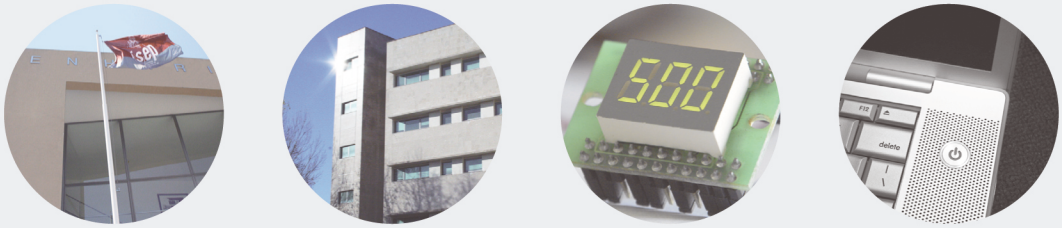
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Circuito de demonstração de FPAA com infraestruturas IEEE1149.4

NUNO MIGUEL FERNANDES CRUZ
novembro de 2016

POLITÉCNICO
DO PORTO

CIRCUITO DE DEMONSTRAÇÃO DE FPAA COM INFRAESTRUTURAS IEEE1149.4

Nuno Miguel Fernandes Cruz

Departamento de Engenharia Electrotécnica

Mestrado em Engenharia Electrotécnica e de computadores – Automação e sistemas

Instituto Superior de Engenharia do Porto

2016

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Nuno Miguel Fernandes Cruz, Nº 1100454, 1100454@isep.ipp.pt

Orientação científica: Prof. Manuel Carlos Malheiro de Carvalho Felgueiras, mcf@isep.ipp.pt

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

14 de novembro de 2016

Agradecimentos

Gostaria de exprimir os meus agradecimentos a várias pessoas que contribuíram direta ou indiretamente para que o desenvolvimento deste trabalho fosse possível.

A nível institucional, pretendo agradecer ao meu orientador Professor Doutor Carlos Felgueiras por todo o apoio e incentivo dado, pela confiança depositada, pela ajuda fornecida durante a realização do trabalho, assim como pela facilidade de fornecimento do material necessário para o desenvolvimento do mesmo.

A nível profissional, agradeço ao Engenheiro Rodolfo por ter dado a oportunidade de me juntar à equipa Evoleo, obtendo assim maiores conhecimentos e capacidades para o desenvolvimento da minha carreira pessoal. Quero agradecer também a toda a equipa da Evoleo pela amizade, pelo apoio e pela ajuda fornecida, em especial aos Engenheiros António e Carlos Silva.

A nível pessoal, gostaria de agradecer a todos os meus amigos que me acompanharam e motivaram durante a realização deste projeto. Ao Diogo Cunha, Manuel Fernando, Bruno Alves, João Lourenço, João Lima, Tiago Fernandes, André Fernandes, Carlos Bessa, Ricardo Balacó, Carla Santos, João Miguel, Hugo Freire e ao Tiago Marques um sincero agradecimento pelos momentos partilhados e pelo apoio que me deram ao longo de todo este tempo.

Agradeço à minha namorada, por estar presente em todos os momentos, pelo carinho dado e pela compreensão.

Por último, gostaria de agradecer aos meus pais e avós pelo apoio e pelo sacrifício que fizeram para que tudo isto fosse possível de ser realizado.

Resumo

Nos últimos anos, tornou-se notável o grande avanço na tecnologia, e com este surgiram novos equipamentos que tornaram a vida dos utilizadores mais facilitada. Cada equipamento teve uma evolução progressiva, notando-se uma diminuição dos componentes constituintes de cada equipamento, de forma a torná-lo mais leve, mais pequeno e com as mesmas funcionalidades. Com este avanço tecnológico, a procura por equipamentos novos aumentou, fazendo com que empresas de desenvolvimento tornassem a produção mais rápida e eficaz. Para tal a realização de testes a lotes ou a equipamentos individuais, tornou-se um papel fundamental para que o equipamento fosse entregue ao cliente pronto para utilização. Sendo o encapsulamento dos componentes cada vez mais pequeno, a realização de testes em produtos finais começou a tornar-se um problema, devido à dificuldade de acesso físico a pontos estratégicos de teste. Para tal foram desenvolvidas infraestruturas normalizadas capazes de ajudar na depuração de cada um dos equipamentos sem acesso direto aos pinos, das quais se pode destacar a norma IEEE1149.1 a nível digital e a IEEE1149.4, sendo esta baseada na norma anterior, mas com extensão para o domínio dos circuitos analógicos e mistos.

Os dispositivos reconfiguráveis têm tornado possível a evolução de equipamentos mais pequenos, permitindo criar diversos circuitos no seu interior através de programação. Ao longo do tempo tem-se deparado que os dispositivos reconfiguráveis têm evoluído maioritariamente através da eletrónica com a utilização de Field-Programmable Gate Array (FPGA). O sucesso destes circuitos no domínio digital teve reflexo também no domínio analógico, os quais assumem especial importância, sendo denominados por Field-Programmable Analog Arrays (FPAAs). Presentemente os dispositivos FPGA’s já incluem meios, frequentemente baseados na infraestrutura IEEE1149.1, que permitem a realização de um conjunto importante de operações de depuração do circuito. No entanto, as FPAA’s, encontram-se desprovidas desses meios, estando as operações de depuração e/ou teste limitadas às de acesso físico aos pinos antes da respetiva introdução no circuito global. É apenas possível realizar uma simulação de forma a perceber o possível estado do sistema, sendo necessário o acesso direto aos pinos para validar que o sistema funciona tal como foi

configurada. Dado que uma parte importante do sucesso da infraestrutura IEEE1149.1 se deveu às suas características notáveis para apoiar operações de depuração em circuitos digitais, vale a pena analisar de que modo é que a infraestrutura IEEE1149.4 poderá apoiar as mesmas operações no domínio analógico.

Desta forma, para criar um mecanismo de verificação funcional, realiza-se a interligação entre as FPAA's e os dispositivos que implementem a infraestrutura IEEE1149.4.

Para alargar o interesse pelo desenvolvimento de aplicações com utilização de FPAA's, sem necessidade de utilização de uma placa de desenvolvimento, foram desenvolvidos meios de apoio ao ensino. Assim, a aproximação do aluno ao projeto para configuração de uma ou várias FPAA's, com utilização de um microcontrolador externo, será mais facilitada.

No presente trabalho, desenvolve-se uma solução capaz de tornar as FPAA's acessíveis através de um único ponto para controlo e observação do sistema, sem necessidade de acesso direto aos pinos, facilitando-se assim as tarefas de teste e/ou depuração.

Palavras-Chave

Teste, depuração analógica, verificação funcional analógica, IEEE1149.1, IEEE1149.4, FPGA, FPAA

Abstract

In the last years, it has become remarkable the great advance in technology, and with this one emerged new equipment that made life of users easier. Each equipment had a progressive evolution, noting a decrease of the components that constitute each equipment, in order to make it lighter, smaller and with the same features. With this technological advancement, the demand for new equipment increased, causing that development companies made production more fast and efficient. For such a testing of batches or individual equipment, it has become a major role so that the equipment was delivered to the customer ready for use. As the package of the components became smaller, making tests in final products started becoming a problem, due to difficulty of access to strategic points of test. For such standard infrastructure were developed, capable of helping in the debugging of each of the equipment without direct access to pinout, of which we can highlight the norm IEEE1149.1 in digital level and the IEEE1149.4, being this one based on the previous norm, but extending to the domain of the analog and mixed circuits.

Reconfigurable devices are making possible the evolution of smaller devices, allowing to create several circuits inside each one by programming. Over time it has noted that reconfigurable devices have evolved mainly through the electronics with the use of Field-Programmable Gate Array (FPGA). The success of these circuits in the digital domain was also reflected in the analog domain which takes special importance, being called by Field-Programmable Analog Arrays (FPAA). Nowadays the FPGA's devices already include means, often based on IEEE1149.4 infrastructure, that allow the realization of an important set of circuit debugging operations. However, the FPAA's are devoid of such means, being the debug operations and/or test limited to physical access to pinout before the introduction into the global circuit. Is possible only to carry out a simulation in order to understand the possible system state, requiring direct access to pinout to validate that the system works as it was configured. As an important part of the success of IEEE1149.1 infrastructure was due to its notable features to support debug operations in digital circuits, it is worth examining how is that IEEE1149.4 infrastructure may support the same operations in the analog domain.

So, to create a functional verification mechanism, is carried out the interconnection between the FPAA's and devices that implement the IEEE1149.4 infrastructure.

To extend the interest in developing applications with use of FPAA's, without need to use a development board, means have been developed to support education. So, the approach of the student to the project to configure one or several FPAA's, using an external microcontroller, will be facilitated.

In the present work, it is developed a solution capable of making the FPAA’s accessible through a single point for control and observation of the system, without the need of direct access to pinout, thereby facilitating the test task and/or debugging.

