse teste consiste em deixar um equipamento por um longo período em teste funcional, variando a temperatura ambiente. Sua realização provoca o envelhecimento da placa, antecipando a ocorrência de defeitos. Desta maneira, problemas relacionados a solda podem manifestar-se em horas de teste. Prevenindo que um defeito não perceptível em um teste funcional comum passe despercebido.

2.5.2.2 Testes analógicos

Testes analógicos visam medir e comparar um padrão de níveis de tensão contínua em determinado circuito. Eles são simples e possuem uma eletrônica básica, em alguns casos esses testes podem ser manuais (GENRAD, 1984).

A automatização dos testes analógicos, necessitam que a placa a ser testada esteja preparada para recebe-los com a adição de pontos de teste, para que um dispositivo com ponteiras de prova, conhecido vulgarmente como “cama de pregos”, realize as medições e, através de circuitos comparadores de tensão, seja possível analisar se ela está ou não dentro da tolerância. A cama de pregos é uma peça mecânica que deve ser precisa, assim necessita de manutenções preventivas e corretivas, o que faz com que o equipamento se torne caro e pouco utilizado em produtos de baixo volume, como pode-se observar pela Figura 15. (JUKNA; JIN, 2018)

Figura 15 - Teste analógico



Fonte: O Autor, (2018)

Com a placa pressionando as ponteiras de medição o circuito comparador é ativado, analisando o resultado obtido e o valor configurado no mesmo. O resultado é geralmente demonstrado em um painel de LEDs indicando o sucesso ou a falha do teste, como pode-se observar pela Figura 15 (GENRAD, 1984).

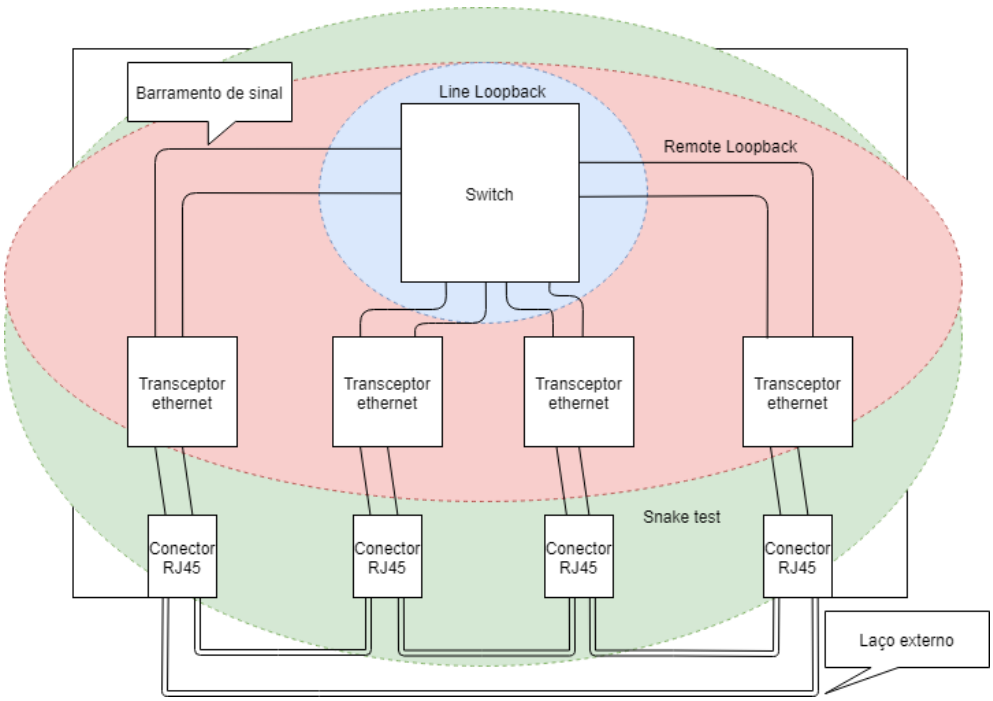
2.5.2.3 Teste digital

Testes digitais visam testar os circuitos digitais presentes em uma placa, ou seja, testam os circuitos com inteligência embarcada como processadores, memórias e CIs (circuitos integrados) específicos. Possuem duas formas de ser realizados, por um controlador externo ou por alguma lógica embarcada no CI. (LARSSON, 2005)

Os testes digitais embarcados possuem grande valor, pois, complementam os testes funcionais quando necessário encontrar a origem de um defeito. Esses testes têm a capacidade de testar a comunicação de alguns barramentos entre os CIs da placa, ou ainda, executar um teste dentro do próprio CI. O teste interno em um CI é denominado *Line Loopback* (laço de retorno em linha) pela Broadcom, enquanto o teste entre diferentes circuitos integrados através do barramento de sinal é denominado *Remote Loopback* (laço de retorno remoto). Esses testes são embarcados em circuitos comutadores (*switchs*), e controladores específicos, como o controlador GPON. (BROADCOM, 2014)

Há também testes embarcados que substituem a necessidade de um circuito externo complexo por uma conexão simples. A família de *switchs* Katana da Broadcom por exemplo possibilita a execução de um teste realizando um laço externo em suas portas, esse teste é chamado *Snake test* pode-se observar esses testes na Figura 16. (BROADCOM, 2014)

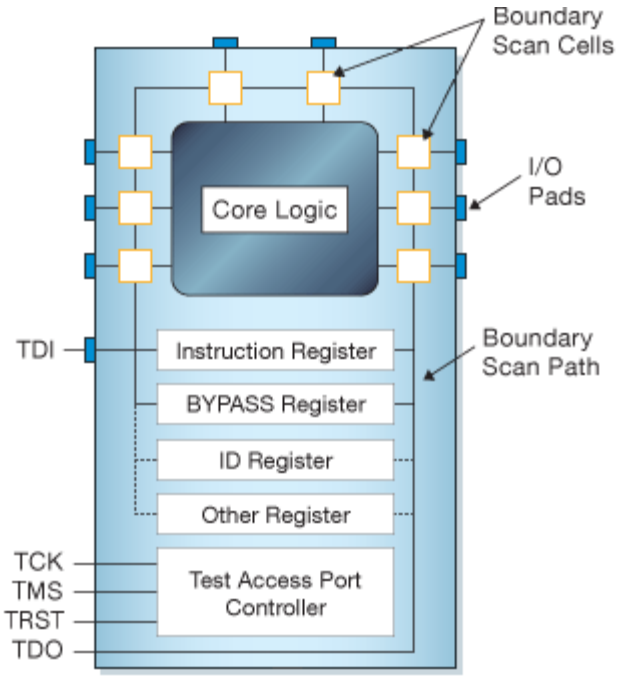
Figura 16 - Abrangência dos testes embarcados em circuitos integrados da Broadcom



Fonte: O Autor, 2018

Além dos testes embarcados, existem os testes que podem ser realizados com o auxílio de um equipamento específico para teste digital, chamados de JTAG. Eles possuem um barramento de comunicação próprio para execução de testes, são empregados com frequência em CIs do tipo BGA (*Ball Grid Array* – Matriz de grade de esferas), onde devido à dificuldade de realizar medições em seus pinos, foi criada essa tecnologia que permite realizar o *Boundary Scan* (scaneamento do sinal nos pinos), que permite habilitar e ler os valores dos pinos sem o acesso físico dos mesmos, a Figura 17 demonstra o diagrama interno de um CI que possui a função de *Boundary scan* (LARSSON, 2005).

Figura 17 - Diagrama esquemático em um CI com JTAG habilitado



Fonte: (XJTAG, 2018)

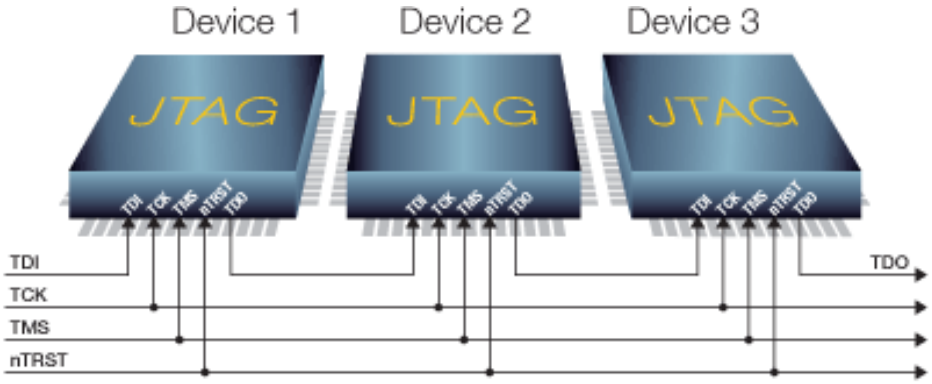
Todos os sinais entre o núcleo do CI e seus pinos são interceptados por um pacote de escaneamento serial, conhecido como *Boundary Scan Register* (BSR), o qual consiste no número da célula de escaneamento. Na operação normal essas células são invisíveis ao CI, no entanto, quando em modo de teste as células podem ser utilizadas para habilitar e/ou ler valores dos pinos ou do núcleo do integrado (LARSSON, 2005).

Segundo a XJTAG (2018) a interface de sinal do JTAG, conhecida como TAP (*test access port* – porta de acesso para teste), utiliza os seguintes sinais para executar o *Boundary Scan*.

- **TCK** (*Test Clock*) – sinal de relógio utilizado para sincronizar as operações da máquina de estados interna.
- **TMS** (*Test Mode Select*) – seleção de modo de teste, sinal amostrado na borda de subida com TCK para determinar o próximo estado do teste.
- **TDI** (*Test Data In*) – Entrada de dados de teste, sinal que representa os dados transferidos para a lógica de teste ou programação do dispositivo. É amostrado na borda de subida do TCK quando a máquina de estado interna está no estado correto.
- **TDO** (*Test Data Out*) – Saída de dados de teste, sinal que representa os dados deslocados da lógica de teste ou programação do dispositivo e é válido na borda descendente do TCK quando a máquina de estados interna está no estado correto.
- **TRST** (*Test Reset*) – reiniciar teste, sinal utilizado para reinicializar o CI após ou durante os testes, esse pino não é obrigatório habilitar no circuito.

Esse teste pode ser realizado em cadeia de até 3 CIs entre os presentes na placa como apresentado na Figura 18, ou seja, a placa deve ser previamente preparada para receber esse teste através de uma conexão, podendo atender todos os circuitos onde o *Boundary Scan* pode ser realizado. (XJTAG, 2018)

Figura 18 - Cadeia de conexão para JTAG



Fonte: (XJTAG, 2018)

Para circuito com mais de 3 CIs presentes na placa, deve-se realizar uma conexão de seleção física, podendo assim habilitar e desabilitar fisicamente em qual CI passará os dados de teste (XJTAG, 2018).

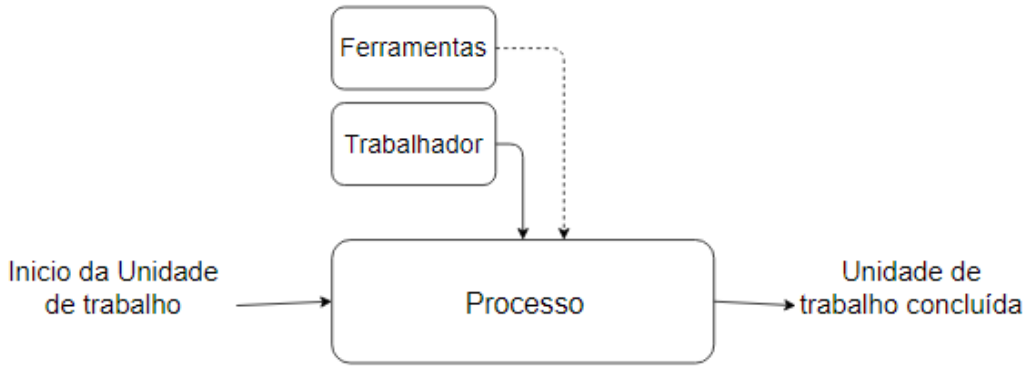
2.6 AUTOMATIZAÇÃO DE TESTES

Os principais motivos para realização de automatizações das etapas de testes em linhas de produção, são: a necessidade de diminuir o tempo gasto com os mesmos e de garantir que o teste será realizado, removendo o fator de erro humano.

Groover (2011) aponta uma série de vantagens em automatizar a manufatura de produtos em fábricas, alguns dos conceitos que ele apresenta, podem ser aplicados de forma direta na automatização de testes. A automação de uma linha de produção reduz os custos com mão de obra e a quantidade de ciclos de produção, ou seja, pode reduzir custos diretos e indiretos, diminuindo a carga de trabalho de um colaborador e retrabalhos nos produtos, além de melhorar a qualidade e a consistência deles. Um dos objetivos da automação é obter uma produção enxuta, realizando um número maior de trabalho com menos recursos, sejam eles humanos ou equipamentos.

Groover (2011) divide os sistemas de produção em 3 categorias, sendo elas, sistema de trabalho manual, sistema trabalhador-máquina e sistema automatizado, observados a seguir nas Figuras 19, 20 e 21:

Figura 19 - Sistema de trabalho manual

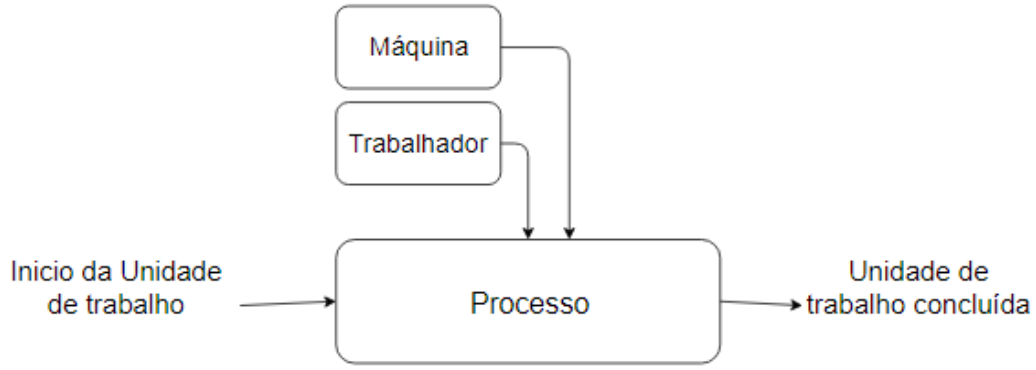


Fonte: Adaptado de Groover, (2011)

Como pode ser observado pela Figura 19, o sistema de trabalho manual pode ser formado por um ou mais funcionários, sem o auxílio de ferramentas motorizadas. Analogamente, para um sistema de testes automatizados, podemos considerar, que um programa que roda uma série de comandos, executando testes em sequência é a ferramenta motorizada. Em equipamentos eletrônicos complexos existem *n* variáveis a serem testadas, a

sequência de comandos para realizar todos os testes podem chegar a centenas, o fator humano torna um teste manual muito suscetível ao erro nessas condições (GROOVER, 2011).

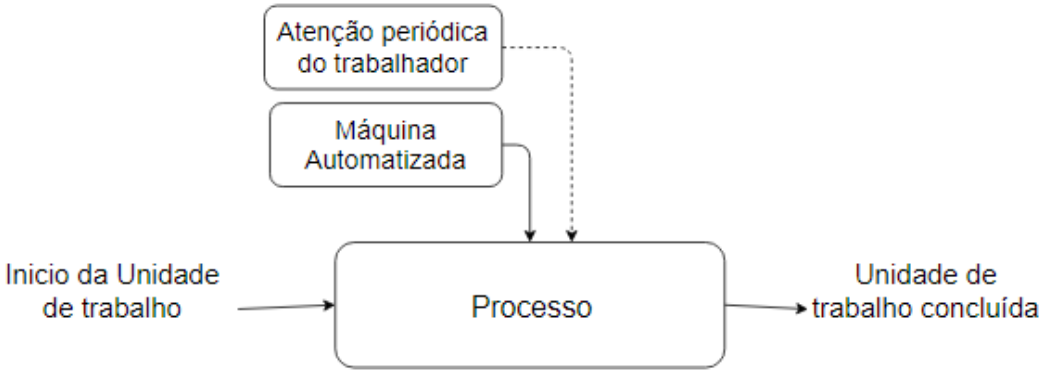
Figura 20 - Sistema trabalhador-máquina



Fonte: Adaptado de Groover, (2011)

Nos sistemas trabalhador-máquina conforme a Figura 20 o trabalhador opera um equipamento motorizado, ou seja, nesse caso, o trabalho não é mais executor da tarefa do processo, apenas manipula a ferramenta que o executa. Para sistemas de teste, pode-se considerar que o motor é nosso programa, sendo assim, o trabalhador não executaria os comandos necessários para realizar os testes, somente realizará a instalação do equipamento na Jiga, e executará o programa que os realiza de forma automática. No final dos testes, analisa o resultado para filtrar equipamentos bons e equipamentos com defeito.

Figura 21 - Sistema automatizado



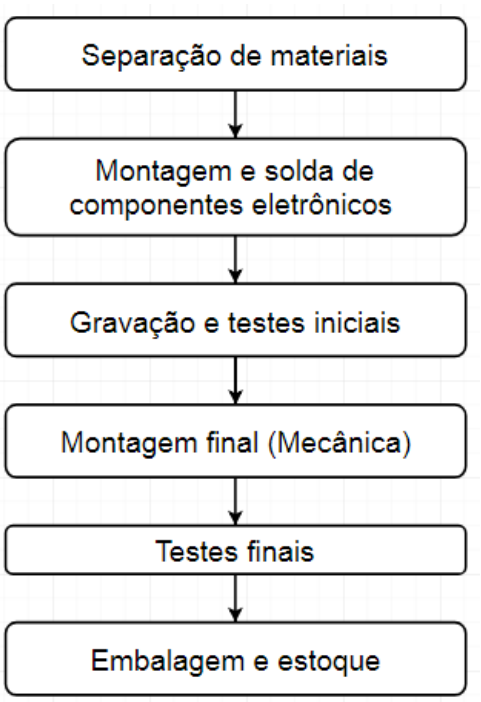
Fonte: Adaptado de Groover, (2011)

Já os sistemas automatizados como os da Figura 21, em que trabalhador não executa tarefas manuais, sua função é verificar se o sistema está executando a instalação do

equipamento na Jiga de testes, se o teste está sendo executado e se a separação entre equipamentos bons e equipamentos defeituosos está correto.