Keywords

Test, analogic debugging, Analog functional Verification, IEEE1149.1, IEEE1149.4, FPGA, FPAA.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AB1/AB2	–	Linhas de barramento interno analógico
ABM	–	*Analog Boundary Module*
ADC	–	*Analog-to-Digital Convertor*
AMPOP	–	AMPlificadores OPeracionais
AOI	–	*Automated Optical Inspection*
ATAP	–	*Analog Test Acess Port*
BGA	–	*Ball Grid Array*
BSC	–	*Boundary Scan Cell*
BSR	–	*Boundary Scan Register*
CAB	–	*Configurable Analog Blocks*
CAM	–	*Conigurable Analog Module*
CD	–	*Compact Disc*
CLB	–	*Configurable Logic Block*
CMOS	–	*Complementary Metal-Oxide-Semiconductor*
CSP	–	*Chip Scale Package*
DBM	–	*Digital Boundary Module*
DIP	–	*Dual In-Line Package*
dpASP	–	*Dynamically Programmable Analog Signal Processors*
FPAA	–	*Field-Programmable Analog Arrays*
FPGA	–	*Field-Programmable Gate Array*
GND	–	*Ground*
ICT	–	*In-Circuit Test*
IEEE	–	*Institute of Electrical and Electronics Engineers*
IFSA	–	*International Frequency Sensor Association*
JETAG	–	*Joint European Test Action Group*
JTAG	–	*Joint Test Action Group*
MOSI	–	*Master out – Slave in*
MUX	–	*MUltipleXeres*
NI	–	*National Instrumments*

OCR	–	*Optical Character Recognition*
PCB	–	*Printed Circuit Board*
PIC	–	*Programmable Intelligent Computer*
PLD	–	*Programmable Logic Device*
PROM	–	*Programmable Read-Only Memory*
PWM	–	*Pulse-Width Modulation*
QFN	–	*Quad Flat No leads*
QFP	–	*Quad Flat Package*
S&H	–	*Samples and Hold*
SIP	–	*System-in-Package*
SPI	–	*Serial Peripheral Interface*
TAP	–	*Test Acess Port*
TBIC	–	*Test Bus Interface Circuit*
TCK	–	*Test ClocK*
TDI	–	*Test Data Input*
TMS	–	*Test Mode Select*
TDO	–	*Test Data Output*
TRAC	–	*Totally Reconfigurable Analog Circuit*
TRST	–	*Test ReSeT*
TTL	–	*Transistor-Transistor Logic*
UART	–	*Universal Asynchronous Receiver/Transmitter*
USB	–	*Universal Serial Bus*
VG	–	*Voltage Ground*
VH	–	*Voltage High*
VL	–	*Voltage Low*
VLSI	–	*Very-Large-Scale Integration*
WLP	–	*Wafer Level Package*

Glossário

Ciclo de vida do circuito – Conjunto de etapas que um circuito passa durante o seu tempo de vida útil. O ciclo de vida de um circuito é iniciado no conceito do projeto até que a utilidade do circuito para o utilizador termine.

Circuito analógico – Circuito que processa sinais puramente analógicos. Exemplo: amplificador.

Circuito digital – Circuito que processa sinais puramente digitais. Exemplo: microprocessador.

Circuito de missão – Circuito que tem como finalidade realizar uma determinada função

Circuito integrado – Circuito eletrónico que incorpora diversos componentes no seu interior de forma a realizar a função para o qual foi produzida.

Circuito misto – Circuito que processa sinais analógicos e digitais, sendo estes interligados entre si. Exemplo: conversor analógico para digital.

Comunicação SPI – Comunicação realizada entre dois dispositivos, sendo um o dispositivo master e os restantes os slaves, esta comunicação é realizada através do envio de dados série.

Controlo – Operação realizada de forma a impor um estado, num ponto específico do circuito.

Defeito – Estado físico que poderá impedir o bom funcionamento do circuito.

Depuração – Atividade realizada para detetar e eliminar erros de projeto e eventuais defeitos presentes num circuito.

Detecção – Operação que define se um sistema possuiu erros de projeto e eventuais defeitos.

Dispositivo lógico reconfigurável – Componente eletrónico que permite ser configurado através de programação, de forma a realizar diferentes tipos de funções

Duty-cycle – Parâmetro da técnica de PWM que define a relação entre o tempo em que o sistema se encontra com a sua saída ativa e o tempo em que se encontra com a saída inativa.

Erro – Representa um desvio face ao comportamento esperado pelo circuito

Experimentação remota – Infraestrutura que possibilita o acesso a recursos, permitindo realizar experiências práticas em qualquer hora e em qualquer lugar.

FPAA – Dispositivo lógico reconfigurável que possibilita desenvolver circuitos analógicos através de configuração.

FPGA – Dispositivo lógico reconfigurável que possibilita desenvolver circuitos digitais através de configuração.

Frequência – Grandeza física que indica o número de ocorrências de um evento, num determinado espaço temporal.

Gerador de funções – Dispositivo eletrónico utilizado para gerar sinais elétricos com a capacidade de definir a forma de onda, frequência e amplitude do sinal.

Hexadecimal – Sistema de numeração, que representa os número em base 16, permitindo gerar números de 0 a 9 e letras de A a F, que no sistema decimal representam os valores de 0 a 15.

Infraestrutura IEEE1149.1 – Infraestrutura de apoio ao teste e depuração de circuitos digitais

Infraestrutura IEEE1149.4 – Infraestrutura de apoio ao teste e depuração de circuitos analógicos e mistos.

Máquina de estados – Definição dos diferentes estados que o controlador poderá ter, sendo apenas possível estar num estado de cada vez. Para existir uma mudança de estado deverá ser validada uma condição que permita realizar a transição entre o estado atual e o estado seguinte.

Multiparametricidade de um sinal – Capacidade de utilizar apenas um único sinal de forma a obter mais do que uma simples infinidade de valores, permitindo observar as diferentes grandezas individualmente.

Observação – Operação realizada de forma a analisar um estado, num ponto específico do circuito.

Osciloscópio – Instrumento de medida de sinais eletrônicos que apresenta gráficos bidimensionais de um ou mas sinais elétricos.

Placa de circuito impresso – Placa que apoia e liga eletricamente os componentes eletrónicos através de condutores gravados em cada camada de cobre.

Sinal analógico – Sinais análogos aos sinais físicos. A característica principal destes sinais é que podem conter qualquer valor na amplitude (uma infinidade de valores) dentro de certos limites.

Sinal digital – Sinais com informação por meio de uma sequência de números discretos, onde cada um dos números representa um valor de sinal para cada instante de tempo. Estes podem assumir um conjunto finito de valores na amplitude.

Sinal quasi-digital – Sinais que exploram alguns parâmetros com variação contínua ao longo dos seus valores limites. Um exemplo prático de utilização destes sinais é a técnica de modulação de largura de impulso PWM.

Técnica de PWM – Técnica normalmente utilizada para controlo contínuo de grandezas, e.g velocidade de motores de corrente contínua. Esta técnica permite controlar a frequência e o parâmetro *duty cicle*.

Teste – Atividade que consiste em aplicar estímulos num circuito e de observar respostas de forma a validar funcionalmente o circuito.

Verificação – Operação que consiste em confirmar se o estado do sistema está de acordo com o esperado.

1. INTRODUÇÃO

Neste capítulo é apresentado o tema principal deste trabalho, sendo demonstrado o prolema atual e definida uma proposta de solução para diminuir o impacto do mesmo. Para a solução aqui definida serão identificados os objetivos principais, realizando uma pequena interligação com os requisitos necessários de modo a desenvolver a solução apresentada. A parte final deste capítulo irá apresentar a organização do relatório.

1.1. CONTEXTUALIZAÇÃO

Os dispositivos reconfiguráveis denominados por *Field-Programmable Gate Array* (FPGA), têm vindo a afirmar-se no mundo da eletrónica digital, permitido embutir diversos circuitos digitais no seu interior. Com o objetivo de alargar o leque deste tipo de circuitos à área analógica, existem os dispositivos denominados por *Field-Programmable Analog Arrays* (FPAAs). Torna-se assim importante desenvolver uma plataforma que permita explorar esta vertente tanto em ambiente de aula presencial como por acesso remoto. Sendo estes dispositivos programáveis, torna-se necessário validar que o sistema para o qual foi configurado se encontra a funcionar corretamente. As FPGA's utilizam sistemas capazes de realizar o controlo e observação, sem necessidade de acesso direto aos pinos, tendo assim a capacidade alargada para validar o seu funcionamento, no entanto as FPAA's ainda não são capazes de realizar a depuração sem acesso direto ao dispositivo.

Dado que a infraestrutura 1149.1 se revelou um mecanismo incontornável para apoio à depuração dos circuitos digitais, tipo FPGA, importa analisar de que modo a 1149.4 pode ser reutilizada no mesmo sentido para o domínio dos analógicos e mistos. Desta forma este trabalho irá ser desenvolvido de forma a implementar este sistema de depuração para permitir a realização de testes de forma mais simplificada.

1.2. OBJETIVOS

O presente trabalho tem por objetivo desenvolver um circuito através da qual se possa configurar uma FPAA, verificando se o respetivo funcionamento corresponde ao pretendido. Para tal é necessário injetar um conjunto de sinais nas entradas e comparar as saídas com os valores esperados, sem que seja necessário aceder diretamente aos pinos do dispositivo configurável. A infraestrutura IEEE1149.4 é especialmente útil para este tipo de operações no domínio analógico. Nesta fase pretende-se desenvolver um circuito alvo para experimentação local mas que deverá posteriormente evoluir para experimentação remota.

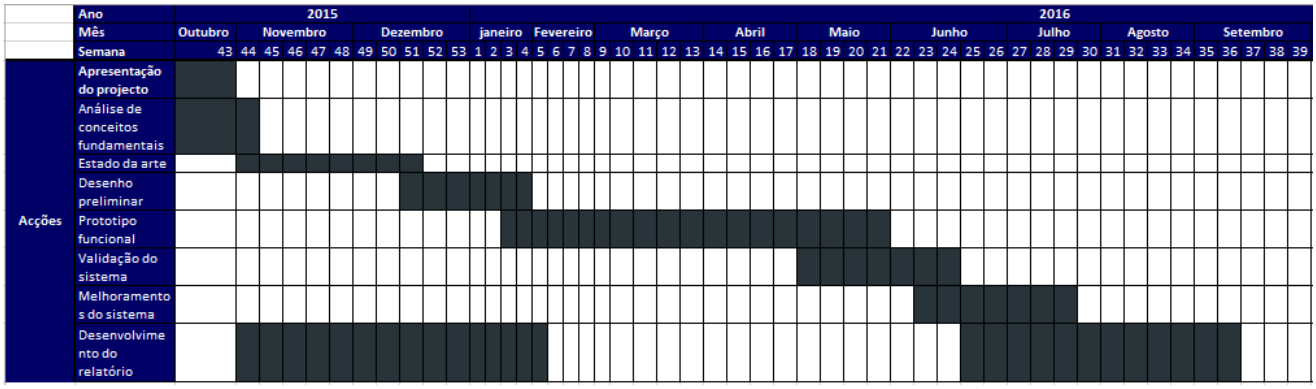
Tarefas a realizar:

- Estudo das infraestruturas IEEE 1149.4 e IEEE1149.1;
- Estudo de FPAAs (arquiteturas, modos de funcionamento, fabricantes, etc);
- Verificação funcional de circuitos analógicos flexíveis;
- Detalhe do projeto de configuração da FPAA, para apoio ao ensino;
- Desenvolvimento de circuitos e correspondente carta de circuito impresso;
- Desenvolvimento de aplicação única capaz de realizar controlo, observação e programação do sistema;
- Desenvolvimento de uma infraestrutura que permita integrar todos os elementos desenvolvidos com vista à experimentação remota.

1.3. CALENDARIZAÇÃO

Para o desenvolvimento da solução existiu a necessidade de realizar um planeamento progressivo de todas as tarefas principais a serem desenvolvidas ao longo do projeto. Este planeamento encontra-se apresentado na Tabela 1, onde é possível verificar os tempos previstos a serem gastos em cada uma das tarefas.

Tabela 1 Calendarização do projeto



1.4. ORGANIZAÇÃO DO RELATÓRIO

O relatório encontra-se dividido e 7 capítulos, sendo esta divisão detalhada como se segue.

No capítulo 1 é feita uma introdução ao tema principal deste trabalho, de forma a dar a conhecer ao leitor, em poucas palavras, o objeto que se pretende desenvolver.

No capítulo 2 são introduzidos alguns conceitos fundamentais em que o leitor deverá ter noção de forma a compreender algumas das opções tomadas para a elaboração da Tese.

O capítulo 3, o do estado da arte, será feita uma análise de todos os subsistemas principais deste trabalho de forma a tornar claro o funcionamento de cada um, bem como dar a conhecer alguns sistemas úteis para o desenvolvimento do mesmo.

Conhecendo o problema atual, no capítulo 4, encontra-se a proposta de solução, onde serão descritas todas as tarefas a realizar.

De forma a descrever todos os meios de desenvolvimento deste trabalho de forma mais aprofundada, no capítulo 5, encontra-se a implementação da solução. Será neste capítulo que serão descritas de forma mais clara todas as tarefas propostas no capítulo 4.

No capítulo 6 encontra-se a validação do sistema final, com realização de testes de funcionamento de todo o sistema.

Para finalizar, no capítulo 7 serão apresentados os resultados conclusivos de todo este trabalho.

2. CONCEITOS FUNDAMENTAIS

Neste capítulo apresentam-se alguns dos conceitos fundamentais úteis para a compreensão do desenvolvimento do trabalho. Todos os conceitos aqui abordados, serão explicados para que qualquer leitor consiga perceber os capítulos seguintes.

2.1. TIPOS DE SINAIS

Ao nível elétrico/eletrónio um sinal é uma forma de transmissão de informação, informação esta, que pode ser relacionada com uma vasta gama de parâmetros no mundo físico. Podem ser identificados vários exemplos tal como, a troca de informação sobre o tempo atual utilizando a leitra de pressões, temperatura, velocidade do vento, humidade relativa do ar, entre outros. Cada uma destas informações são transmitidas entre dois equipamentos utilizando sinais. Para obter informação de cada um dos sinais é necessário realizar processamento dos mesmos utilizando sistemas eletrónicos. Inicialmente é necessário converter a informação em sinal elétrico (tensão ou corrente). Esses processos são realizados por dispositivos conhecidos como transdutores. Um transdutor de pressão, tal como é utilizado num microfone, transforma as ondas sonoras produzias pela voz de um ser humano em sinais elétricos. Um sinal é considerado uma grandeza variável no tempo que transmite o conteúdo de informação através das variações na amplitude com o decorrer do tempo. A forma de transmissão de informação pode ser realizada de três formas distintas:

- Sinais digitais
- Sinais analógicos
- Sinais quasi-digitais

No que respeita aos sinais digitais, estes enviam a informação por meio de uma sequência de números discretos, em que cada um dos números representa um valor de sinal para cada

instante de tempo. Os sinais digitais podem assumir um conjunto finito de valores na amplitude. Um exemplo prático de utilização de sinais digitais é a leitura de informação de CDs de música. Na Figura 1 encontra-se ilustrado um exemplo de um sinal digital, como representação de um código binário[16].

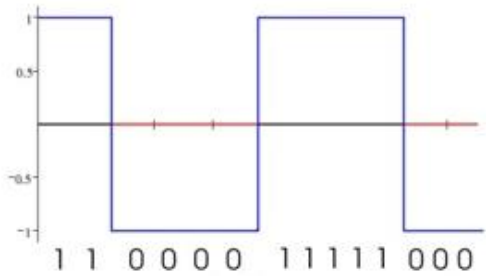


Figura 1 Exemplo de um sinal digital em código binário [1]

Os sinais analógicos são sinais que são análogos aos sinais físicos. A característica principal destes sinais é que podem conter qualquer valor na amplitude (uma infinidade de valores), dentro de certos limites. Um exemplo típico de utilização de sinais analógicos é a representação de um som captado através de um microfone. Na Figura 2 encontra-se ilustrado um exemplo de um sinal analógico [16].

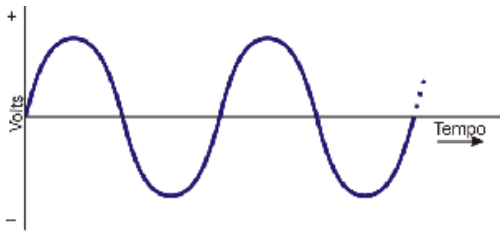


Figura 2 Exemplo de sinal analógico[1]

Os sinais quasi-digitais caracterizam-se por explorarem alguns parâmetros com variação contínua ao longo dos seus valores limites. Com apenas um único sinal, torna-se assim possível obter mais do que uma infinidade de valores, propriedade esta que é denominada de multiparametricidade. Um exemplo prático de utilização destes sinais é a técnica de modulação de largura de impulso (PWM). A técnica de PWM tem como parâmetro o *duty cicle*, i.e.,é a razão entre o tempo em que o sinal se encontra no estado High (i.e. t na Figura 3) e o período (i.e. T na Figura 3) do sinal. Estes sinais são muito utilizados para controlo contínuo de grandezas e.g velocidade de motores de corrente contínua. Na Figura 3 encontra-se representado o exemplo de utilização de um sinal quasi-digital, utilizando a técnica de

PWM. Tal como pode ser analisado através da figura, o sinal utilizado é digital, no entanto comporta-se como sendo um analógico devido ao parâmetro duty-cycle.

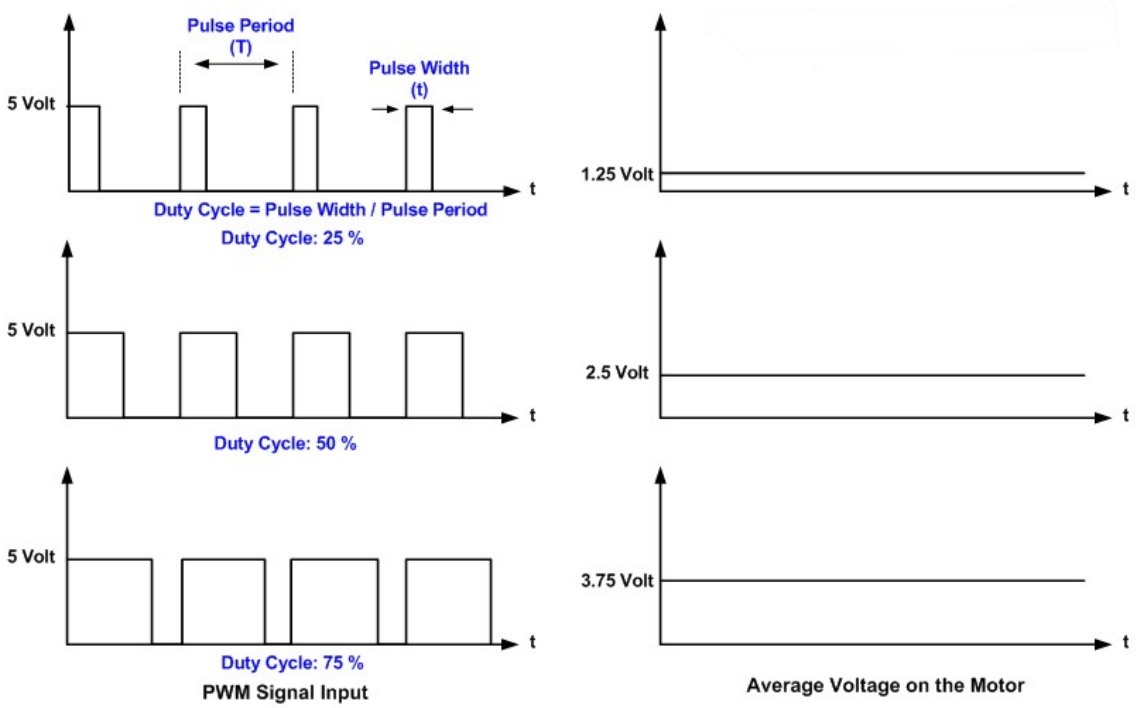


Figura 3 Exemplo de um sinal de PWM [2].