Muitas vezes é difícil dizer se um sistema é automático ou trabalhador-máquina. Para que essa definição esteja correta, deve-se analisar a etapa de manufatura em que o processo está sendo realizado. Por exemplo, a fabricação de uma placa eletrônica possui 6 etapas de manufatura conforme pode ser observado pela Figura 22:

Figura 22 - Diagrama de produção de eletrônicos



Fonte: O Autor, (2018)

A automatização do sistema pode englobar os 6 processos, nesse caso ter-se-ia um sistema sem riscos de falhas humanas, ou ainda, pode-se ter 6 sistemas automatizados, um para cada processo. Nesse caso, se considerando a fábrica como um todo, o sistema é trabalhador-máquina, pois o deslocamento entre os processos é manual, e uma placa pode cair no chão entre os processos e sofrer avarias.

Uma abordagem para a criação de Jigas é fragmentar os processos na realização de tarefas menores, ou seja, os processos de testes podem ser divididos em *n* processos, assim, a Jiga também pode ser fragmentada (GROOVER, 2011).

2.7 SOFTWARE PARA TESTES

Nesse capítulo serão abordadas algumas definições sobre estrutura e desenvolvimento de software para testes. Qualquer equipamento de teste necessita de uma lógica atuando para realização e análise dos resultados, essa lógica pode ser simples, como por exemplo um amplificador operacional verificando se um nível de tensão está de acordo, ou ainda pode ser complexa ao ponto de retornar um relatório com várias páginas de resultado. (SILVA, 2007)

Existem diversos softwares desenvolvidos para testar ou criar uma sequência de comandos e eventos, sejam eles pagos ou de distribuição gratuita. A National Instruments (2018) utiliza o LabVIEW como plataforma para executar os testes em seus equipamentos, outro software muito utilizado é o Robot Framework (2018), uma plataforma de testes gratuita desenvolvida na linguagem de programação Python.

O que difere um software do outro é o nível de abstração que ele oferece para criar a rotina de teste e analisar o resultado. No LabVIEW, com cliques simples é possível criar uma página similar a uma página WEB, com botões para executar o teste e receber o resultado na forma de alertas na tela. Já no Robot Framework, as rotinas são criadas na forma de linha de comando e é criado um arquivo de texto com os resultados no formato de um relatório final (COMUNIDADE ROBOT FRAMEWORK, 2018).

Segundo Jones (2007) softwares são normalmente desenvolvidos em camadas, onde é possível abstrair algumas delas facilmente. Utilizando o Linux presente em um computador pode-se observar as seguintes camadas:

- Interface de chamada do sistema SCI (*system call interface*): ou também conhecida por interface do usuário, é o que pode ser visualizado pelo usuário. A partir dessa interface que se pode interagir com a máquina, abrindo programas ou executando funções.
- Gerenciamento de processos e memória: responsável por executar os processos iniciados pela interface de chamada do sistema e armazenar temporariamente ou permanentemente a resposta do processo,
- Device drivers: software que compõem a primeira camada, executando comandos e lendo respostas diretamente do hardware.

2.7.1 Sistema embarcado

Um sistema embarcado é a combinação do hardware e software presentes em um equipamento. Ao encapsular o software no equipamento ele passa a fazer todo o controle sobre o hardware, executando com a melhor performance possível certa tarefa. Junto a esse software é necessário criar um software de interface homem-máquina, caso ele necessite de interatividade (HINTERMANN, 2007).

Segundo Hintermann (2007) diferente de um computador pessoal, que é feito para executar diversas atividades como, ser um servidor, uma estação de trabalho ou ainda para jogos, um sistema embarcado normalmente é dedicado a executar tarefas restritas, por exemplo um switch não gerenciável, o software embarcado tem a tarefa de comutar os pacotes para a porta correta, sem necessidade de interações. Um computador é composto de diversos sistemas embarcados, como a placa de vídeo, mouse, teclado etc.

2.7.2 *Back-end*

As camadas de *front-end* e *Back-end* estão dentro de uma SCI, onde a interface direta com o usuário é o *front-end*, e todo o restante pode ser considerado como *Back-end*. As principais linguagens de programação utilizadas no *Back-end* são C, C++, Java, Python entre outras que estão surgindo.

Segundo a Comunidade Python (2018) a linguagem Python apresenta uma vasta gama de bibliotecas prontas, que auxiliam na comunicação entre equipamentos, junto a sua fácil interpretação de chamada de funções e respostas o faz uma poderosa ferramenta para desenvolver testes automatizados. A seguir é descrito algumas das principais bibliotecas para obter controle de todo o processo de testes centralizado em uma única plataforma.

- Biblioteca Pexpect: essa biblioteca permite uma fácil interação entre Jiga de teste e equipamento a ser testado. Em poucas linhas de programação, é possível realizar o envio de comandos e o recebimento de eventos, através de diversos meios de comunicação como SSH (*Secure Shell*) utilizados em acessos de internet ou em conexões seriais. Essa biblioteca possui dois principais comandos, o *sendline* – enviar linha – e o *expect* – agurada retorno.

- Biblioteca Subprocess e OS (*operational system*): utilizadas para executar comandos no sistema operacional do próprio equipamento, possibilitando ações como, alterar o estado ou configuração de conexões, ou ainda, realizar a leitura de algum arquivo de texto.

O *Back-end* possibilita também a interação com diversas ferramentas de um sistema operacional, como por exemplo:

- SSH: permite administrar máquinas remotamente, encapsulando outros protocolos. A grande vantagem do SSH sobre outras ferramentas de acesso remoto é a grande ênfase na segurança. Um servidor SSH bem configurado é virtualmente impenetrável. Ele utiliza um conjunto de técnicas de criptografia baseada em chave pública e chave privada para assegurar que apenas as pessoas autorizadas terão acesso ao servidor. Dentro do SSH existem funções especificas para transferência de arquivos como o SCP (*Secury Copy*), que permite especificar o login e endereço do servidor em uma única linha, junto com o arquivo que será transferido. Com isso, ele é muito usado em scripts (MEIRELLES, 2006)
- NFS (*Network file system*): é uma ferramenta de compartilhamento de arquivos, através dela é possível um equipamento utilizar o sistema de arquivos de outro de forma livre. Diferente do SSH o NFS não criptografa a comunicação, tornando a comunicação muito mais rápida, podendo rodar por exemplo um sistema operacional através da rede sem travamentos (MEIRELLES, 2006).
- Docker: É uma ferramenta de virtualização de máquinas, com seu uso é possível multiplicar a execução de uma rotina diversas vezes, possibilitando a paralelização de processos simples, isolados do sistema operacional da máquina que o roda. (SRIVASTAV, 2019)

2.7.3 *Front-end*

Segundo a W3C (2019) o *Front-end* é uma camada considerada fina de software, responsável pelo formato de como as informações serão mostradas ao usuário, nessa camada os softwares são desenvolvidos em linguagens como HTML, CSS, JavaScript entre outras dependendo do nível de interação que o usuário precisa ter com a máquina.

A W3C (2019) descreve o HTML como a linguagem para descrição de estrutura de páginas Web, possibilitando aos programadores meios de:

- Publicar documentos online com títulos, testos, tabelas, listas, fotos etc.
- Recuperar informações online via hyperlinks, com o clique de um botão.
- Projetar formulários para a realização de transações com serviços remotos, para uso na busca de informações, reservas, pedidos de produtos etc.
- Incluir planilhas, videoclipes, clipes de som e outros aplicativos diretamente em seus documentos.

Já o CSS conforme a W3C (2019) é a linguagem que descreve a apresentação das páginas Web, incluído cor, formato e fontes. Ele possibilita a adaptar a apresentação para equipamentos diferentes, como telas grandes ou pequenas ou ainda impressoras. CSS não depende do HTML, a separação deles facilita a manutenção dos sites, o compartilhamento de estilos entre páginas.

Scripts são programas que não necessitam de compilação antes de ser utilizados. Para navegadores *Web*, *scripts* são normalmente codificados na linguagem JavaScript, e executam as funções desejadas na página web, como por exemplo ao clicar um botão executar um comando ou ainda fazer um download, deixando as páginas *Web* mais dinâmicas. Eles permitem a ligação entre o que o usuário está executando e o que será processado na plataforma de *Back-end* da página (W3C, 2019).

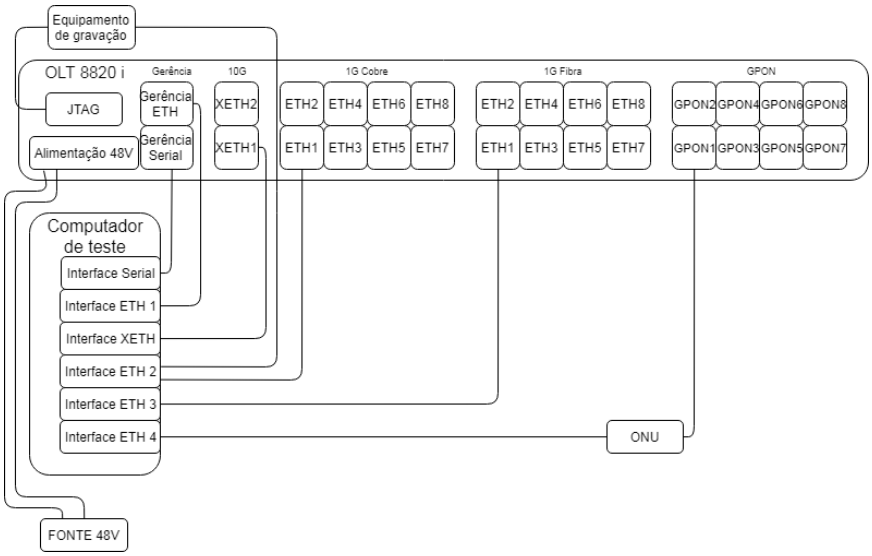
3 DESENVOLVIMENTO DA JIGA DE TESTES

Esse projeto visa desenvolver e apresentar diferentes soluções para implementação de uma Jiga de testes funcionais para OLTs, abordando cenários com alta interatividade entre operador e equipamento, e outros cenários com automatizações, demonstrando os métodos utilizados para efetuar essas automatizações e os ganhos que elas oferecem ao produto de forma geral.

3.1 CENÁRIO ATUAL

A primeira etapa, é o cenário atual sem automatizações. Nele conta-se a OLT8820i, representada com as interfaces GPON, 1G ethernet de “cobre” e de fibra, 10G ethernet, gerências ethernet e serial, e as conexões JTAG e de alimentação. O cenário de teste consiste em um BDI3000 (equipamento JTAG para gravação preestabelecido), um computador com quatro interfaces ethernet, uma interface 10G ethernet e uma interface serial, uma ONU, uma fonte de alimentação 48V, além dos cabos e módulos conversores de sinal óptico para elétrico. O computador utilizado no teste pode ser substituído por computadores separados, para atender a quantidade de interfaces necessárias. Nesses casos pode-se utilizar *single board computers* (computadores de placa única) para atender a necessidade do cenário. A Figura 23 demonstra os equipamentos e suas conexões para execução dos passos de gravação e testes manuais.

Figura 23 - Cenário mínimo necessário para gravação e testes

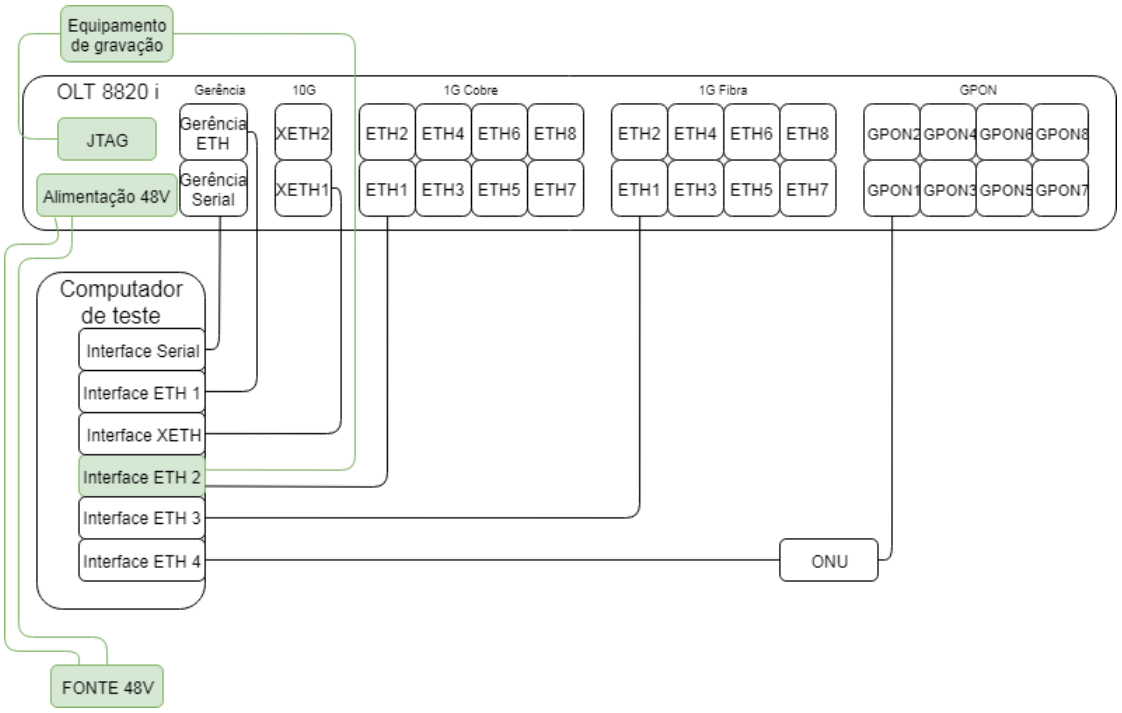


Fonte: O Autor, (2019)

3.1.1 Gravação do u-boot manual

Com base no cenário completo, pode-se destacar as partes atuantes em cada processo. A Figura 24 demonstra o cenário utilizado para execução da gravação manual do u-boot. Nessa etapa são necessários a utilização do BDI3000 conectado à entrada JTAG da OLT, e uma conexão ethernet com o computador. O u-boot é o primeiro software gravado no equipamento, nele constam configurações básicas do hardware:

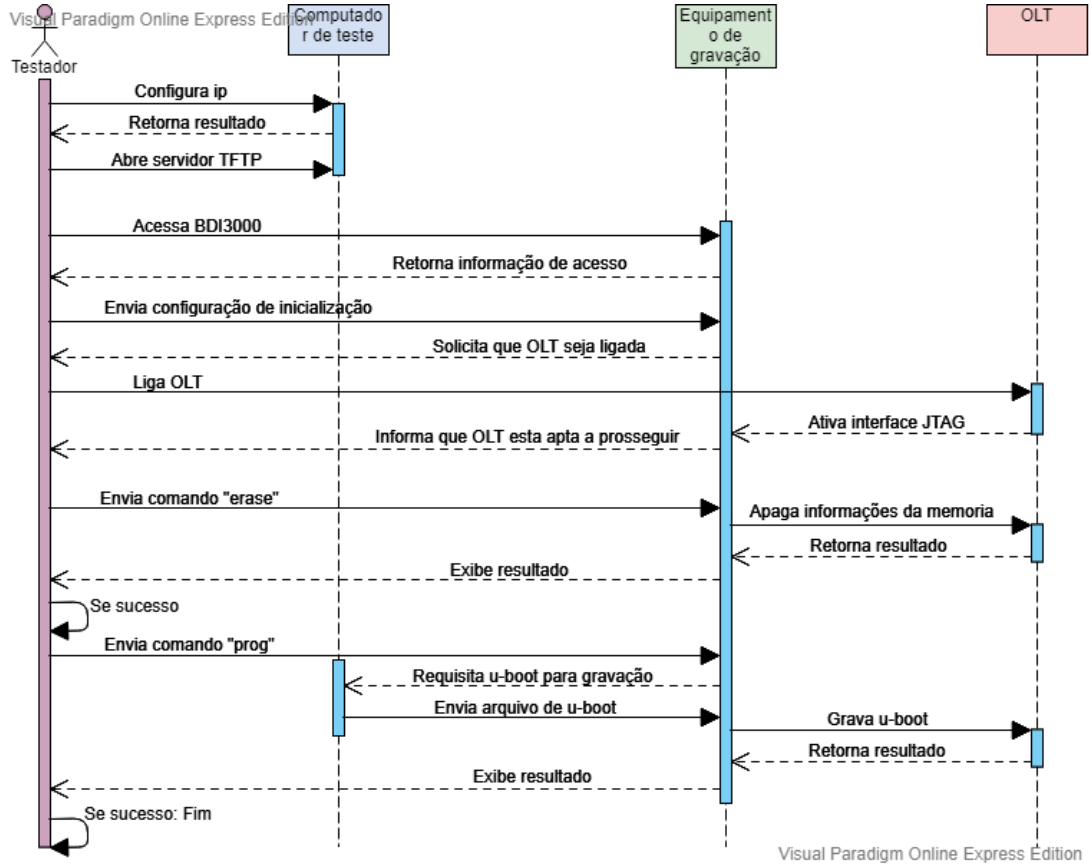
Figura 24 - Cenário de gravação do u-boot



Fonte: O Autor, (2019)

Para a gravação do u-boot o operador deve configurar o computador com IP padrão e iniciar o servidor TFTP, em seguida, conectar o BDI3000 na OLT com a mesma desligada, acessar o BDI3000 e ligar a OLT, ao aparece “OLT8820i>” na tela a comunicação entre os equipamentos foi estabelecida, então, pode-se executar os comando “*erase”* para limpar a memória onde o u-boot será gravado e “*prog”* para gravar a mesma. Pode-se observar sequência de eventos para a etapa de gravação do u-boot na Figura 25.