O exemplo aqui apresentado para o sinal *quasi-digital* é o parâmetro *duty-cycle*, no entanto podem ser utilizados outros parâmetros tais como a frequência. É possível converter um sinal analógico em tensão para um sinal com variação do parâmetro de frequência. Estes podem corresponder a uma infinidade de valores na entrada (tensão) convertendo o sinal para uma infinidades de valores na sua saída (frequência).

Analisando os dois parâmetros *duty cycle* e a frequência é possível criar um sistema capaz de adquirir dois sinais analógicos de entrada (tensão V1 e V2, por exemplo) e obter uma correspondência na sua saída de duas grandezas (*duty cycle* e frequência). Uma vez que o *duty cycle* corresponde à relação entre o tempo em que o sinal está em High relativamente ao período do sinal e a frequência é inversamente proporcional ao período, ambas se relacionam podendo junta-las num único sinal. Desta forma, será possível obter um sistema, capaz de controlar dois parâmetros (*duty cycle* e frequência), gerando apenas um sinal que contem as duas informações.

Esta é uma grande vantagem da utilização dos sinais quasi-digitais, permitindo codificar mais do que uma infinidade de valores (face aos sinais analógicos). Adicionalmente, estes sinais possuem características digitais e, por isso, são mais imunes ao ruído do que os sinais analógicos.

Face às vantagens que os sinais quasi-digitais têm face aos restantes sinais estes ainda são pouco utilizados. Segundo um estudo realizado em 2006 pela Associação internacional de sensores de frequência (IFSA) [3] os sinais mais utilizados em sensores são os analógicos, tal como ilustrado na Figura 4.

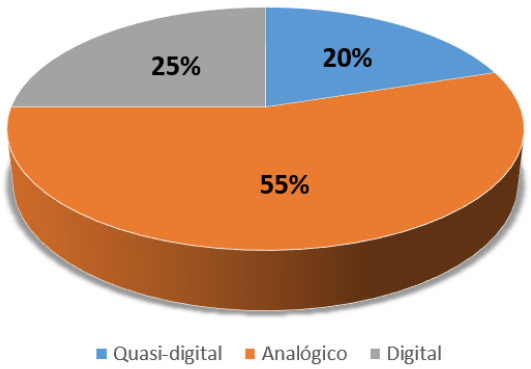


Figura 4 Classificação dos sinais de saídas dos sensores [3]

2.2. TIPOS DE CIRCUITOS

Tal como explicado anteriormente, para se obter informação útil a partir de um sinal é necessário processá-los. Para isso é necessário que estes passem por circuitos capazes de realizar esse processamento de sinal. Os circuitos existentes são os seguintes:

- Circuitos digitais
- Circuitos analógicos
- Circuitos mistos

Um circuito digital, tal como o nome indica, é utilizado para processar sinais puramente digitais, sinais que são discretos no tempo e na amplitude. Um exemplo de circuitos digitais

é a utilização de portas lógicas (tal como a porta lógica AND), contadores binários (tais como os *flip-flops*), microprocessadores, entre outros.

Um circuito analógico realiza o processamento de sinais analógicos. Um exemplo de componentes elétricos maioritariamente utilizados em circuitos analógicos são as resistências, os condensadores, os díodos, os transístores os Amplificadores operacionais (Ampops), os filtros, entre outros.

Os circuitos mistos são utilizados para processar sinais digitais e analógicos que são capazes de interligar-se. Os componentes utilizados nos circuitos mistos podem ser os multiplexeres analógicos (MUX), os conversores analógicos para digital (ADC), os Samples and Hold (S&H), entre outros.

2.3. CICLO DE VIDA DO CIRCUITO

Para o desenvolvimento de um circuito, seja digital, analógico ou misto, frequentemente passam por um processo denominado de ciclo de vida do circuito que é iniciando no conceito do projeto até que seja transformado no produto final. Na Figura 5 encontra-se ilustrado o ciclo de vida ideal para o desenvolvimento de um circuito.

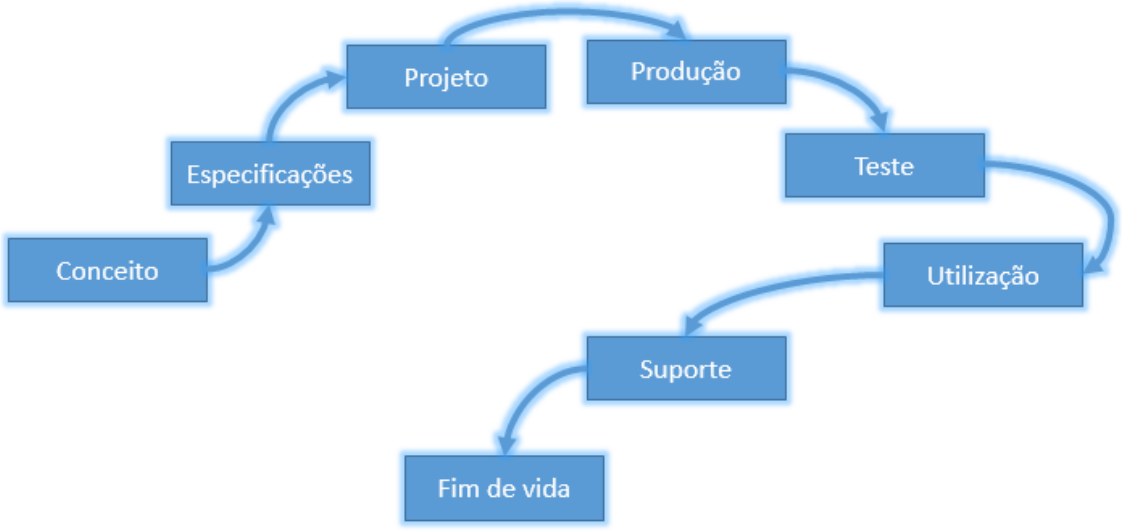


Figura 5 Ciclo de vida ideal de um circuito [4]

A primeira etapa passa por reunir os *conceitos* base do circuito a desenvolver. Nesta etapa a ideia do circuito surge face às necessidades do utilizador, e são aqui que normalmente são analisados alguns dos principais problemas para o desenvolvimento do mesmo.

Após a análise concetual são analisadas as *especificações* técnicas, e definidas as características é que o circuito deve ter face às necessidades do utilizador.

A etapa seguinte é o desenvolvimento do *projeto*, onde terá lugar a implementação dos circuitos de modo a responder a cada uma das especificações analisadas. Durante o decorrer do desenvolvimento poderá haver necessidade de rever as *especificações* e altera-las (e.g. se for determinado que a ideia inicial é impossível de se realizar). Habitualmente é nesta fase que se dá maior enfase aos mecanismos de apoio teste e depuração dos circuitos, para que possa ser feita uma validação funcional do sistema. No desenvolvimento de um circuito utilizado para apenas um equipamento, frequentemente não se utilizam estes tipos de mecanismos, uma vez que são produtos que não constituem esse requisito. No entanto para *produção* de um circuito que será para introduzir em vários equipamentos (uma série de telemóveis por exemplo), torna-se importante introduzir os mecanismos necessários de apoio à realização de teste e a depuração dos circuitos desenvolvidos. Para além destes mecanismos, na fase de projeto é importante desenvolver um modelo de simulação para validação das especificações, permitindo desta forma analisar o sistema antes de o desenvolver fisicamente.

Após o *projeto* estar definido passa-se à fase de *produção*, em caso de necessidade é realizado inicialmente um protótipo do circuito projetado de modo a validar que a estrutura desenhada durante o projeto corresponde às necessidades definidas nas especificações. Na fase de produção (esta fase só existe, se for desenhado um circuito, que é utilizado para produtos desenvolvidos em série) são desenvolvidos vários produtos iguais de acordo com o circuito projetado. Durante a produção existirão diversos defeitos, e.g defeitos nos componentes utilizados para o circuito, componentes mal soldados, componentes mal colocados, entre outros. Sendo esta uma das fases que mais pode trazer problemas para os clientes finais, é importante que sejam realizados testes aos produtos, de modo a garantir que o número de componentes avariados que é enviado ao cliente é aceitavelmente baixa (aceitável porque se trata de uma relação contratual e aceito por ambas as partes).

A fase de *teste* é tipicamente, a que fica mais cara para o produto, uma vez que estes não adquirem nenhuma característica nova decorrente daquela. O objetivo desta fase é realizar os testes principais e mais críticos do circuito. Não é frequentemente possível, realizar os testes a todos os tipos de falhas uma vez que esse tipo de testes pode não ser possível e, mesmo que possível, seria tão caro que inviabilizada o produto.

Uma vez o circuito testado e, frequentemente, com a geração do relatório estrutural, o circuito será fornecido ao cliente para que utilize o circuito conforme as suas necessidades.

Durante a *utilização* normal do circuito, o cliente poderá detetar alguns erros, uma vez que na fase de *testes*, alguns dos componentes defeituosos acabam por passar sem serem detetados, ou poderá ter necessidade de alteração de alguma característica no produto, sendo esse o momento em que o circuito irá dar início à fase de *suporte*.

Durante a fase de *teste*, muitos dos circuitos poderão apresentar erros, podendo ser necessário voltar a fase de *projeto* para análise do problema. O mesmo acontece na fase de *suporte*, sempre que um utilizador deteta um erro no produto, é necessário voltar a fase do *projeto* e no pior caso às *especificações* do produto, de forma a resolver o problema do utilizador. Nestes casos, torna-se necessário realizar a depuração de circuitos, no entanto o foco principal de depuração do circuito encontra-se na fase de projeto.

Após várias utilizações do equipamento e com o passar do tempo o circuito começa a ficar sem utilidade para o utilizador chegando ao *fim do ciclo de vida* do circuito.

2.4. TESTE E DEPURAÇÃO DE CIRCUITOS

A fase de testes, tal como apresentado no subcapítulo 2.3, é o ultimo passo a ser executado antes de ser entregue ao utilizador final. Esta fase tem como finalidade verificar se o produto se encontra apto ou não para ser entregue ao cliente. Após se considerar uma falha de um produto é necessário perceber o que levou ao problema, é neste momento que é aplicada a depuração de circuitos. Depuração é a atividade de determinar a natureza de um erro e de o remover. Nos circuitos existem dois tipos de erros:

- Erro de desenho
- Erro de fabricação

Para ultrapassar os erros de desenho torna-se essencial utilizar *softwares* de simulação para procurar garantir que os circuitos desenhados irão estar de acordo com as características esperadas. Os erros de fabricação poderão acontecer devido a distração dos funcionários, má codificação dos robots utilizados para realizar a montagem dos circuitos, entre outros fenómenos. Para realizar a *deteção* dos erros são utilizados diversos equipamentos (osciloscópios, multímetros, etc) e diversas metodologias tais como [19][20][21][22]:

- Inspeção ótica automática (AOI)
- Inspeção de raio X
- *In-Circuit Test* (ICT)
- *Burn-in* sendo esta uma técnica de deteção e eliminação no mesmo processo
- Teste funcional

O teste de inspeção ótica automática é útil para garantir que o posicionamento e a solda dos componentes está correto. Tal como ilustrado na Figura 6 são utilizados equipamentos automáticos que possuem câmaras de alta resolução e várias fontes de luz para análise de circuitos, e utilizam *software* capaz de analisar os problemas e identifica-los. Por norma estes *softwares* são capazes de realizar leituras de códigos de resistências através do reconhecimento ótico de caracteres (OCR).

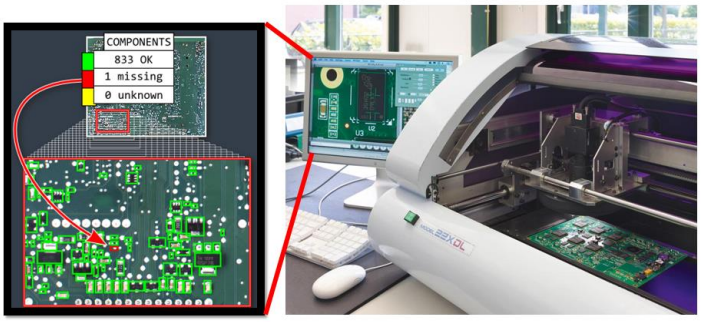


Figura 6 Resultados da inspeção ótica automática (AOI) [5]

Para análise dos circuitos, utilizando inspeção ótica, apenas existe a possibilidade de analisar circuitos integrados e soldas que estão visíveis na área da câmara, no entanto existem circuitos integrados que possuem encapsulamentos onde os *pads* que entram em contacto com a placa de circuito impresso (PCB) são no seu interior, tal como os encapsulamentos *Ball Grid Array* (BGA) e *Quad Flat No leads* (QFN) que impossibilitam realizar a análise do seu interior [6]. Na Figura 7 encontra-se ilustrado um exemplo de encapsulamento BGA. De modo a ultrapassar este problema foram criados sistemas de teste capazes de analisar através de raio x o interior dos componentes, permitindo uma análise mais detalhada de curto circuitos, componentes mal soldados, etc.

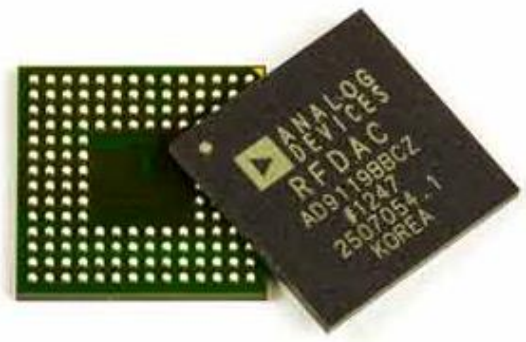


Figura 7 Exemplo de integrado com encapsulamento BGA. [7]

Os sistemas de raio x possuem um emissor de raios x e um recetor que absorve a radiação emitida. Os raios x ao serem penetradas nos circuitos integrados e nos restantes componentes, absorvem quantidades de radiação diferentes dependendo das densidades dos mesmos. Utilizando este método é possível analisar pistas quebradas ou pinos de circuitos inexistentes assim como falhas existentes nas soldaduras [8], tal como ilustrado na Figura 8. Um dos grandes problemas deste tipo de sistema é o de que não existe a possibilidade de

analisar se a polaridade e se o valor dos integrados são os esperados, sendo apenas possível analisar o padrão dos componentes.

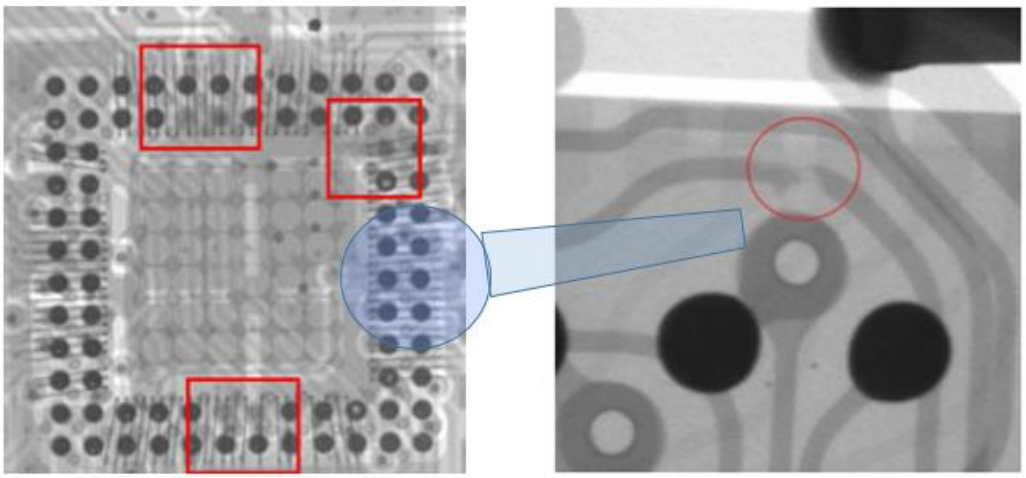


Figura 8 Inspeção de circuitos utilizando raio x [8] [9]

O método de ICT é utilizado para realizar testes elétricos ao nível dos componentes e das pistas, este tipo de teste é realizado com o objetivo de verificar se o desenvolvimento se encontra de acordo o esquema (o projetado) [10]. Por regra, são realizadas medições de resistências e continuidades assim como são introduzidos estímulos na placa para verificação de valores de tolerâncias em componentes analógicos e digitais. De forma a poupar tempo durante a produção, por vezes são realizadas outras operações, tais como, medições de valores de frequências, programações ou configurações dos dispositivos assim como verificar a funcionalidade dos mesmos. Por regra para realizar um teste pelo método de ICT utilizam-se dois tipos de máquinas, a *bed-of-nails* e o *flying probe*. As máquinas de *bed-of-nails* consistem numa matriz de agulhas distribuídas em pontos específicos (pontos estes que já são desenhados no projeto) que entrarão em contacto com as placas elétricas para realizar os testes aos componentes existentes na placa. Na Figura 9 encontra-se um exemplo de utilização de *bed-of-nails* para o teste de PCBs, sendo possível observar que para este tipo de testes as PCBs já devem conter vários de pontos de acesso para o teste.

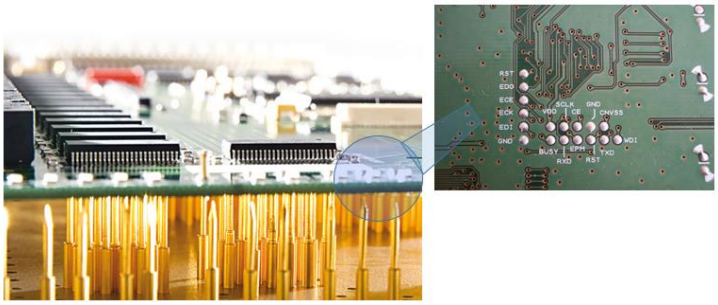


Figura 9 Método de ICT com *bed-of-nails* [11] [12]

A utilização de máquinas deste tipo, torna-se vantajoso pois permitem realizar testes bastante rápidos, no entanto este tipo de sistemas é pouco modular, não podendo ser facilmente alterado ao longo do tempo. De modo a ter a possibilidade de realizar testes a diversos tipos de placas, utilizando a mesma máquina, é utilizado outro tipo de equipamentos para o método de ICT.