Figura 25 - Diagrama de sequência da gravação do u-boot



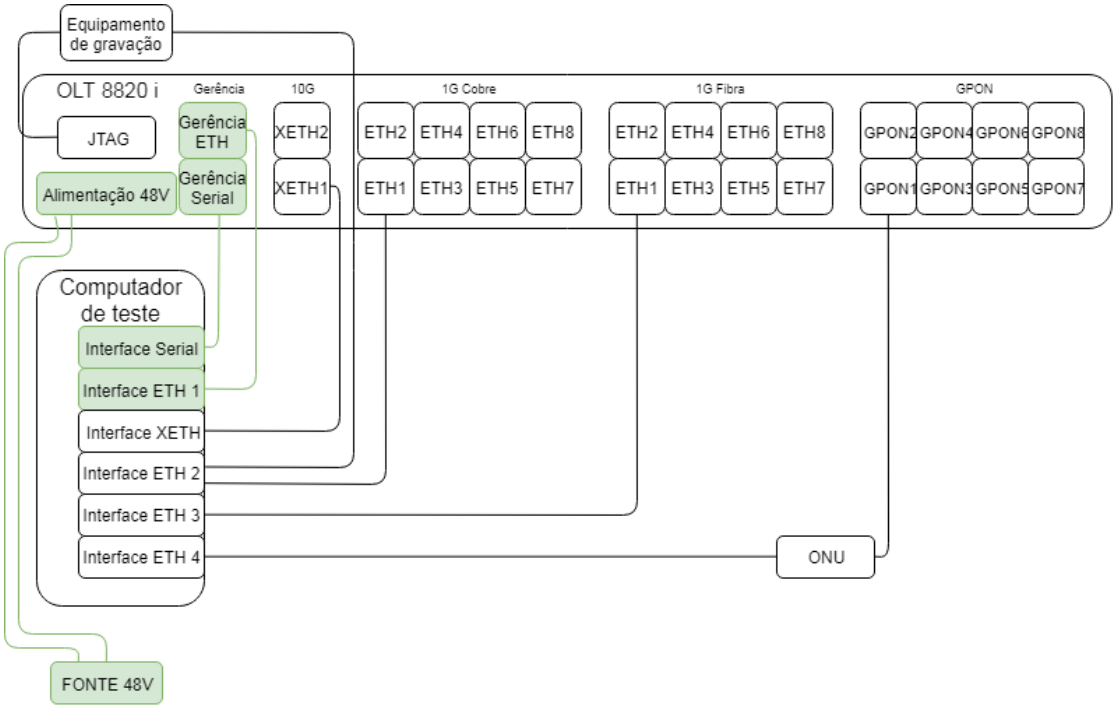
Fonte: O Autor, (2019)

O número de interações necessárias entre o testador e os equipamentos, observado a partir da Figura 25, onde cada flecha que sai da coluna do operador é uma ação, em cada flecha que retorna o operador deve verificar se a execução foi bem-sucedida ou não. Como pode-se observar a gravação possui 13 interações, e o tempo de execução desses passos é em média de 3 minutos.

3.1.2 Gravação do firmware manual

O firmware é todo o software que será embarcado na OLT, nele encontra-se toda a lógica que faz o equipamento possuir as funcionalidades projetadas. O cenário necessário para essa gravação depende da conexão das interfaces de gerência serial e ethernet ao computador, a Figura 26 demostra essas conexões para gravação do firmware.

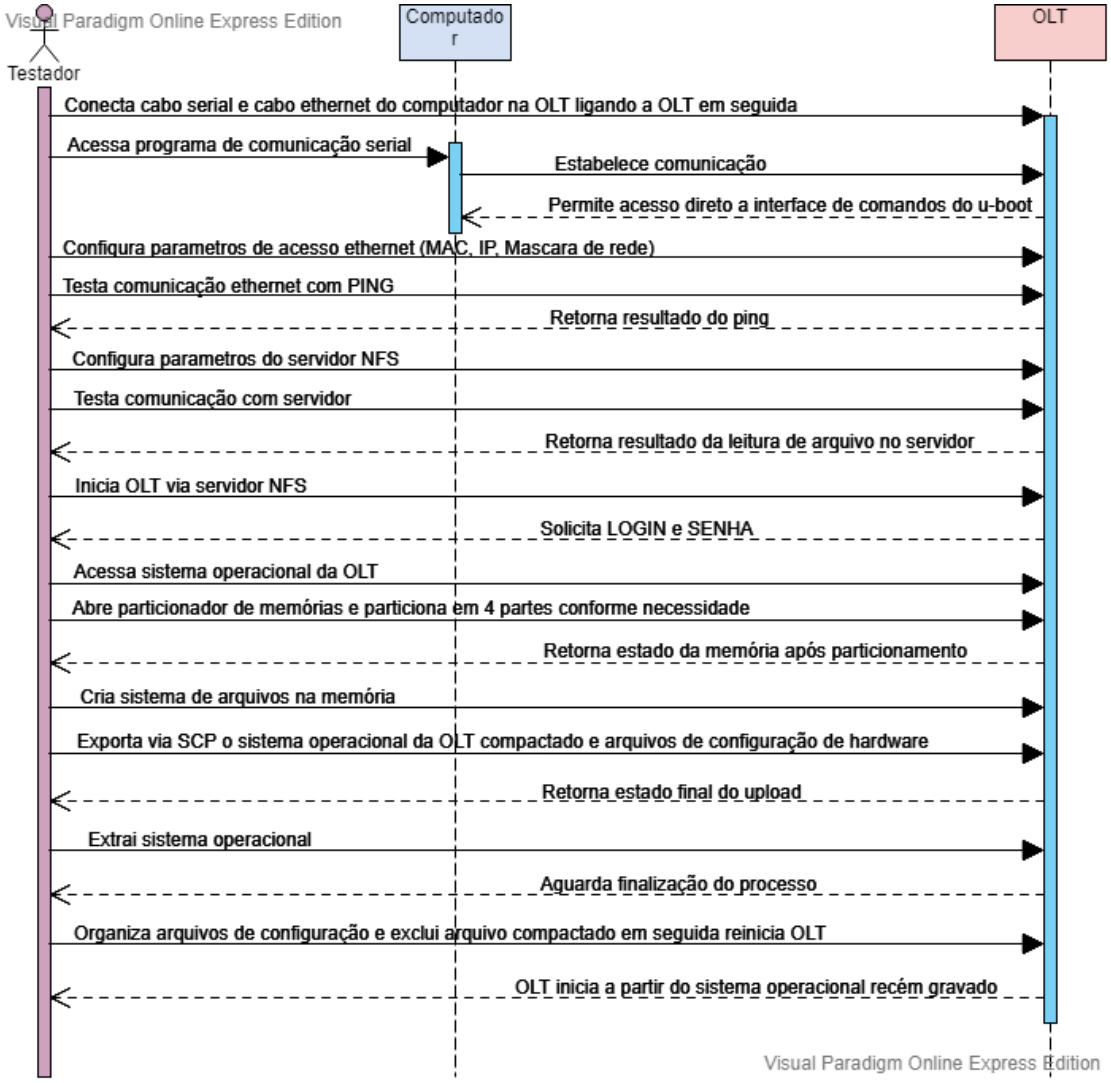
Figura 26 - Cenário de gravação do firmware



Fonte: O Autor, (2019)

A gravação do firmware possui 5 etapas principais, primeiro com a OLT ligada e conexões estabelecidas, deve-se configurar a OLT para acessar o servidor NFS do computador. Uma vez configurada pode-se inicializar o sistema operacional da OLT via NFS, possibilitando a execução de comandos na mesma. Então particiona-se a memória MMC da OLT, criando em seguida, o sistema de arquivos para abrigar o firmware. Feito isso é realizado o upload da imagem compactada e de arquivos de configuração, logo após, é descompactada a imagem e alocado os arquivos em seus respectivos diretórios. Pronto, ao reiniciar a OLT o firmware está embarcado. A Figura 27 demostra o passo a passo detalhado para a execução da etapa de gravação do firmware da OLT.

Figura 27 - Diagrama de sequência de gravação do firmware da OLT



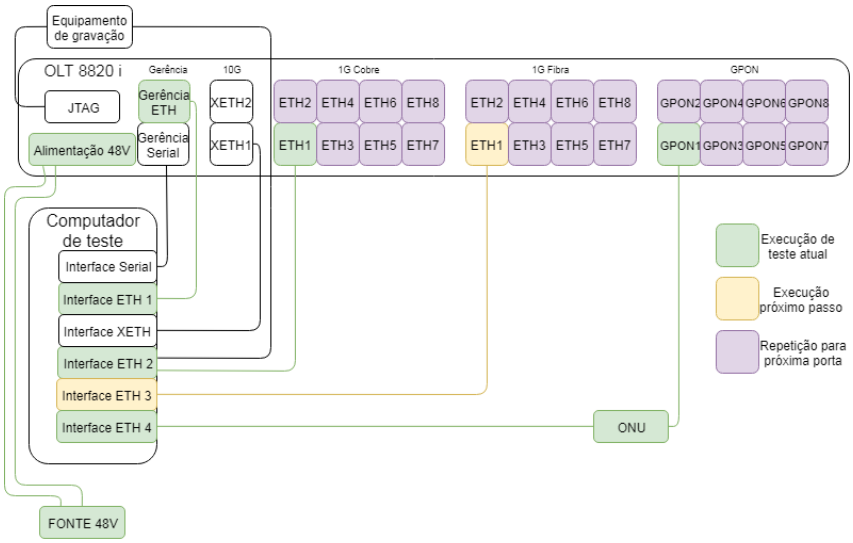
Fonte: O Autor, (2019)

Como pode-se observar pela Figura 27, são necessárias 20 iterações para gravar o firmware do equipamento. O firmware possui tamanho de 150MB compactado, e de 700MB descompactado, gerando um tempo de 15 minutos de download e descompactação, o particionamento e a criação do sistema de arquivos gasta cerca de 1 minuto para o processamento das informações, esses tempos junto ao tempo gasto pelo testador para executar os passos, gera um total de 25 minutos para concluir a gravação do equipamento.

3.1.3 Teste funcional manual

Finalizada a etapa de gravação, é possível iniciar os testes no equipamento. As Figuras 28 a 33 demostram as interações físicas necessárias para testar todas as conexões. Iniciando pela Figura 28, testa-se as portas ETH1 (cobre) e GPON1.

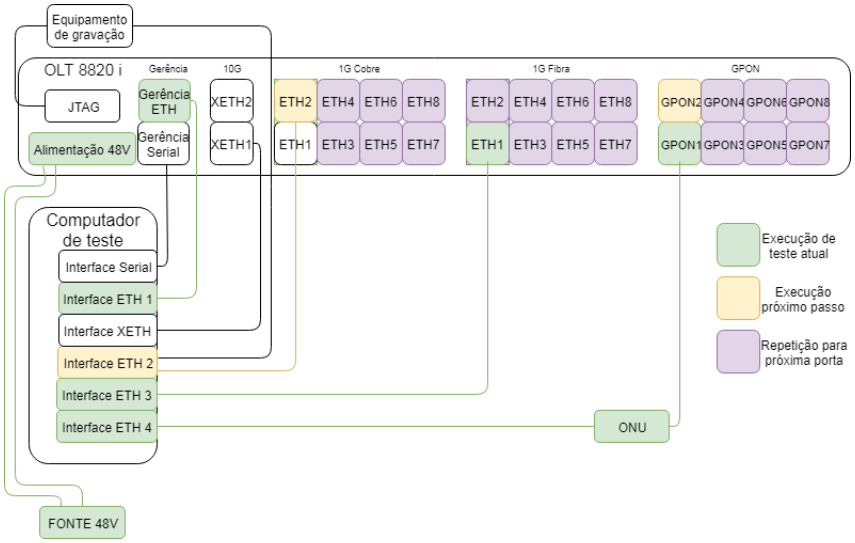
Figura 28 - Etapas de teste da porta ETH1 de cobre e GPON1



Fonte: O Autor, (2019)

Pela Figura 29 pode-se observar a desconexão da porta ETH1 de cobre e a conexão da porta ETH1 de fibra, como as portas são espelhadas não é necessário realizar uma nova configuração, apenas é executado o tráfego de dados de testes, finalizando o teste das portas 1.

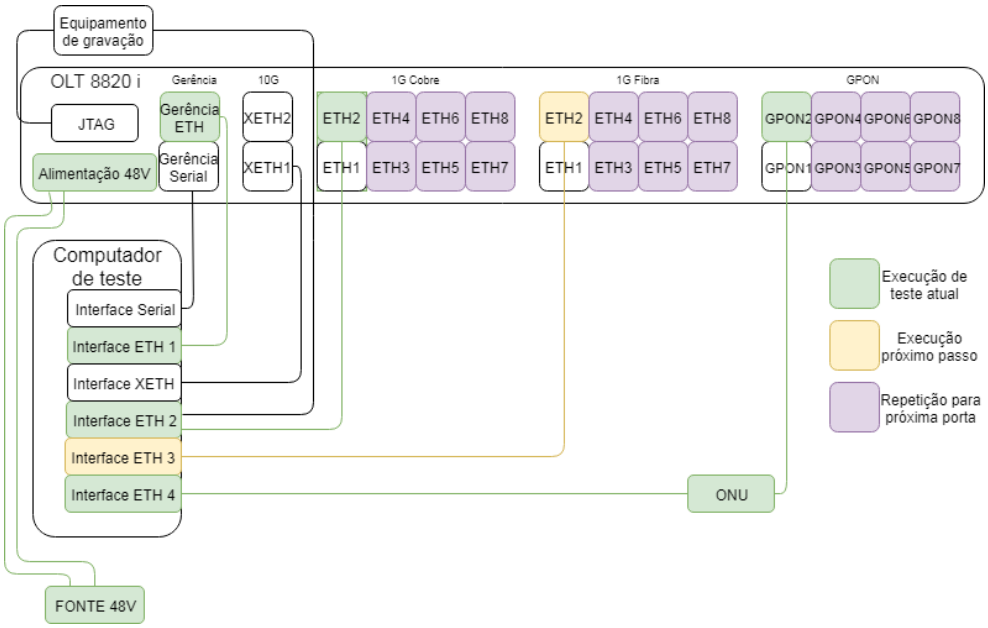
Figura 29 - Etapas de teste da porta ETH1 de fibra e GPON1



Fonte: O Autor, (2019)

Na Figura 30 pode-se observar os testes das portas 2, desacoplando o sinal das portas 1 e conectando a porta ETH2, em seguida é realizada a configuração da porta para rodar o teste.

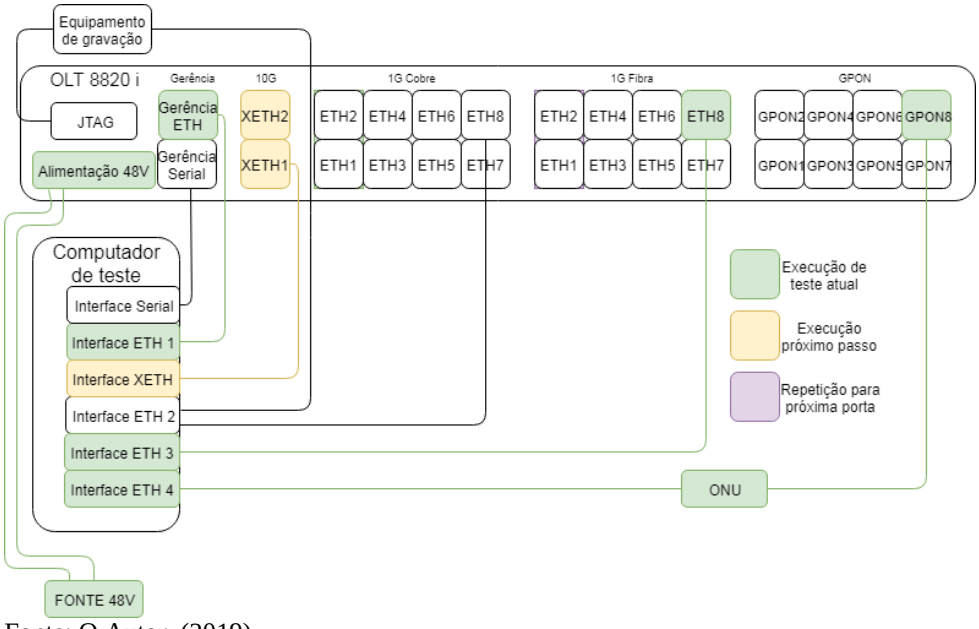
Figura 30 - Etapas de teste da porta ETH2 de cobre e GPON2



Fonte: O Autor, (2019)

Esse processo se repete até chegar nas portas 8, exemplificadas na Figura 31, finalizando o primeiro ciclo de testes, com 24 manobras no cenário, e 8 configurações no equipamento testado.

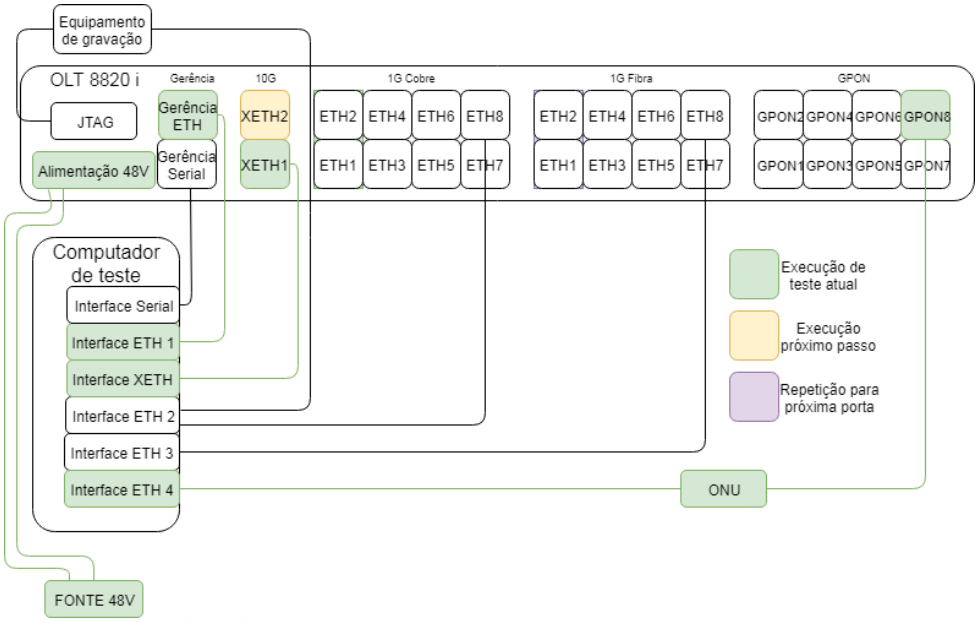
Figura 31 - Etapas de teste da porta ETH8 de fibra e GPON8



Fonte: O Autor, (2019)

A Figura 32 demonstra o início do ciclo de teste das interfaces 10G, onde se aproveita a porta GPON8 para executar o teste na porta XETH1, sendo necessário configurar apenas metade do equipamento.