O *Flying probe* são cabeças de testes que se deslocam ao longo da placa de circuito impresso para entrar em contacto direto com os componentes ou com pontos de teste para realizar o teste ao circuito. Uma vez que é necessário realizar o deslocamento das cabeças de agulhas durante o teste este tipo de testes por regra é mais lento que os anteriores. Por vezes durante o desenho das PCBs não são considerados pontos de teste para alguns dos integrados, sendo necessário aceder diretamente aos pinos do integrado, na Figura 10 é possível observar que este tipo de máquinas são muito precisas, quando se pretende testar integrados que possuam os seus pads muito próximos uns dos outros.

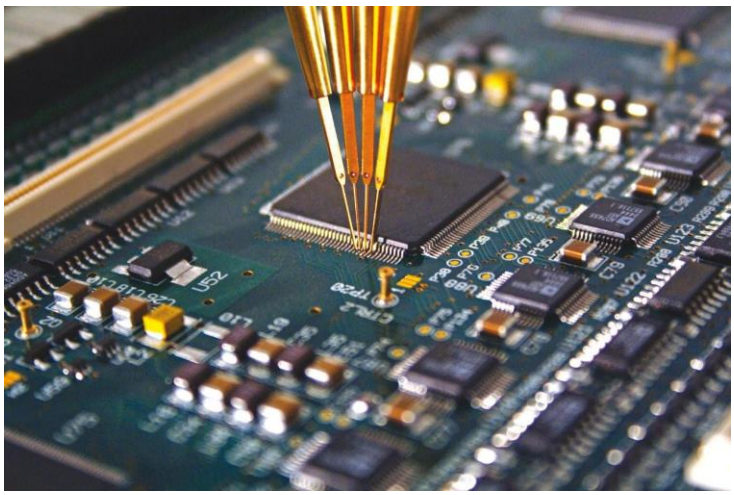


Figura 10 Método de ICT com *flying probes* [13]

Com o decorrer dos anos e com a evolução dos componentes eletrónicos, estes tenderam a ficar mais pequenos fazendo com que os circuitos criados tendessem a ficar mais pequenos também. Deste modo algumas das ligações existentes nos circuitos ficam inacessíveis para se poder realizar o teste utilizando os métodos anteriormente citados.

A técnicade *Burn-in* permite que se realize uma verificação funcional dos circuitos quando submetidos em condições de limites extremos dos componentes. Um exemplo de utilização deste método passa por colocar o circuito em funcionamento, em condições de temperaturas elevadas ou bastante baixas durante longos espaços temporais, de forma a validar que estes irão funcionar nas condições submetidas. Este método nem sempre é aplicado nos circuitos desenvolvidos para deteção de erros, visto que é um método habitualmente moroso e que irá evidenciar um aumento do número de rejeições dos produtos. Na Figura 11 encontra-se um exemplo de teste *Burn-in* para verificar o tempo de vida útil de uma série de lâmpadas *LED* em condições de temperatura modificadas.

Figura 11 Método *Burn-In* de lâmpadas *LED* [14]

O teste funcional dos circuitos é realizado para procurar garantir que as funcionalidades para o qual o circuito foi desenhado, se encontram dentro do esperado. Este teste determina se os circuitos são aprovados ou rejeitados para serem fornecidos aos clientes finais. Os requisitos de um teste funcional variam bastante de circuito para circuito pois dependem das funcionalidades para o qual foram desenhados. Regra geral, estes testes são realizados através de interfaces com os circuitos simulando o ambiente elétrico final para o qual o circuito irá ser utilizado. Os circuitos são submetidos a diversos sinais elétricos e são monitorizadas as ações em alguns dos pontos específicos. Um exemplo prático deste tipo de testes poderá ser testar que num circuito o botão *Retornar* é detetado, que o envio de um

comando específico atualiza o display com os símbolos esperados, que o sinal do *clock* está correto, entre outros exemplos.

Tal como indicado anteriormente, a evolução dos equipamentos está a torná-los cada vez mais pequenos desenvolvendo novas funcionalidades. Antigamente um telemóvel apenas servia para realizar chamas, na atualidade estes equipamentos são capazes de funcionarem como câmaras fotográficas, como rádio, como mp3 entre outras funcionalidade, na Figura 12 encontra-se ilustradas as diversas funcionalidades que os telemóveis da atualidade possuem.

Figura 12 Funcionalidades de um telemóvel [15]

Estas novas tecnologias só são possíveis devido à evolução dos componentes elétricos. Os circuitos integrados cada vez ficam mais pequenos, e com maiores capacidades. Na Figura 13 é apresentada a evolução dos encapsulamentos dos circuitos integrados relativamente ao decorrer dos anos, sendo que foram iniciados com os encapsulamentos DIP (Dual In-line Package), passando pelos QFP (Quad Flat Package), BGA , CSP (Chip Scale Package), SIP (System-in-Package) e pelo WLP (Wafer-Level Package). Um dos grandes problemas com o aparecimento destes integrados passa pela dificuldade que se sente para aceder aos seus pinos, levando a que a realização do projeto se torne demasiado dispendioso para que os processos de teste e de depuração do circuito seja realizado. Perante esta situação, em meados de 1980 um grupo de engenheiros reuniram-se para examinar o problema e definir possíveis soluções. O nome do grupo de engenheiros ficou inicialmente denominado por

Joint European Test Action Group (JETAG), uma vez que reunia apenas engenheiros da Europa. A solução apresentada permitiu ultrapassar o problema de incapacidade de acesso direto aos pinos, fazendo com que fosse possível testar o circuito utilizando a arquitetura *Boundary-Scan*. Mais tarde, juntaram-se ao grupo representantes de empresas do norte da América pelo que o nome inicial do grupo foi alterado para *Joint Test Action Group* (JTAG). Este foi o grupo que desenvolveu as normas que respondem aos problemas anteriormente explicados, utilizando as infraestruturas IEEE1149.1 e IEEE1149.4. [17]

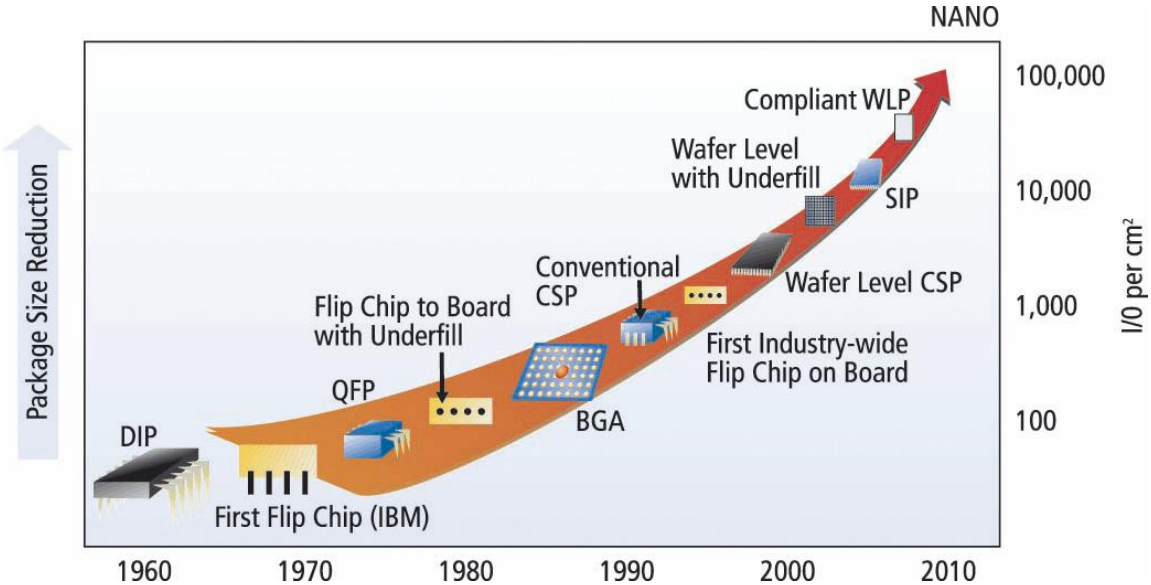


Figura 13 Evolução dos encapsulamentos dos circuitos integrados [18]

De forma a tornar os dispositivos cada vez mais versáteis, tem-se tornado notório o desenvolvimento de circuitos flexíveis, i.e. dispositivos que permitem que sejam facilmente modificados (sem necessidade de alteração de *hardware*), para desenvolver a sua função. Inicialmente estes dispositivos tiveram maior relevância no domínio digital, com o aparecimento de dispositivos denominados de *Field-programmable gate arrays* (FPGA's). Perante a necessidade de utilização deste tipo de circuitos no domínio analógico, foram desenvolvidas as *Field programmable analog arrays* (FPAA's).

3. ESTADO DA ARTE

Na secção aqui presente, irá ser descrito duas das infraestruturas existentes para realizar testes e apoiar na depuração de circuitos, sendo estes mecanismos normalizados em IEEE1149.1 e IEEE1149.4 que já foram referidos no subcapítulo 2 (Conceitos fundamentais). Este trabalho desenvolvido tem como objetivo desenvolver um circuito de demonstração de FPAA que incorpora a utilização da infraestrutura IEEE1149.4, no entanto a norma IEEE1149.1 deverá ser descrita uma vez que é uma a estrutura base da infraestrutura IEEE1149.4. Para além da explicação global de implementação destas infraestruturas, também será apresentado um dispositivo que implementa a infraestrutura IEEE1149.4 e uma ferramenta educacional para apoio ao teste de dispositivos com infraestruturas IEEE1149.1. Não sendo apenas estas duas infraestruturas o foco principal do trabalho, também será dado a conhecer os dispositivos lógicos configuráveis assim como o funcionamento principal de um dispositivo analógico programável (FPAA).

3.1. INFRAESTRUTURA IEEE1149.1

Toda a informação presente neste subcapítulo teve como base o documento [23]. A infraestrutura IEEE1149.1 foi desenvolvida com o objetivo principal de realizar testes a circuitos digitais. Embora esta infraestrutura não esteja diretamente relacionada com o trabalho a ser desenvolvido, esta será aqui descrita uma vez que a infraestrutura que será utilizada, a IEEE1149.4, se apresenta formalmente como extensão da IEEE1149.1. Esta infraestrutura foi desenvolvida para apoiar no teste e na depuração de circuitos de modo a permitir uma observação, controlo e verificação dos circuitos reduzindo os acessos físicos necessários. Esta norma é aplicada meramente em circuitos digitais e a sua utilização em circuitos integrados permite:

- Testar a interligação entre os circuitos integrados após serem integrados numa placa de circuito impresso (PCB).
- Testar a funcionalidade para o qual foi desenvolvido

- Observar ou modificar a atividade do circuito durante a operação normal do componente.

O desenvolvimento desta norma procura fornecer uma solução para a dificuldade apresentada de acesso aos componentes que se encontram muito juntos ou que possuem encapsulamentos com acesso restrito para a realização de testes em placas de circuito impresso (PCB's). A infraestrutura IEEE1149.1 tem por base a técnica *boundary-scan* que consiste na utilização de *virtual nails* (tal como no método de teste de *bed of nails* já apresentado anteriormente) para que seja possível observar e controlar cada pino da parte lógica (*core*) do circuito integrado. Durante o funcionamento normal do circuito integrado é possível analisar os sinais que estão a ser injetados tanto pelo sistema *core* como por sinais externos. As *virtual nails* são interligadas entre a parte lógica do circuito integrado e os pinos externos, sendo denominadas por *Boundary Scan Cells* (BSC's). As BSC's, tal como o nome indica, são células que possuem um comportamento específico com a capacidade de capturar sinais e desloca-los em série para pinos exteriores de forma a realizar a comparação com os resultados esperados.

Tal como se pode analisar na Figura 14, é utilizado um controlador de modo a ser possível especificar o comportamento das BSC's tendo em conta que o circuito deverá ser observado e/ou controlado sem que haja interferências diretas no sistema.

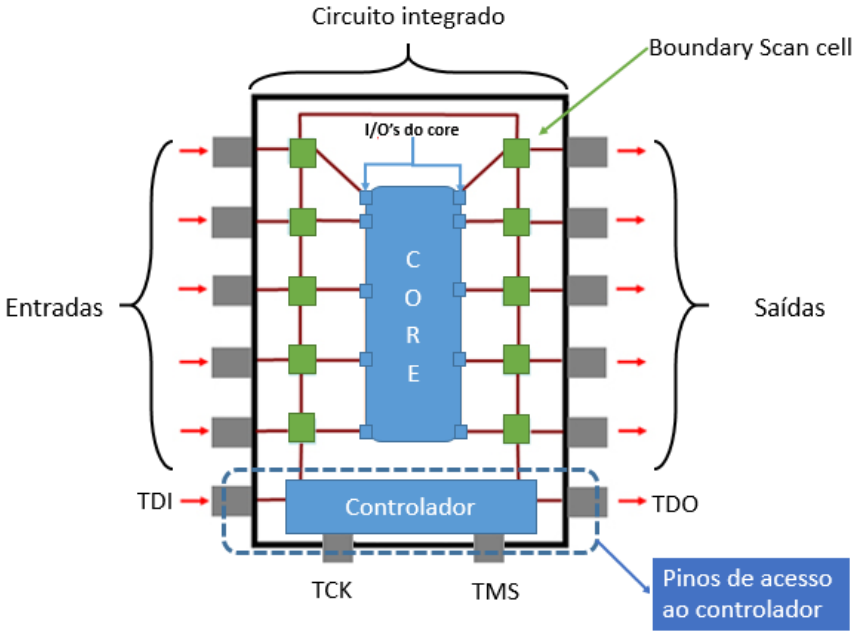


Figura 14 Infraestrutura IEEE1149.1 aplicada num circuito integrado

A norma define, que para ser possível ter acesso ao controlador, são necessários no mínimo quatro pinos, três de entrada e um de saída. A norma possibilita a utilização de um quinto pino opcional de entrada, para realização de reset da lógica de teste. Este grupo de pinos é denominados de *Test Acess Port* (TAP) e é constituído pelo:

- *Test Clock Input* (TCK)
- *Test Mode Select Input* (TMS)
- *Test Data Input* (TDI)
- *Test Data Output* (TDO)
- *Test Reset Input* (TRST) (opcional)

Para que o circuito integrado esteja de acordo com a norma, qualquer pino utilizado para comunicação com o controlador deverá ser dedicado a essa funcionalidade, não podendo ser utilizado para qualquer outra finalidade.

O *Test Clock Input (TCK)* é utilizado para gerar o sinal de relógio para ser utilizado na logica de teste, sempre que este sinal se encontra no estado lógico 0 os dispositivos contidos na logica de teste mantêm o estado indefinidamente. A utilização de um sinal de relógio independente, garante que os dados do teste podem ser movidos para o circuito, ou do circuito, sem alterar o estado lógico do sistema.

O sinal recebido no *Test Mode Select Input (TMS)* é descodificado através do controlador para que possa controlar as operações do teste. O sinal presente no TMS deve ser amostrado pela logica de teste apenas no flanco ascendente do sinal TCK permitindo realizar o controla da máquina de estados do controlador. De modo a respeitar a norma é necessário que este seja ligado sempre no estado lógico 1, sempre que o circuito se encontra deligado do circuito externo. Para os circuitos que utilizam níveis de tensão Transistor-Transistor Logic (TTL) basta utilizar uma resistência de pull-up na entrada TMS do componente.

Os dados e as instruções de teste serie são recebidos pelo *Test data input (TDI)*. O sinal presente no TDI deve ser amostrado no flanco ascendente do sinal TCK, tal como o pino TMS, este deve estar no estado logico 1 quando se encontra desligado do circuito externo. Quando os dados são movidos do TDI para o TDO, os dados de teste recebidos no TDI

devem aparecer sem inversão no TDO após um número determinado de flancos ascendentes e descendentes do TCK, número este que é definido pelo comprimento da instrução ou do registo de teste selecionado. Os pinos TDI e TDO permite um movimento série dos dados em teste através de todo o circuito. O requisito para os dados serem propagados do TDI para o TDO sem inversão é incluído para simplificar a operação da compatibilidade dos componentes sem ligação a esta norma que estão contidos na placa de circuito impresso.

O pino *Test data output (TDO)* é utilizado para obtenção dos dados série para as instruções do teste e dados da logica de teste definidos na norma. A alteração do estado do sinal apresentado no pino TDO deverá ocorrer apenas no flanco descendente do sinal TCK. Este pino deverá estar em modo inativo, exceto quando os dados se encontram em progresso, para garantir este estado, utilizando os níveis de sinal TTL ou *complementary metal-oxide-semiconductor* (CMOS) é necessário a utilização de um buffer de saída em alta impedância.

A entrada opcional *Test reset input (TRST)* permite uma inicialização ao controlador de forma assíncrona. O *reset* é ativado quando é aplicado o estado lógico 0 no pino TRST, quando o circuito externo se encontra desligado do mesmo deverá estar o estado lógico 1. O TRST não deve ser utilizado para inicializar qualquer sistema lógico no interior do componente.

Tal como pode ser analisado na Figura 15 as BSC's possuem 4 canais principais que contêm informação útil para o teste e 4 canais de configuração das BSC's. As BSC's possuem quatro modos de operação: Modo de operação normal, controlo, observação e deslocamento.

Durante a operação normal, os dados que chegam ao canal *Signal In* são passados diretamente para o canal *Signal Out*.

Quando este se encontra em modo de operação de controlo, o conteúdo da célula *Update Hold Cell* passa para o canal *Signal out*.

Durante o modo de observação o sinal que se encontra no canal *Signal In* é passado para a entrada da célula *Capture Scan* e o valor é capturado pelo próximo *ClockDR*, este sinal deriva do sinal TCK.

Durante o modo de deslocamento, o sinal contido na célula *Capture scan* é enviado para o canal *Scan Out* podendo ser transmitido para o *Scan In* de uma BSC seguinte.