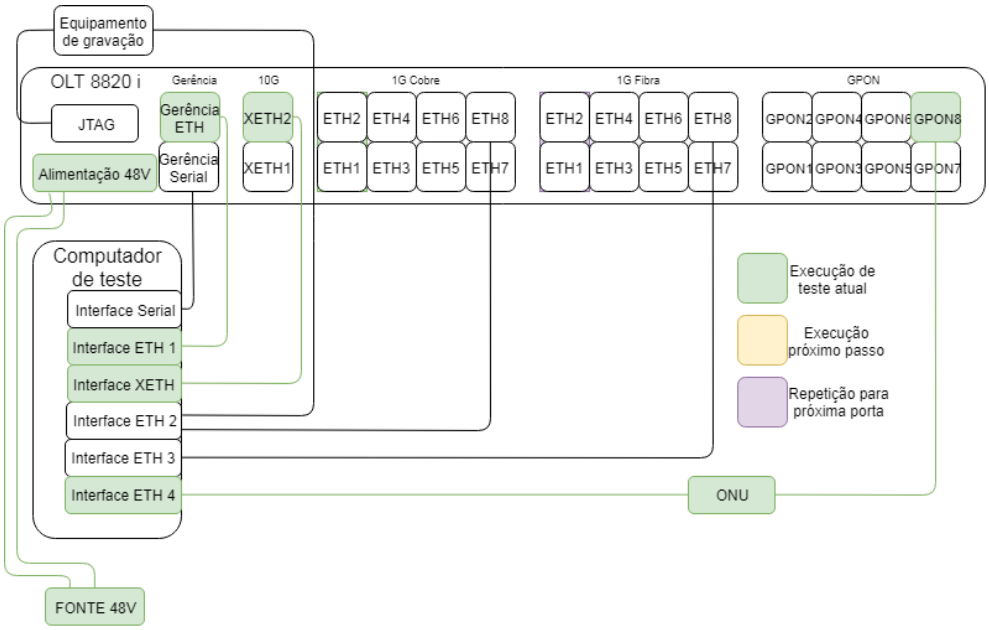
Figura 32 - Etapas de teste da porta XETH1 de fibra e GPON8



Fonte: O Autor, (2019)

E em seguida, a Figura 33 demonstra a finalização desse teste na porta XETH2, removendo a conexão e configuração da porta XETH1 e replicando para a porta XETH2.

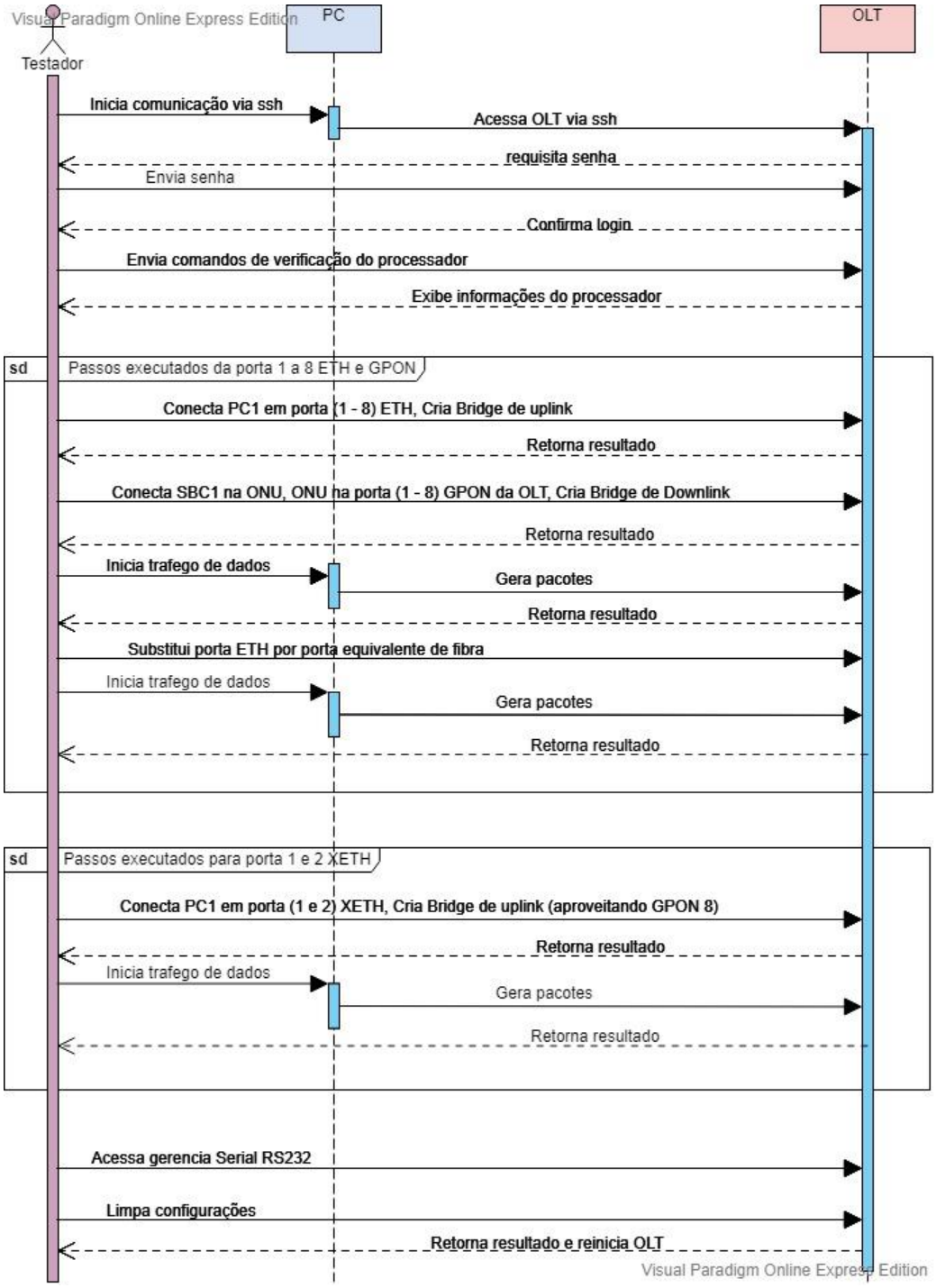
Figura 33 - Etapas de teste da porta XETH1 de fibra e GPON8



Fonte: O Autor, (2019)

São vinte e seis interfaces testadas uma a uma, para cada uma delas, é necessário executar a configuração da OLT, para possibilitar o tráfego de dados de teste. Além dessas interfaces essa etapa contempla a execução de alguns comandos que permitem testar periféricos do processador, como memórias e sensores de temperatura. A Figura 34 expõe a sequência de passos que o testador deve executar para testar as interfaces do equipamento.

Figura 34 - Diagrama de sequência dos testes manuais

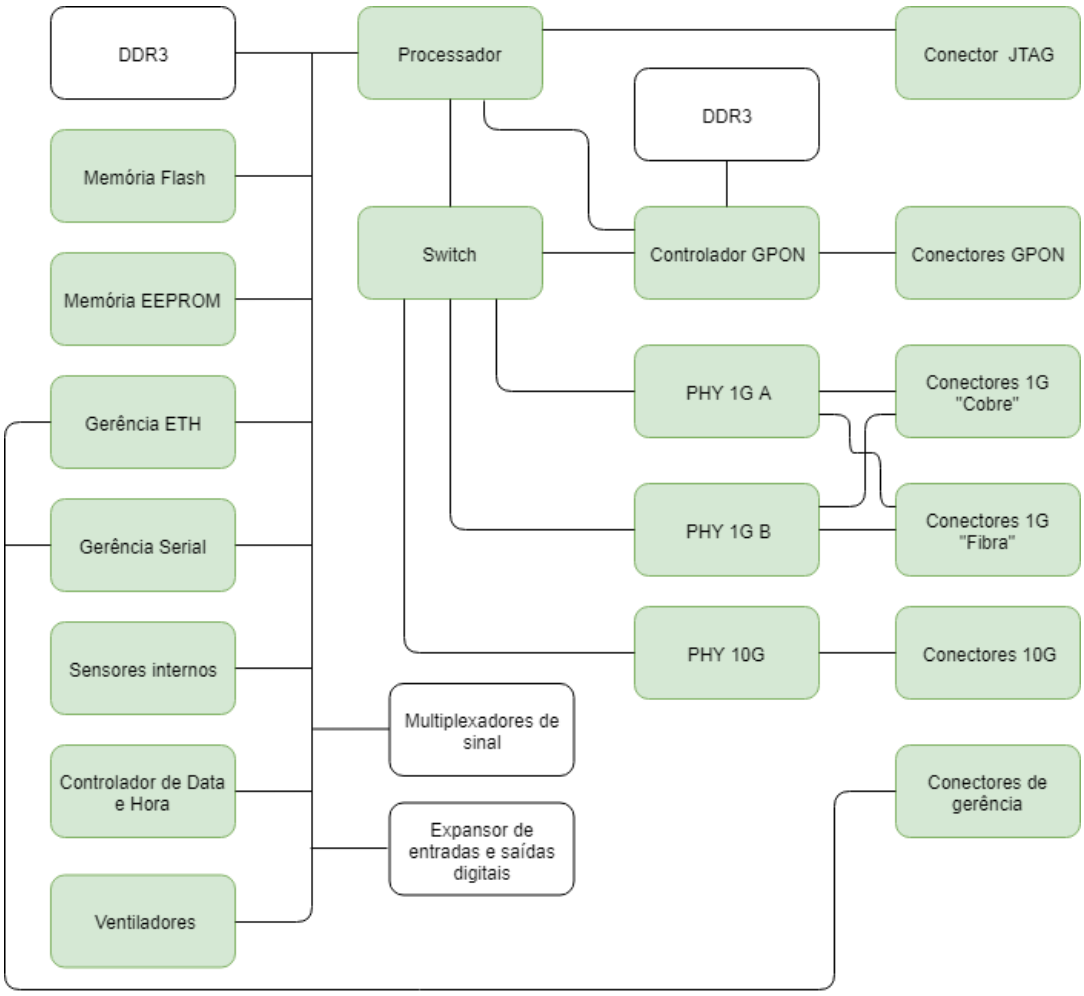


Fonte: O Autor, (2019)

Considerando-se que é necessário aplicar oito vezes os testes entre portas GPON e portas ETH, mais duas vezes o teste GPON com as portas XETH, totaliza-se 89 interações para testar todas as interfaces do equipamento. O tempo para execução desses testes com uma pessoa treinada para executar os passos, fica em torno de 80 minutos, considerando 10 segundos de tráfego de dados por conexão.

Esse teste pode ser considerado *white box*, pois, apesar de haver um documento indicando a sequência de teste, ele pode ser executado em diferentes ordens, desde que se respeite os passos de teste de cada interface. Com ele se consegue testar todas as interfaces do equipamento e alguns componentes internos, a Figura 35 demonstra em verde os circuitos cobertos por esse teste.

Figura 35 - Cobertura do teste funcional



Fonte: O Autor, (2019)

Os circuitos em branco não são comtemplados no teste manual, pois, não há um acesso direto aos mesmos, suas funcionalidades são exclusivas para uso interno. Para testar esses circuitos, é necessário obter acesso a outro perfil de usuário, conforme será exposto no item 3.3.

3.2 GRAVAÇÃO E TESTES AUTOMATIZADOS.

O processo de automatização abordado nessa etapa, visa diminuir o tempo gasto nos processos de gravação e testes, sem alterar a qualidade deles, dessa forma, o desenvolvimento do software de teste visa simular as interações que o operador do teste tem com o equipamento, mantendo o cenário próximo ao utilizado nas etapas manuais.

Nos testes manuais o operador iniciava uma sessão de comunicação entre o computador e a OLT, ou seja, o computador era apenas uma ferramenta em um processo manual de trabalho. Ao simularmos computacionalmente o operador atuando sobre o equipamento, se passa a ter um sistema de teste onde o processo é trabalhador-máquina, ou seja, o operador não executará os comandos para o equipamento, mas sim comandos no computador, as interações restringem-se a parte física, como efetuar conexões e ligar a OLT.

Para fazer o computador se comunicar com a OLT será utilizada a linguagem de programação Python, junto a biblioteca Pexpect. Essa biblioteca permite invocar o programa que inicia a comunicação serial com a OLT, a partir o comando:

```
1-  child = pexpect.spawn('minicom -D /dev/ttyS0 -b 115200')
```

Desta forma, o programa "Minicom" utilizado para comunicação serial fica sob o poder do Pexpect atribuído a variável *child*. A partir desse ponto, basta utilizar os métodos *sendline* para enviar linhas de comando, e *expect* para aguardar uma resposta configurada. Para manter o programa sob controle, é necessário aplicar pontos de parada conhecidos, caso não feita essa boa prática, alguns comandos podem ser executados mais rápidos que o processamento da OLT, sem o resultado da última ação ter finalizado, causando a perda da sequência lógica necessária. A sequência a seguir demostra um exemplo de como utilizar essa biblioteca.

```
2-  child.sendline('erase')
3-  child.expect('Erase passed successfully')
4-  child.expect('OLT8820i>')
```

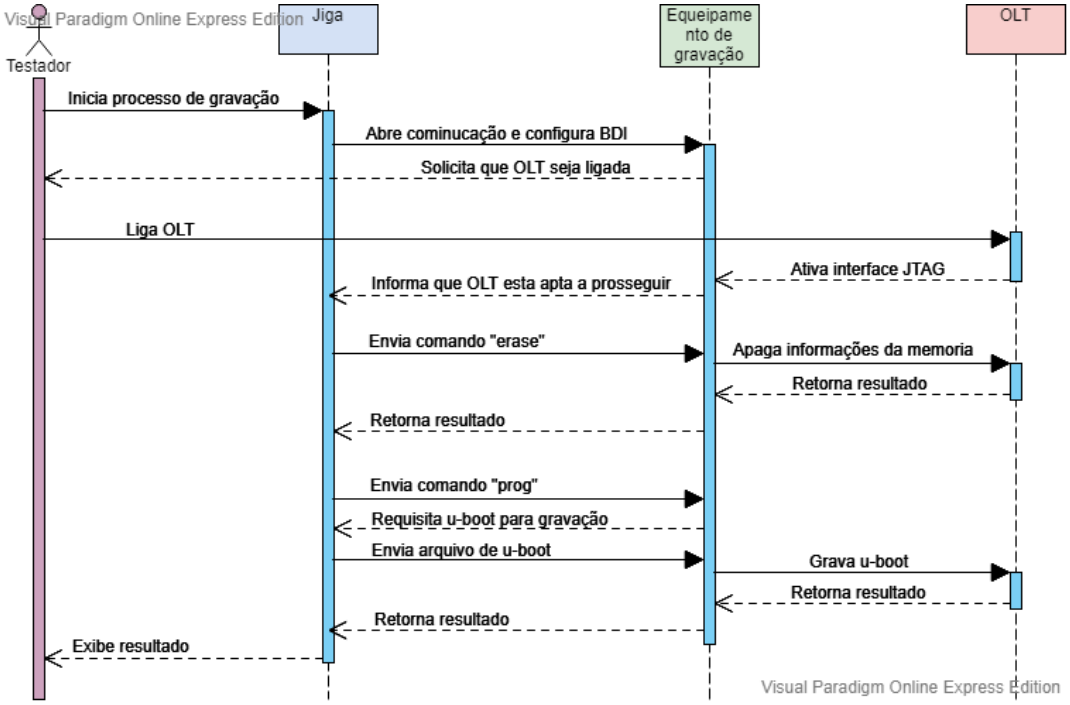
A sequência acima simula a comunicação entre o Pexpect e a OLT, para limpar a memória antes da gravação do u-boot. O comando *"erase"*, executa a função que limpa a memória, em seguida, aguarda o retorno da frase *"Erase passed successfully"*, confirmando

que a memória foi apagada com sucesso, e por último aguarda o ponto conhecido "*OLT8820i>"*, continuando a sequência de comandos apenas após esse evento.

3.2.1 Gravação do u-boot automatizado

Visto como funciona essa comunicação, ao automatizar a etapa de gravação do u-boot ele permanece com o cenário físico da Figura 24, porém, o diagrama de sequência é alterado. O operador interage apenas com o computador da Jiga, executando o início da gravação e as conexões necessárias. Em seguida, a Jiga se comunica com o equipamento simulando os passos que o operador executaria no processo manual, como pode-se observar pela Figura 36.

Figura 36 - Diagrama de sequência da gravação do u-boot automatizado



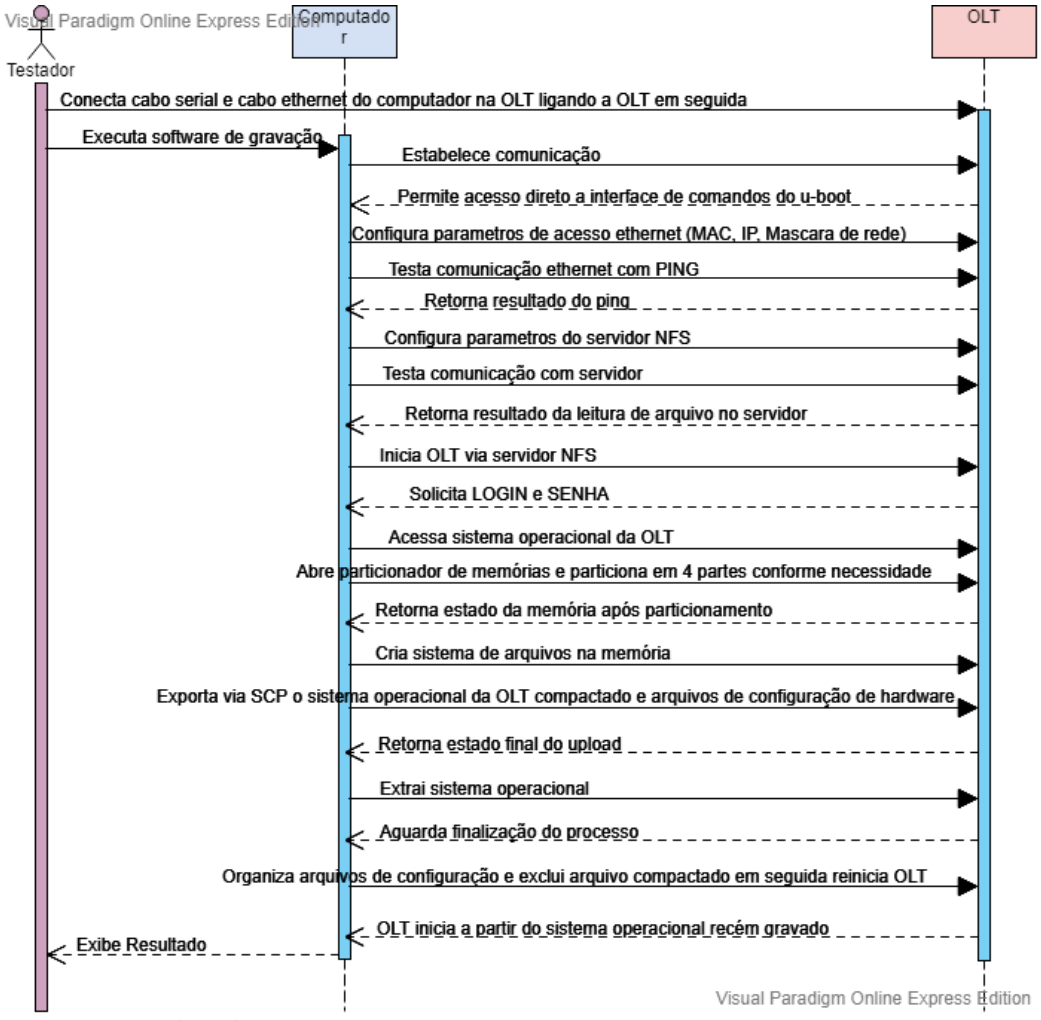
Fonte: O Autor, (2019)

Como apresentado na Figura 36, o operador executa apenas 4 interações, enquanto na gravação manual do u-boot, da Figura 25, precisava de 13 interações. O tempo de gravação dessa etapa não foi alterado significativamente, pois as interações eram de baixa complexidade, após a automatização a média de tempo gasto permaneceu em 3 minutos.

3.2.2 Gravação do firmware automatizado

Esse processo assim como o descrito na etapa 3.2.1, permanece com as conexões físicas da gravação manual, sendo automatizado pela utilização do Pexpect, apenas as interações de envio de comandos. O operador apenas inicia o processo de gravação no computador da Jiga, e ela executa todos os passos que antes eram feitos manualmente, a Figura 37 demonstra a sequência de eventos da gravação automatizada.

Figura 37 - Diagrama de sequência de gravação automatizado



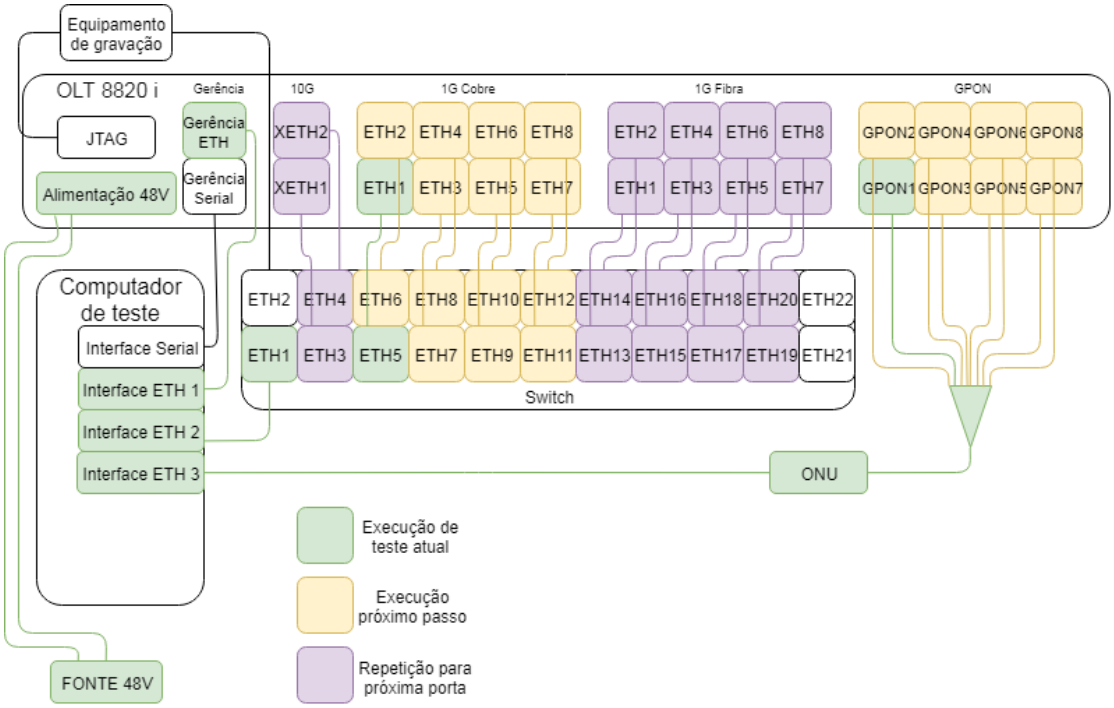
Fonte: O Autor, (2019)

Nessa etapa a redução de interações foi de 20 para 3, um ganho que se justifica ainda mais devido à complexidade existente nesse processo. Diferente da gravação do u-boot, em que os comandos se resumiam a um *"erase"* e um *"prog"*, nesta etapa havia linhas de comando com mais de 80 letras. A redução no tempo de gravação também foi um ganho, sendo reduzido de 25 minutos do sistema manual para 17 no sistema automatizado, 8 minutos ou 32% a menos.

3.2.3 Teste funcional automatizado

Diferente dos itens abordados até o momento, nessa etapa, o teste automatizado sofreu alterações no cenário, diminuindo as interações físicas, que eram elevadas, otimizando a automação. Nesse processo introduziu-se um *switch* e um *splitter* no cenário, permitindo conectar todas as portas em 3 momentos distintos, como pode-se observar na Figura 38.

Figura 38 - Cenário de teste funcional automatizado



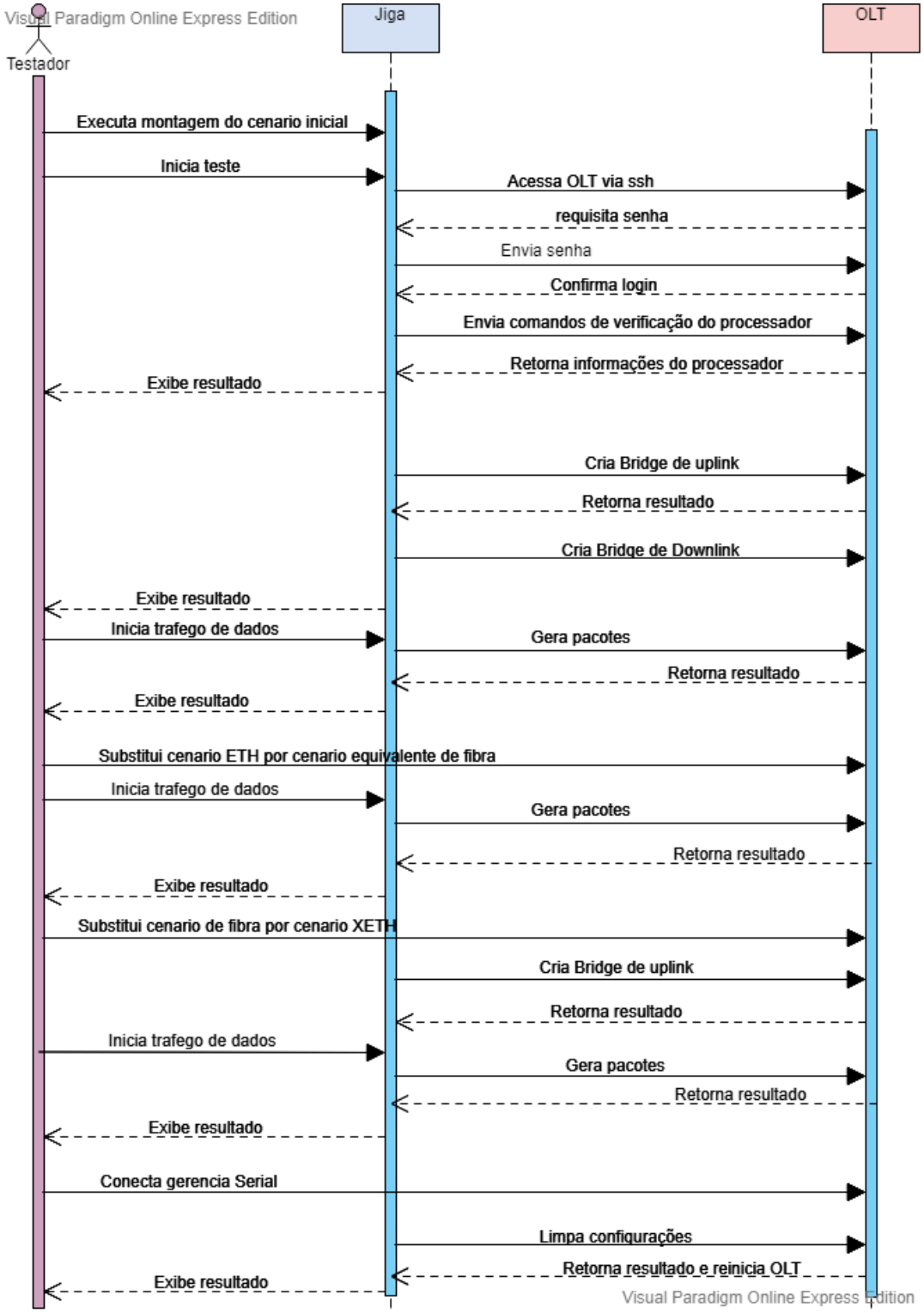
Fonte: O Autor, (2019)

Esse cenário viabiliza-se a partir de uma função presente na OLT, que desliga as portas GPON que não são desejadas, assim pode-se utilizar um *splitter* invertido, saindo uma fibra da ONU e entrando oito na OLT. No início do teste, todas as portas GPON são desligadas, e o teste liga uma a uma conforme a necessidade. Na parte ETH são realizadas 3 sequências de conexões, onde, na primeira conecta-se as oito portas de "cobre", na sequência as 8 de fibra, e por último as XETH. No momento dessas trocas de cenário, o software de teste fica aguardando um "ENTER" do operador para seguir a próxima etapa corretamente. Observa-se também, que a adição do *switch*, permite a redução de duas interfaces do computador de teste.

As interações para executar o teste automatizado, são mais simples comparadas as do teste manual, o testador monta o primeiro cenário e executa o teste, aguarda as primeiras respostas enquanto a Jiga executa os comandos na OLT, em seguida, é alterado o cenário e

iniciada uma nova etapa. Após o resultado dessa etapa, inicia-se a última etapa de testes, finalizando-a quando a Jiga retornar o resultado. Por fim com a conexão serial, são limpadas as configurações de teste, a Figura 39 demonstra a sequência de eventos desse sistema.

Figura 39 - Diagrama de sequências teste funcional automatizado



Fonte: O Autor, (2019)

Comparando a Figura 39 com a Figura 34, há uma queda de 89 para 14 interações, reduzindo drasticamente o tempo de teste de 80 para 28 minutos, ou seja, 65% mais rápido, mantendo a cobertura de testes. Esse teste altera o sistema de w*hite box*, para *gray box*, uma vez que a Jiga controla os passos eles não podem ter a ordem alterada, mas o operador ainda tem interações na execução do teste.

3.3 TESTE EMBARCADO

Todo sistema embarcado possui vantagens como:

- Acesso as camadas de software mais próximas do hardware;
- Melhor desempenho em funções básicas;
- Quebra de regras de negócio aplicadas a usuários finais do produto.

Ao embarcar testes no produto, pode-se ter acesso a todos esses benefícios, a OLT Intelbras limita as operações e acessos do usuário para a aplicação do produto, ou seja, concentrar *uplinks* de *backbones* e transmitir sinal para a última milha das conexões de internet, para isso o usuário tem acesso aos comandos presentes no perfil administrador do produto. Os testes funcionais dos itens 3.1 e 3.2 faziam uso desse perfil para executar os testes.

A primeira barreira quebrada no teste embarcado é ter acesso ao *root*, esse perfil permite maiores liberdades dentro de um sistema baseado em Linux, com ele é possível trabalhar em qualquer camada de software, para desenvolvimento de testes, desde os drivers, até criar um *front-end* exclusivo para teste, ou ainda, executar rotinas de teste repetidamente.

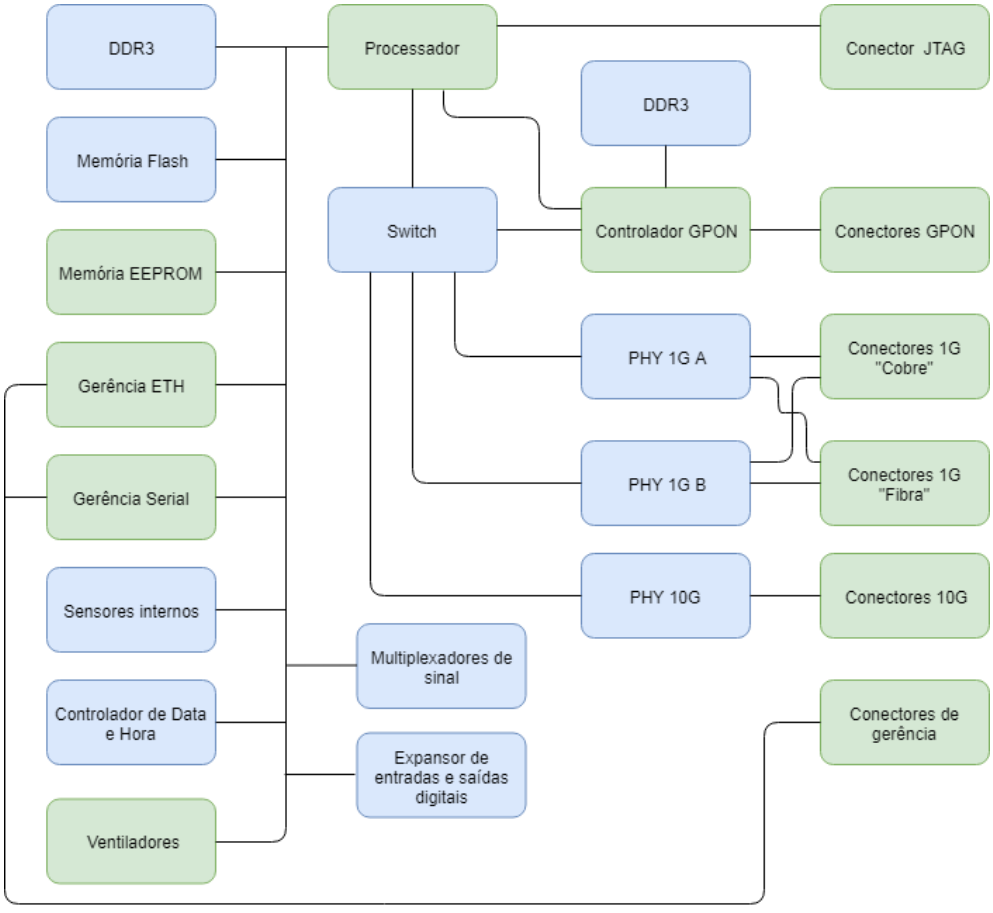
Só o acesso a esse perfil, já poderia melhorar algumas aplicações em testes desenvolvidos anteriormente, porém, seu uso permite acessar os drivers de cada CI conectado ao processador da placa, ou seja, o software criado para trabalhar com o perfil administrador, sofreria mudanças gigantescas, sendo melhor iniciar a construção de um novo software, aproveitando o que estava bom no anterior e aplicando as melhorias que esse acesso permite. Além de executá-lo diretamente no equipamento a ser testado, o que permite, por exemplo, o desenvolvimento do item 3.4 (teste de *burn-in*) como será apresentado mais adiante.

Conforme demonstrado na Figura 35, alguns circuitos não eram testados nos testes manuais e automatizado, devido à falta de acesso aos mesmos. Com o desenvolvimento de um teste embarcado e acesso ao perfil *root*, é possível elaborar testes para as memórias DDR, para os multiplexadores, e para o expansor de entradas e saídas de sinal. Além disso, é possível

aproveitar os testes disponibilizados pelos fabricantes dos Cis presentes na placa, como o *remote loopback*, *local loopback* e *snake test*, apresentados na Figura 16, e ainda aplicar melhorias pontuais em outros testes, como por exemplo ler o barramento de comunicação de alguns CIs.

A expansão da cobertura de testes pode ser observada pela Figura 40, onde as caixas azuis são testes que foram criados ou melhorados.

Figura 40 - Circuitos cobertos por testes embarcados

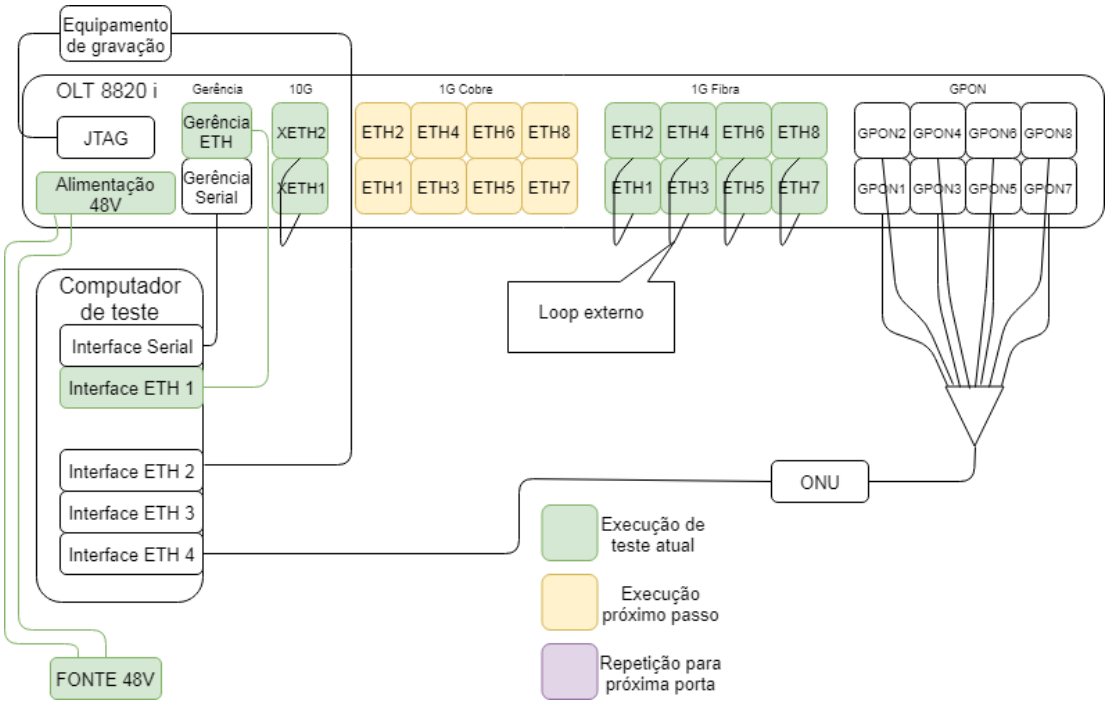


Fonte: O Autor, (2019)

A partir da concepção do teste embarcado, tanto o cenário de teste quanto sua sequência obtiveram melhorias. O novo cenário de testes é demonstrado na Figura 41a, 41b e 41c. Para executar todos os testes são necessárias 3 interações físicas, onde na primeira etapa, testa-se todos os periféricos do processador, verifica-se se os módulos ópticos estão conectados, testa os *loopacks* ativos e o *snake test* no switch, com as portas de fibra conectadas. Na segunda etapa, desconecta-se os módulos das interfaces ETH de fibras, e conecta-se *loops* nas interfaces

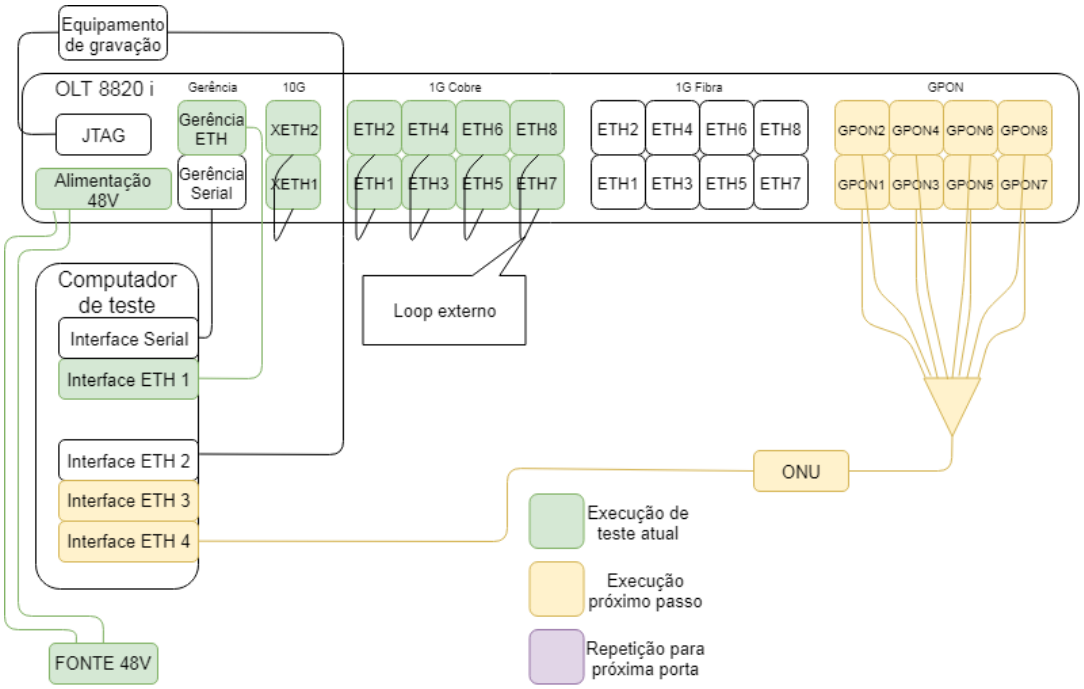
de “cobre”, por último, removem-se os *loops* e conecta-se uma interface ethernet do computador na porta ETH2 da OLT, para execução dos testes GPON:

Figura 41a - Cenário de teste para processador, módulos e interfaces ETH de fibra



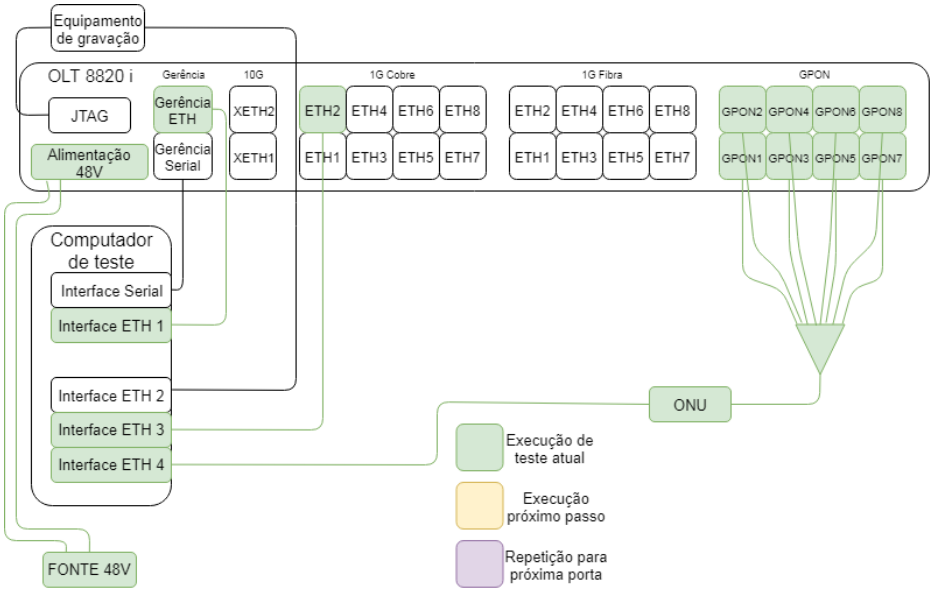
Fonte: O Autor, (2019)

Figura 41b - Cenário de teste para interfaces ETH de “cobre”



Fonte: O Autor, (2019)

Figura 41c - Cenário de teste para interfaces GPON

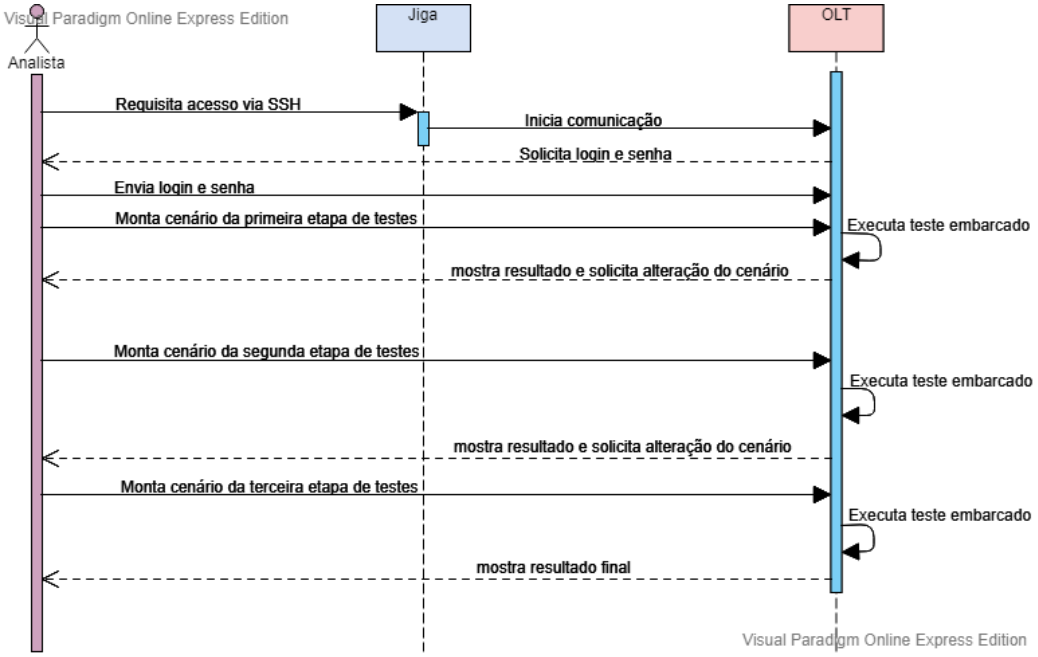


Fonte: O Autor, (2019)

O cenário, torna-se mais simples devido ao fato de utilizar recursos internos da placa, possibilitando a execução funções, que nos testes anteriores dependiam de cenário externo ou demasiadas interações.

Com a alteração do cenário, a execução dos passos de testes, segue a sequência de eventos do diagrama presente na Figura 42, onde a execução do teste, se resume a acessar a OLT rodar software de teste e alterar os cenários quando solicitado.

Figura 42 - Diagrama de sequências para executar teste embarcado



Fonte: O Autor, (2019)

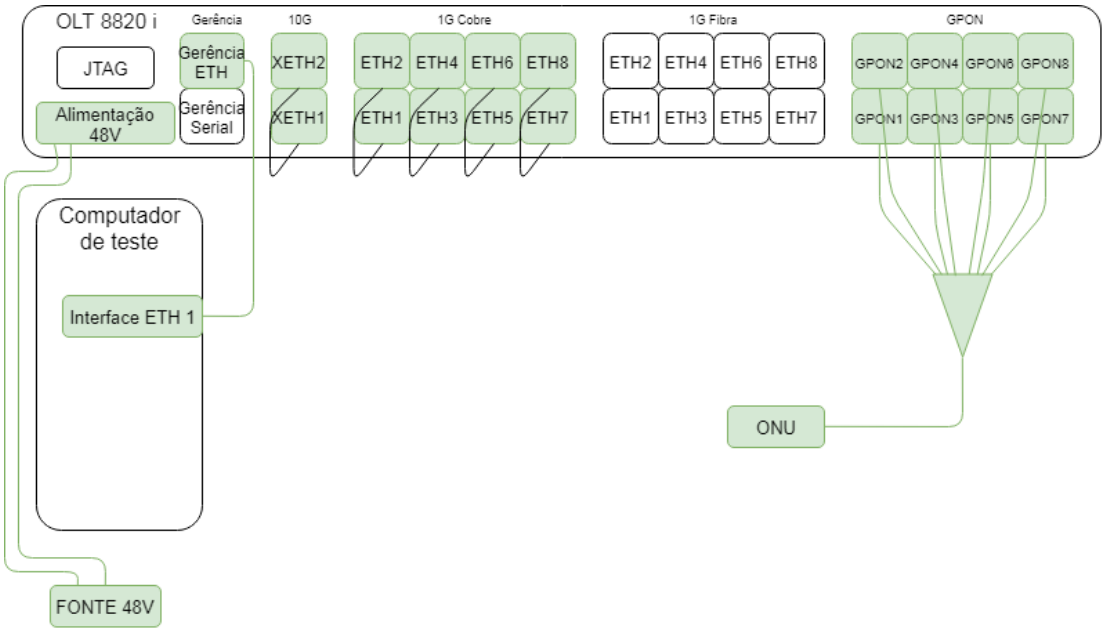
Nesse teste há apenas nove interações entre o testador e o equipamento, devido a simplicidade do cenário e otimizações nos comandos agora embarcados, esse teste é executado em 16 minutos, com maior cobertura de circuitos testados.

3.4 TESTE EM *BURN-IN*

Diferente dos itens anteriores, onde, eram aplicadas melhorias a partir de um cenário pressuposto, com a necessidade de testar a todas interfaces da placa fielmente, o teste de *burn-in* possui outro objetivo, o de evidenciar problemas no processo de fabricação, como solda fria, ou falta de solda. Para isso, esse teste é realizado com longa duração, inicialmente de 76 horas e diminuindo conforme o produto amadurece.

Para atender a necessidade de longa duração, foi alterado o teste embarcado na OLT, o cenário de testes agora precisa ser estático, ou seja, sem interações durante o teste, sendo montado apenas o cenário inicial. Assim esse teste contempla a verificação de periféricos do processador, interfaces ethernet de “cobre” e interfaces GPON sem tráfego de dados de teste, a Figura 43 demonstra esse cenário para uma OLT.

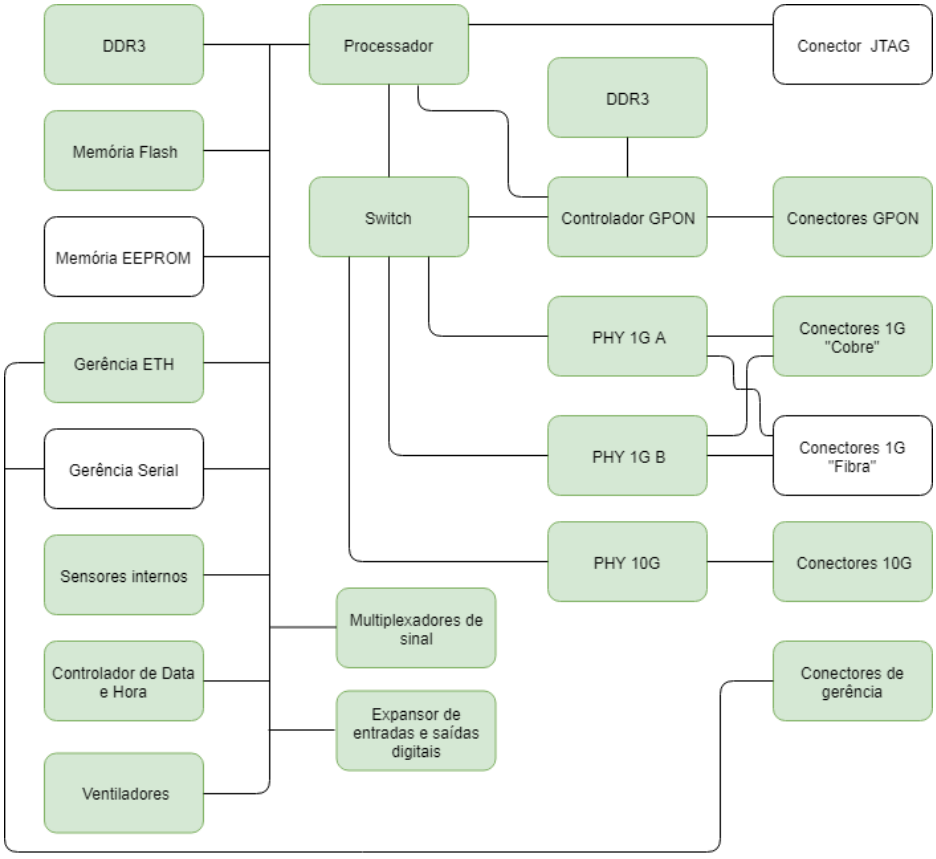
Figura 43 - Cenário de teste em *burn-in*



Fonte: O Autor, (2019)

Com a diminuição do cenário de teste, diminui-se também a cobertura de testes, como pode se observar, os testes das interfaces ethernet de fibra, e interface serial não são cobertas por ele, e a parte GPON é testada apenas com dados de configuração, sendo esse um teste mínimo para a interface, a Figura 44 mostra os circuitos cobertos pelo *burn-in*.

Figura 44 - Circuitos cobertos por teste de *burn-in*



Fonte: O Autor, (2019)

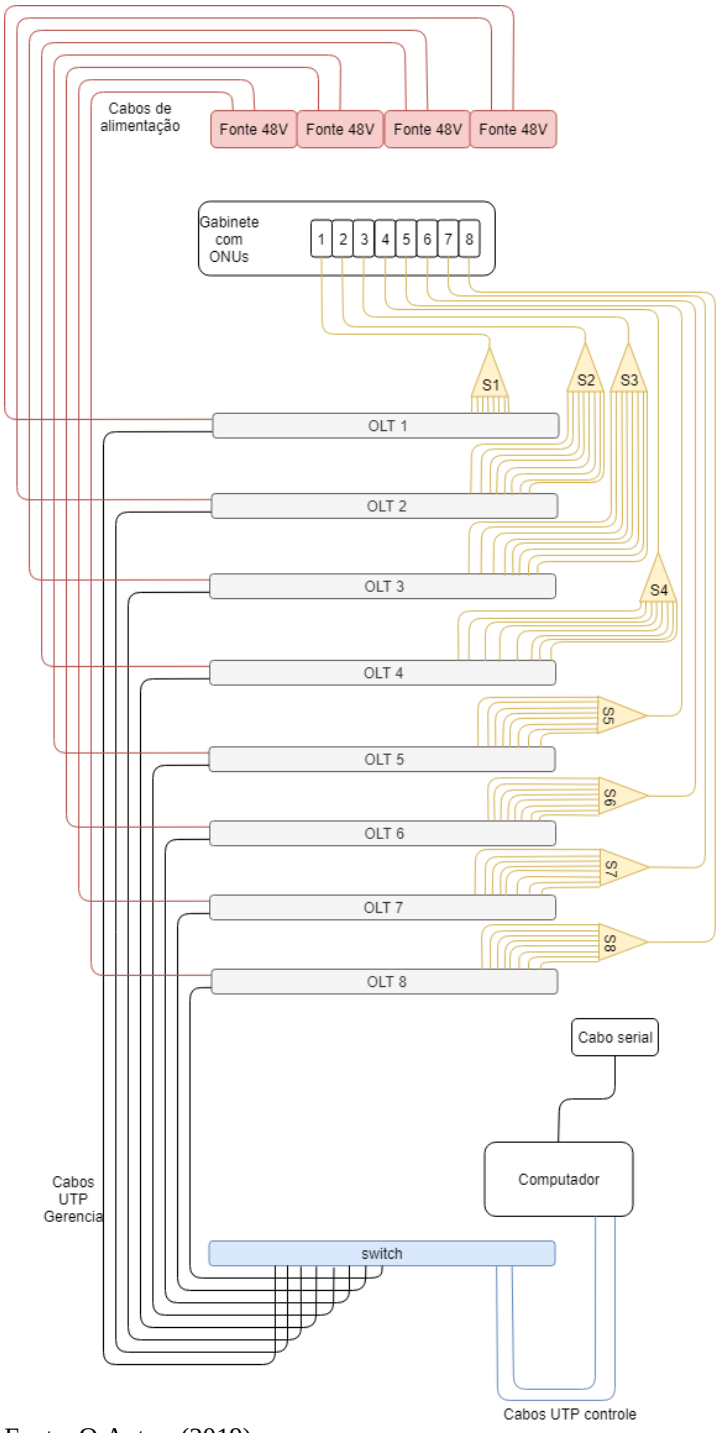
Para viabilizar esse teste de longa duração, é necessário paralelizar o cenário, onde, para atender uma estimativa inicial de 40 OLTs/mês, idealizou-se um teste de 8 OLTs ao mesmo tempo, para isso, além de multiplicar o cenário por oito, foi incluso um switch fazendo a interligação do computador com as interfaces de gerência das OLTs. O cenário completo possui:

- 1 computador
- 1 switch
- 8 fontes de alimentação 48V
- 9 cabos de rede
- 16 módulos 10G com suas respectivas fibras.
- 32 loops de cabo de rede

- 64 módulos GPON
- 8 splitters 1:8
- 8 ONUs
- 8 bandejas de rack para acomodação das OLTs.

O cenário montado pode ser observado pela Figura 45.

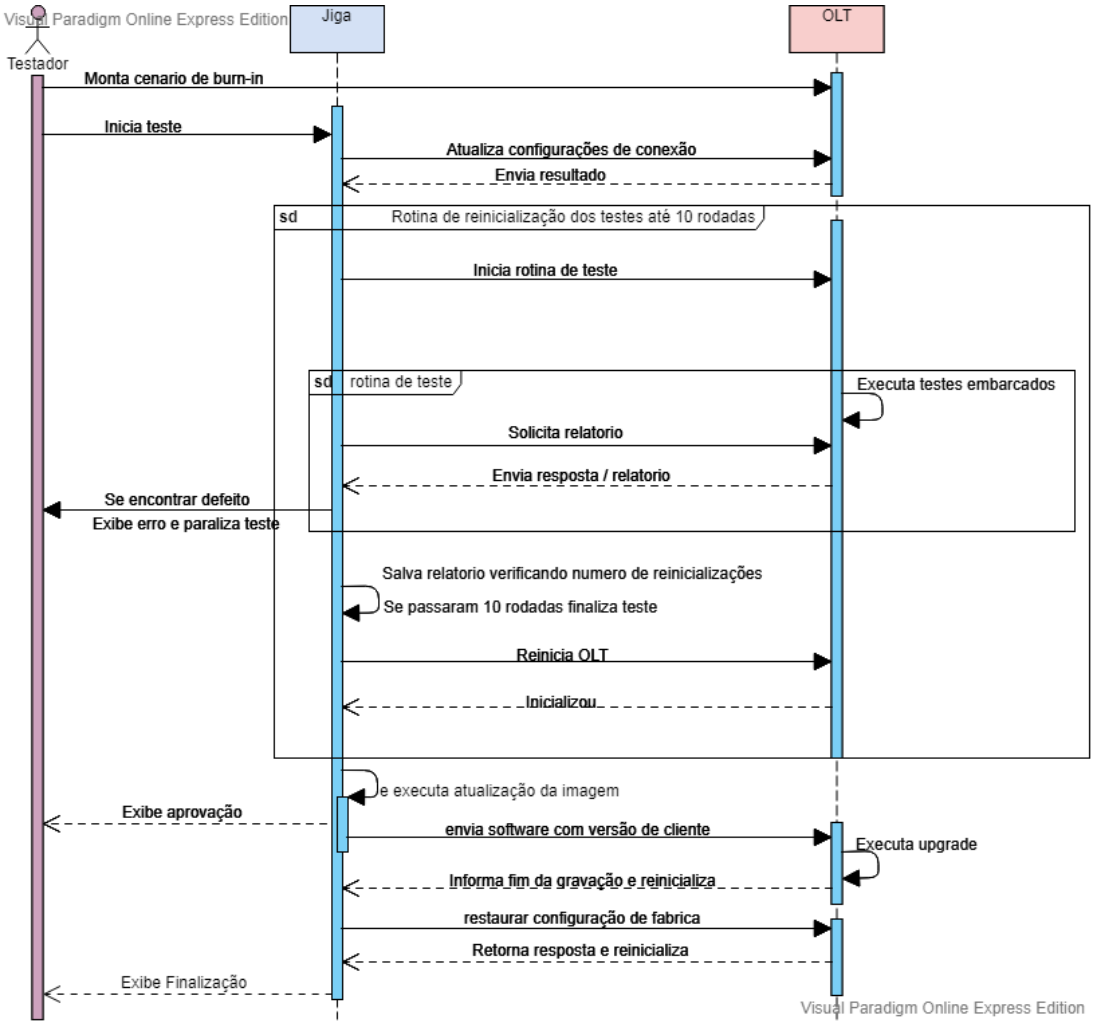
Figura 45 - Cenário de teste em *burn-in*.



Fonte: O Autor, (2019)

Nesse teste, o operador executa as ações de montar o cenário e iniciá-lo, verificando periodicamente se a OLT apresentou algum erro durante o tempo de teste. A Jiga é responsável por manter o teste em andamento, buscando os relatórios parciais na OLT e reiniciando a mesma ao finalizar uma rodada completa de teste. Quando a OLT é aprovada, a Jiga automaticamente atualiza seu firmware para a última versão disponível, após essa atualização, a OLT está apta para seguir para a próxima etapa processo industrial, a Figura 46 demostra o teste de *burn-in* de uma OLT.

Figura 46 - Diagrama de sequências para o teste em *burn-in*



Fonte: O Autor, (2019)

Nesse processo o testador não tem nenhuma interação durante o teste, ou seja, é um teste *black box*, obtendo as vantagens de diminuir os erros humanos inerentes aos processos em geral.

Para abstrair a camada de controle da Jiga que atua sobre o teste embarcado, criou-se máquinas virtuais que rodam a partir do Docker, permitindo que o mesmo teste, rode em oito OLTs diferentes, separadas virtualmente pelos containers do Docker.

3.5 SOFTWARES DA JIGA DE TESTES

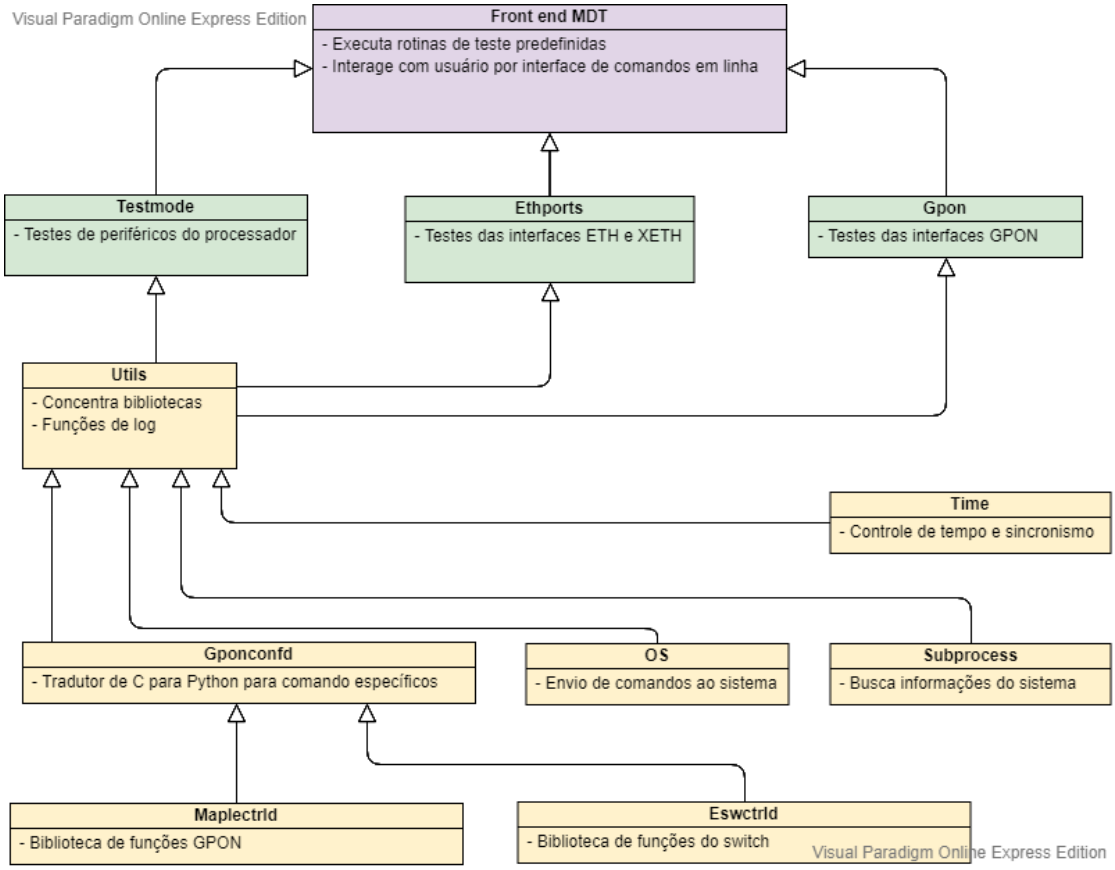
Conforme apresentado anteriormente, os testes das OLTs estão embarcados no equipamento, porém, os softwares de gravação e o controle dos testes de *burn-in* são externos, ou seja, o conjunto de gravação e testes possui dois softwares distintos, um que está presente no equipamento e um que está na Jiga.

3.5.1 Software embarcado

O software de testes embarcado recebeu o nome de modo de teste (MDT). Ele pode ser executado apenas a partir de um firmware específico para testes, dessa forma, esse teste não está disponível para usuários finais do produto, sendo utilizado apenas na validação da fabricação dos equipamentos.

Esse software pode ser dividido em 5 camadas sendo elas, o *front-end*, os comandos de teste, utils e logs, bibliotecas e por fim o software proprietário, o diagrama de classes da Figura 47 ilustra a disposição dessas camadas.

Figura 47 - Diagrama de classes do modo de teste



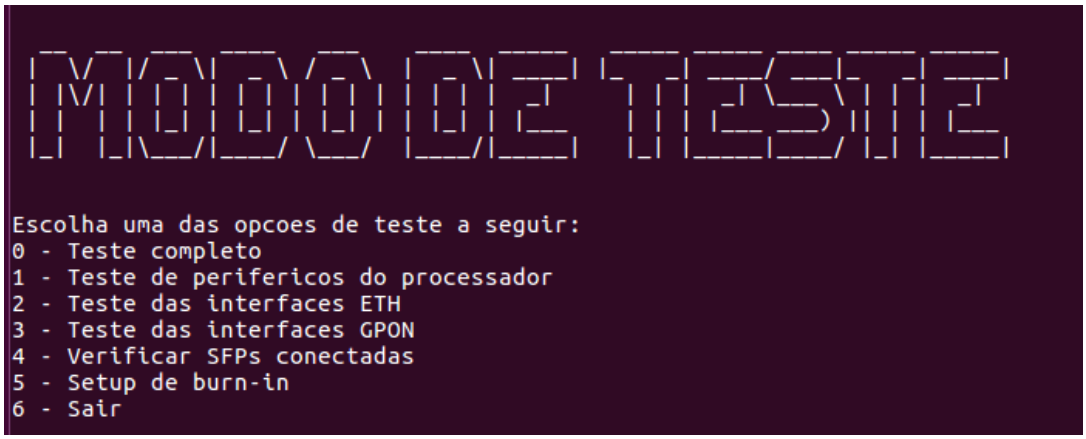
Fonte: O Autor, (2019)

3.5.1.1 *Front-end* do MDT

A primeira camada é o *front-end* do MDT, nela encontrasse a interface de interação entre o operador da Jiga e o equipamento, uma vez que o software está embarcado, foi utilizada uma interface simples, baseada em CLI (*comand line interface*, ou interface de linha de comandos), assim, mantem-se um padrão de interface conforme a que o usuário final possui.

Essa interface possui dois elementos, um banner informando ao operador que ele entrou no MDT, e um menu, onde, a seleção do teste desejado, é feito a partir da digitação do número que corresponde ao teste requerido, pode-se observar essa interface na Figura 48.

Figura 48 - Menu do modo de teste



Fonte: O Autor, (2019)

As opções do menu são:

- Teste completo: executa todas as rotinas em sequência.
- Teste de periféricos do processador: executa a sequência de comandos e verificações, averiguando as memorias e demais CIs conectados ao processador.
- Teste ethernet: Executa os testes locais e remotos do switch, transceptores 1G e do transceptor 10G, além do *snake test*.
- Teste GPON: executa os comandos necessários para configurar os elementos, que compõe o cenário de teste predefinido, validando as interfaces GPON.
- Verificação das SFPs: Testa se os módulos ópticos estão conectados e respondendo corretamente ao processador.
- *Burn-in*: executa a rotina de teste e configurações, para que o cenário do *burn-in* teste as interfaces habilitadas da OLT.

3.5.1.2 Comandos de teste

A segunda camada do MDT, abriga os comandos e suas rotinas menores, nessa camada existem três classes, a testmode, a ethports e a GPON.

- Testmode: Nessa classe, estão concentrados os testes da memória flash, DDRs do processador, Sensores de temperatura, multiplexadores, expansor de IOs, controlador de tempo, ventiladores e a verificação dos módulos SFPs. Cada teste nessa classe possui uma execução diferenciada, não existindo um padrão pré-definido.

- Ethports: Essa classe, utiliza os testes que a Broadcom disponibiliza para seus CIs, incluindo, testes locais e remotos para o switch, e transceptores presentes no equipamento. Há dois métodos de execução para esse teste, um geral, e um para o *burn-in*. O geral, testa as interfaces de "cobre" e as interfaces de fibra em dois momentos distintos, já a rotina de *burn-in*, executa apenas o teste das interfaces de "cobre".
- GPON: O teste realizado por essa classe é 100% funcional, ou seja, ele simula o usuário utilizando o equipamento, porém, com maior eficiência, uma vez que abstrai os comandos predefinidos, aos quais o usuário possui acesso. Como na classe anterior, essa possui dois modos de execução. O primeiro habilita o cenário de teste e aguarda o operador executar um tráfego de dados, enquanto o segundo, utilizado em *burn-in*, executa apenas os comandos de configuração do cenário, testando apenas um tráfego mínimo utilizado na configuração.

3.5.1.3 Logs e concentrador de bibliotecas

Essa classe é um intermediário entre as bibliotecas do firmware e as classes de comando, além de abrigar as funções de log. As funções de log têm grande importância para o teste como um todo, primeiramente essa função é a responsável por manter a rastreabilidade das placas testadas, possuindo logicas aplicadas para parar o teste caso um erro seja encontrado, dessa forma, o operador ou a Jiga de *burn-in*, podem visualizar o erro facilmente durante a execução do teste.

3.5.1.4 Bibliotecas

Nessa camada estão presentes as funções do sistema operacional, ou seja, funções preestabelecidas e de uso comum, tanto o teste quanto a interface do usuário final utilizam essa camada.

As bibliotecas Time, OS, e Subprocess, são utilizadas basicamente para controle, sincronismo geral, e testes de periféricos do processador. A biblioteca OS executa comandos diretamente no sistema operacional do equipamento, e a biblioteca Subprocess permite a verificação de alguns status do sistema operacional.

A biblioteca Gponconfd, roda um tradutor de linguagens de programação, uma vez que o software proprietário é desenvolvido na linguagem C, e as camadas de interface são desenvolvidas em Python.

3.5.1.5 Software proprietário

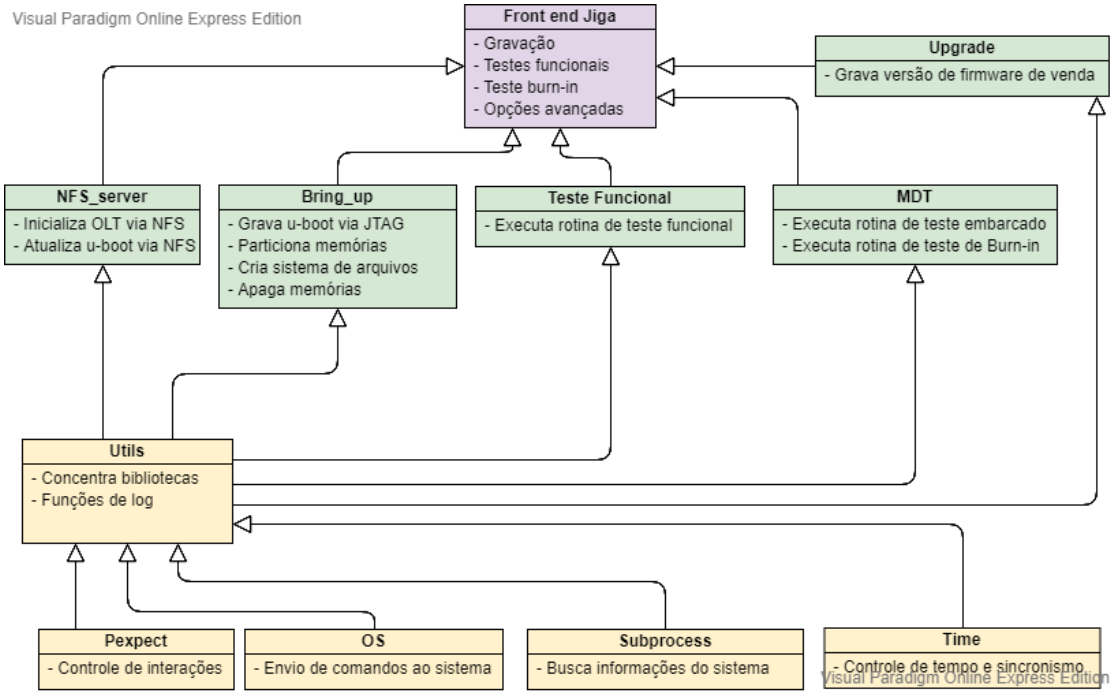
É a camada mais “baixa” de software acessível para o desenvolvedor, a linguagem de programação utilizada é o C, onde cada comando, seja de teste ou de operação, é executado no CI acionado pelo comando. Essa camada é disponibilizada pela fabricante dos CIs, e o desenvolvedor do produto escolhe, habilita e lápida cada função desejada para o equipamento.

Há duas grandes bibliotecas nessa camada o Maplectrld, responsável por administrar as funções do controlador GPON, e o Eswctrld, responsável pelas funções do switch e dos transceptores ethernet 1G e 10G.

3.5.2 Software da Jiga de testes

Esse software possui 4 camadas distintas, como mencionado anteriormente, o software da Jiga de testes não executa os testes diretamente na OLT, mas controla a etapa de *burn-in*, e as gravações necessárias no equipamento, pode-se observar na Figura 49 como as classes desse software se comunicam.

Figura 49: Diagrama de classes da Jiga de testes



Fonte: O Autor, (2019)

3.5.2.1 *Front-end* da Jiga de testes

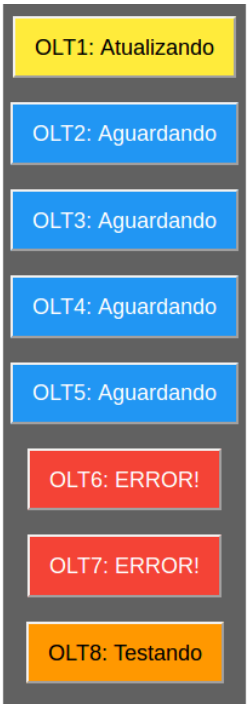
Na primeira camada encontrasse o *front-end* da Jiga, ele é similar ao *front-end* do MDT, porem, possui três menus distintos devido a separação dos processos de montagem. Na primeira etapa, a Jiga precisa executar a gravação do u-boot e rodar um teste completo do MDT, na segunda etapa, grava-se o firmware e é executado o *burn-in*. O terceiro menu é chamado de menu avançado, ele possui todas as funções da Jiga separadas, permitindo a continuação de um processo a partir de pontos conhecido caso seja necessário, pode-se observar pela Figura 50 o menu avançado da Jiga.

Figura 50: Menu da Jiga de testes

Fonte: O Autor, (2019)

Além do *front-end* baseado em CLI, o *burn-in* possui um *front-end* baseado em HTML, para facilitar a visualização do que está acontecendo em cada OLT presente no cenário. O *front-end* do *burn-in* possui cinco estados, “Aguardando”, “Em teste”, “Atualizando”, “Finalizado” e “Erro”, compondo uma interface agradável para o operador trabalhar, pode-se observar o *front-end* do *burn-in* na Figura 51.

Figura 51: Tela de status do *burn-in*.



Fonte: O Autor, (2019)

3.5.2.2 Software de teste e gravação

Nessa camada estão os softwares desenvolvidos para gravar e testar as OLTs. A classe Teste Funcional que se encontra nessa camada é obsoleta, ela está representada na Figura 49 apenas para demonstrar onde ficava o primeiro teste desenvolvido. As outras classes serão explicadas a seguir:

- NFS_Server: Essa classe executa comandos, na sequência necessária para a inicializar uma OLT via NFS, e regravar o u-boot caso seja necessário realizar uma atualização dele.
- Bring_up: Classe responsável por abrigar todas as funções de gravação da Jiga, a partir dela é possível gravar o u-boot via BDI3000, particionar a memória flash, criar os sistemas de arquivos, gravar dados permanentes na EEPROM, realizar o carregamento do firmware compactado e por fim descompactar o firmware.
- MDT: Essa classe é utilizada para controle das rotinas de *burn-in*, como mencionado anteriormente, ela é executada com paralelismo a partir da execução de containers com o Docker. O escopo dessa classe, consiste em iniciar o MDT embarcado na função *burn-in*, e verificar o status do teste, a cada rodada de *burn-in* finalizada, essa classe executa a reinicialização da OLT, e em seguida reinicia o teste, rodando em loop até completar as 76 horas de teste.
- Upgrade: essa classe finaliza o ciclo de processos do produto, após a finalização do *burn-in*, é executado o upgrade do firmware para versão de venda, nas duas partições destinadas a recebê-lo. Em seguida executa uma verificação geral, confirmando que o upgrade foi bem-sucedido.

3.5.2.3 Bibliotecas

Assim como o MDT, a terceira camada da Jiga possui uma classe chamada utils, nela estão concentradas as bibliotecas importadas do sistema operacional, bem como as funções de LOG, na Jiga as funções de log são simples, apenas guardam as informações.

Na quarta camada as bibliotecas utilizadas são, assim como no MDT, o Time, o Subprocess, a OS, adicionadas à biblioteca Pexpect, as três primeiras têm função similar as

descritas no item 3.5.1.4, enquanto a biblioteca Pexpect é responsável pelas funções de interação entre Jiga e equipamento em teste.

4 DISCUSSÃO DOS RESULTADOS

Nesse capítulo são apresentados os resultados e discussões do presente trabalho. Os resultados são expostos por meio de comparações das etapas de desenvolvimento aplicadas.

Com intuito de realizar comparações entre os métodos de gravação e testes, foram realizadas medições de tempo, estimando-se o custo homem/hora, com base em um colaborador que recebe R$ 2000,00 por 220 horas de trabalho mensal.

A Figura 07 deste trabalho, apresenta a OLT com 23 blocos de circuitos, esses blocos representam os circuitos ativos passiveis de teste, omitindo a presença das fontes e circuitos puramente passivos, onde os testes realizados fogem do escopo do projeto. Se contabilizarmos as fontes internas a OLT possui 30 blocos, ou seja, considerando o teste dos 23 blocos, cerca de 77% dos circuitos ativos da placa estão cobertos por testes.

4.1 COMPARAÇÃO ENTRE GRAVAÇÃO MANUAL, AUTOMATIZADA

O método de gravação manual possui 33 interações, onde cada uma, representa uma possibilidade de ocorrência de erro humano. Este método possui um elevado tempo de execução, em média 28 minutos, que para o suposto operador, acarretaria para a empresa um custo de R$ 4,25 por produto. Esse procedimento apresenta apenas uma vantagem, não é necessário criar programas que automatizem suas rotinas, as necessidades são apenas configurar os servidores NFS e SSH corretamente para a gravação. Em contrapartida, a documentação necessária para o operador seguir os passos é ampla.

O método de gravação automatizado dispõe de 7 interações, ele apresenta uma redução de 78% em comparação às interações do teste manual, e minimiza as ocorrências de erros humanos no processo. O tempo de gravação diminuiu 38%, aliviando o custo do processo para R$ 3,05. Além disso, a documentação para operar a Jiga é menor do que a de gravação manual.

4.2 COMPARAÇÃO ENTRE TESTES MANUAIS, AUTOMATIZADOS, EMBARCADOS E *BURN-IN*

O método de teste manual possui 89 interações, o que o deixa complexo, demorado, e susceptível a muitos erros. Seu tempo de execução é em média de 80 minutos, custando cerca de R$ 12,10 por produto testado. Assim como a gravação, a única vantagem é não precisar da criação de uma Jiga, o teste é realizado a partir das funções presentes no software do equipamento. Mesmo com todas as desvantagens esse teste ainda possui aplicação, validar se a Jiga de testes está operando corretamente, para isso uma amostra de todos os lotes passa por esse teste após o *burn-in*.

O teste automatizado trouxe um grande ganho de tempo, junto a redução de 75 interações, diminuindo 65% o valor gasto por produto testado, aumentando a robustez do mesmo, o custo por produto testado fica em R$ 4,25.

Ao embarcar o teste obtém-se maior ganho de tempo e robustez, mais circuitos são testados através de testes aprimorados, o software da Jiga ficou mais simples, pois, a grande massa de comandos gerados para testar foi embarcada, liberando processamento do computador da Jiga para executar testes em paralelo, viabilizando a utilização do Docker para o teste de *burn-in*. O *burn-in* garante uma qualidade maior aos equipamentos fabricados, uma vez que ele filtra os equipamentos com intermitências.

Considerando que após o teste funcional embarcado o operador ainda precisa montar o cenário de *burn-in*, o tempo médio gasto por OLT fica inalterado, porém, o produto ganha robustez, pois é testado por mais de 4000 minutos.

A tabela a seguir demonstra os resultados obtidos em cada etapa do projeto.

Tabela 1 - Resultados obtidos

Processo	Tempo operando (gravação e testes)	Circuitos testados (com base nos 23 blocos apresentados)	Redução de gastos em relação ao processo anterior
Manual	108 minutos	83%	-
Automático	45 minutos	83%	68%
Embarcado	33 minutos	100%	27%
Burn-in*	5 minutos	83% (4000 minutos)	-

Fonte: O autor, (2019)

*O *burn-in* é um processo adicional, ele não substitui a primeira etapa de testes, seja ela manual, automática ou embarcada.

Apesar dos testes manuais e automáticos testarem 83% dos circuitos, esses testes são simples e rápidos, cada interface ou circuito passava em média por 10 segundos de testes, o teste embarcado além de reduzir o tempo operacional, aumentou a carga de testes em circuitos críticos, como, o switch, transceptores e controlador GPON, chegando a durar até 2 minutos em determinadas interfaces.

5 CONCLUSÃO

O desenvolvimento desse projeto mostrou-se um grande desafio, a automatização de testes de hardware não possui grande acervo técnico demonstrando resoluções para cada problema encontrado, sendo assim, o projeto foi desenvolvido baseando-se em testes funcionais, e incrementado com o que as fabricantes dos principais CIs disponibilizam e recomendam para teste de hardware.

O primeiro desafio foi entender como o equipamento funcionava, quais periféricos estavam presentes, e como testar cada um deles. Algumas dessas questões ficaram sem respostas no primeiro momento. Tinha-se conhecimento que a comercialização do produto seria inviável com aplicação de gravação e testes manuais para produção.

Grandes avanços foram obtidos ao utilizar-se a linguagem de programação Python. A biblioteca Pexpect apresentou grande valia, sua utilização acelerou de forma considerável o desenvolvimento do projeto, facilitando as interações entre comandos e respostas requeridos.

Realizou-se o levantamento de circuitos ativos testados em cada etapa, o desenvolvimento dos testes apresentou acréscimo de circuitos testados, porém, não foi possível abranger 80% dos circuitos do produto. Os testes aplicados visam averiguar os circuitos lógicos, deixando descobertos testes em fontes e circuitos puramente passivos.

O desenvolvimento da Jiga apresentou redução de tempo gasto com os processos de gravação e testes, por meio de automatização das interações entre testador e produto.

Ao embarcar os testes, obteve-se uma vasta gama de melhorarias. Possibilitando-se interações diretas com cada circuito, testes predefinidos pelo fabricante do CI, acesso direto ao Linux, e não mais à interface de usuário final. A Jiga foi remodelada para trabalhar com esse novo método, diminuindo-se as interações entre Jiga e produto para a aplicação dos testes. A gravação permaneceu como no princípio, uma vez que não é o principal gargalo dos processos fabris. Os testes embarcados permitiram a utilização do *burn-in*, permitindo que um único computador de controle gerencie diversas OLTs em paralelo.

O projeto visou baixo custo, utilizando cenários funcionais e testes proprietários para averiguar o produto.

5.1 PROJETOS FUTUROS

- Aplicação de testes analógicos – As fontes e demais circuitos puramente passivos podem ser testados de forma automatizada com auxílio de uma cama de pregos.
- Aplicação de testes digitais – Os circuitos Ativos com disponibilidade de *boundary scan* podem ter um teste aprimorado com auxílio de uma ferramenta JTAG.
- Aplicação de teste embarcado no controlador GPON – por falta de tempo esse teste não foi aplicado no projeto atual, o controlador GPON possui características de teste similares a apresentada pelo switch na fundamentação teórica, porem não foi aprofundado esse teste durante o projeto.
- Cenários de teste aprimorados – Hoje o *burn-in* possui limitações de testes, com a modificação do cenário, expandindo a Jiga para que cada porta PON tenha uma ONU conectada o teste em *burn-in* teria ganhos significativos, dando maior robustez ao produto
- Melhorias na interface com usuário e logs – Apenas o *burn-in* possui uma interface amigável com o operador da Jiga, mas os menus de testes e gravações podem ser melhorados com aplicações com front-ends melhores
- Portabilidade do software da Jiga – Como visto a Jiga faz uso de servidores Linux como NFS, SSH e Docker, além de utilizar bibliotecas Python que não são nativas, com isso, toda vez que é necessário replicar a Jiga gastasse um tempo para preparar o computador para operar como Jiga, o desenvolvimento de um instalador resolveria o problema.
- Melhorias mecânicas – Apesar da Jiga reduzir as interações entre o testador e o produto ainda é possível desenvolver um robô, capaz de inserir e remover as OLTs das etapas de gravação e testes, com isso avançaria-se para um sistema 100% automatizado.

6 REFERÊNCIAS

AMAZONAS, José Roberto de Almeida. Introdução. In: AMAZONAS, José Roberto de Almeida. Projetos de comunicações ópticas. Barueri: Manole, 2005. 641 p.

BROADCOM. **Advance datasheet BCM56450.** California: Broadcom Corporation, 2014. 211 p.

CISCO, VNI (Org.). **Cisco Visual Networking Index: Forecast and Methodology, 2016–2021.** 2017. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html>. Acesso em: 18 maio 2018.

COMUNIDADE PYTHON (Org.). **Bibliotecas Python.** Disponível em: <https://www.python.org/>. Acesso em: 08 jun. 2018.

COMUNIDADE ROBOT FRAMEWORK (Org.). **Robot Framework.** Disponível em: <http://robotframework.org/>. Acesso em: 08 jun. 2018.

CRANDALL, Earl. **Power Supply Testing Handbook:** Strategic Approaches in Test Cost Reduction. Nova York: Springer, 1997. 320 p.

INTELBRAS S/A (Org.). **Datasheet_a4_xfs121_xfs122_xfs141_xfs142_xfs181_xfs182_xfs1161_xfs1162_xfs1321_xfs1322.** São José: Intelbras, 2018. Disponível em: <http://www.intelbras.com.br/sites/default/files/downloads/datasheet_a4_xfs121_xfs122_xfs141_xfs142_xfs181_xfs182_xfs1161_xfs1162_xfs1321_xfs1322.pdf>. Acesso em: 08 jun. 2018.

INTELBRAS S/A (Org.). **Datasheet_kpsd_1120_g.** São José: Intelbras, 2018. Disponível em:< http://www.intelbras.com.br/sites/default/files/downloads/datasheet_kpsd_1120_g.pdf>. Acesso em: 08 jun. 2018.

INTELBRAS S/A (Org.). **manual_olt_8820_g_portugues_01-18_site.pdf.** São José: Intelbras, 2018. Disponível em: < http://www.intelbras.com.br/sites/default/files/downloads/manual_olt_8820_g_portugues_01-18_site.pdf>. Acesso em: 08 jun. 2018.

GROOVER, Mikell P. **Automação industrial e sistemas de manufatura.** 3. ed. São Paulo: Pearson, 2011. 561 p. Disponível em: <http://unisul.bv3.digitalpages.com.br/users/publications/9788576058717>. Acesso em: 08 jun. 2018.

IEEE. **Institute os Eletrical and Electronics (IEEE) Standard Glossary of Softwre Engineering Terminology**. [S.1.: s.n.]. 1990.

IEEE. **Institute os Eletrical and Electronics (IEEE) Standard Test Access Port and Boundary Scan Architecture**. [S.1.: s.n.]. 2001.

INTERNATIONAL TELECOMMUNICATION UNION. **G.984.1**: SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS. Geneva: International Telecommunication Union, 2008.

JUKNA, Rob; JIN, Harry. **A Flexible Fixturing System for In-Circuit Test of High Node Count Circuit Boards.** Disponível em: <http://www.circuitinsight.com/pdf/flexible_fixturing_system_ipc.pdf>. Acesso em: 15 jun. 2018.

MULLER, João. **O futuro das redes PON: tendências de mercado.** 2016. Disponível em: <https://www.cianet.com.br/o-futuro-das-redes-pon-tendencias-de-mercado/>. Acesso em: 15 abr. 2018.

NATIONAL INSTRUMENTS. **What Is LabVIEW?** Disponível em: <http://www.ni.com/en-us/shop/labview.html>. Acesso em: 08 jun. 2018.

NATIONAL INSTRUMENTS. **Digital Instruments.** Disponível em: <http://www.ni.com/en-us/shop/select/digital-instruments-category>. Acesso em: 08 jun. 2018.

OLIVEIRA, Patrícia Beneti de. Tipos de Rede Óptica Passiva: GPON – Gigabit Passive Optical Network. In: OLIVEIRA, Patrícia Beneti de. Soluções de Atendimento em Fibra Óptica I. São Paulo: Teleco, 2014. Cap. 1. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialsolfo1/pagina_3.asp>. Acesso em: 22 abr. 2018.

PASSIVE Optical Network (PON) Equipment Market Analysis, Size and Share to 2024. USA: Grand View Research Inc., 2016. Disponível em: <https://grandviewresearchinc.wordpress.com/2016/06/07/passive-optical-network-pon-equipment-market-analysis-size-and-share-to-2024/>. Acesso em: 06 mai. 2018.

PRADHAN. D. K. **Fault-Tolerant Computer System Design.** [S.1.]: Prentice Hall. 1996. 560p

SALVETTI, Alfredo Roque. **A História Da Luz.** 2. ed. São Paulo: Livraria da Física, 2008. 206 p.

XJTAG LTD (Org.). **Introduction to JTAG:** Technical Guide to JTAG. Disponível em: <https://www.xjtag.com/about-jtag/>. Acesso em: 08 jun. 2018.

SILVA, Karina Rocha Gomes da. **Uma Metodologia de Verificação Funcional para Circuitos Digitais.** 2007. 133 f. Tese (Doutorado) - Curso de Doutor em Ciências no Domínio da Engenharia Elétrica, Universidade Federal de Campina Grande, Campina Grande, 2007.

SAUNOIS, Lucie. **Black box, grey box, white box testing: what differences?** 2015. Disponível em: <https://www.nbs-system.com/en/blog/black-box-grey-box-white-box-testing-what-differences/>. Acesso em: 05 mar. 2019.

GENRAD (Ed.). **Introduction to In-Circuit Testing.** Massachussets: Genrad. Inc, 1984. 123 p.

HINTERMANN, Martin. **Operating System Components for an Embedded Linux System.** 2007. 80 f. Dissertação (Mestrado) - Curso de Robótica, Universidade Técnica de Munique, Munique, 2007.

JONES, M. Tim. **Anatomia do Kernel Linux:** Histórico e Decomposição Arquitetural. 2007. Disponível em: <https://www.ibm.com/developerworks/br/library/l-linux-kernel/index.html>. Acesso em: 05 mar. 2019.

SRIVASTAV, Prakhar. **Docker for beginners.** Disponível em: <https://docker-curriculum.com/>. Acesso em: 05 mar. 2019.

MEIRELLES, Adriano. **Redes e Servidores Linux.** 2. ed. Porto Alegre: Hardware, 2006. Disponível em: <https://www.hardware.com.br/livros/linux-redes/servidores-linux.html>. Acesso em: 05 mar. 2019.

LARSSON, Erik. **Introduction to Advanced System-on-Chip Test Design and Optimization:** Frontiers in Electronic Testing. Nova York: Springer, 2005. 388 p.

W3C (Comp.). **WEB DESIGN AND APPLICATIONS.** Disponível em: <https://www.w3.org/standards/webdesign/>. Acesso em: 05 mar. 2019.