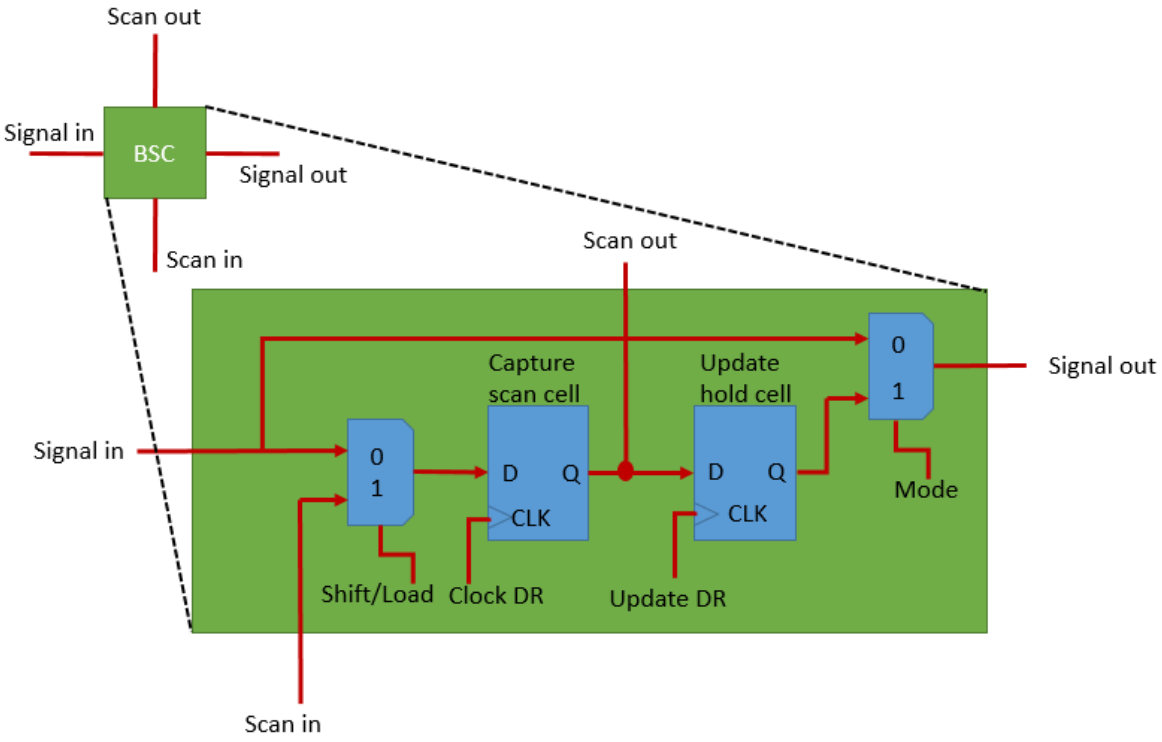


Figura 15 Estrutura de uma BSC

É possível então verificar que os modos de operação de captura e de envio de dados série não interferem com a operação normal permitindo passar os dados que se encontram no canal *Signal In* para o *Signal Out*, podendo observar o seu estado. Isto permite a captura do sinal em tempo real, dos valores operacionais e enviando-os para inspecionar o estado desse sinal sem interferência com o restante.

A infraestrutura IEEE1149.1 prevê a interligação entre circuitos integrados que utilizem a mesma infraestrutura e o dispositivo que controla o barramento de teste ao nível da placa é referido como o *master* do barramento. As interligações possíveis são as seguintes:

- Ligação serie
- Ligação hibrida (duas cadeias série em paralelo)
- Ligação múltipla independente

Na Figura 16 encontra-se ilustrada a interligação série entre vários integrados conforme a norma ieee1149.1. Esta interligação pressupõe que os sinais TMS e TCK são partilhados entre todos os integrados e que são fornecidos pelo *master* do barramento para os *slaves* que

se encontram ligados em paralelo. O caminho série criado através de *daisy-chain* através dos pinos de teste (TDI e TDO).

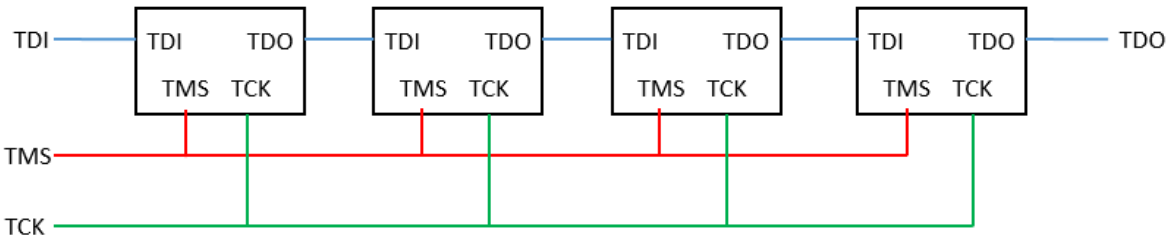


Figura 16 Ligação série

Na Figura 17 encontra-se ilustrada a ligação hibrida, este tipo de ligações consiste em ligar dois módulos em paralelo de dois integrados em série. Uma vez que os sinais TDI e TDO são partilhados pelos dois módulos, torna-se necessário utilizar dois sinais TMS (TMS1 e TMS2) para garantir que apenas um caminho série é analisado de cada vez. Esta configuração utiliza o terceiro estado num dos pinos TDO, garantindo que apenas um dos módulos se encontra a enviar informação enquanto o outro módulo se encontra no estado de alta impedância. O sinal TCK é partilhado entre os dois módulos.

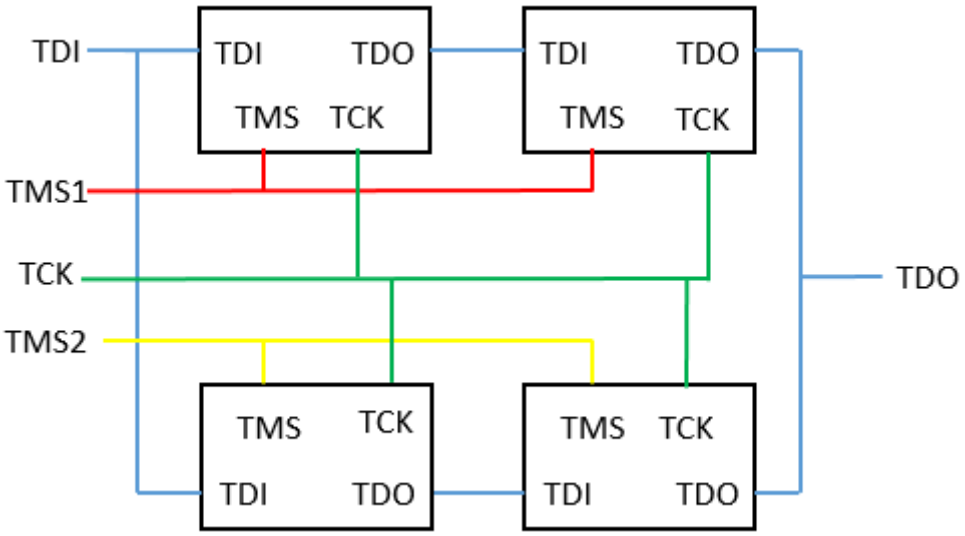


Figura 17 Ligação de duas cadeias série em paralelo

A ligação múltipla independente encontra-se ilustrada na Figura 18, e é caracterizada por ter caminhos série completamente independentes entre os diferentes circuitos integrados. Uma

vez que os sinais de TDI e TDO são independentes, os sinais TCK e TMS são ligados de forma comum entre os circuitos integrados podendo controla-los a todos ao mesmo tempo.

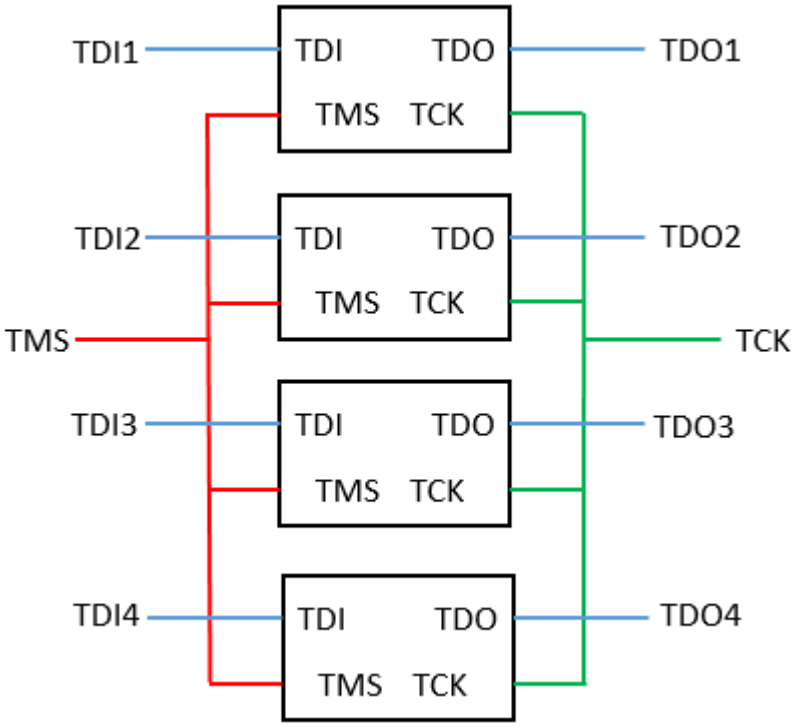


Figura 18 Ligação múltipla independente

A norma estabelece regras para os circuitos integrados que pretendam realizar operações de teste de acordo com esta norma, para isso é determinado que a arquitetura lógica do teste terá de possuir vários elementos:

- Porto de acesso ao teste (Test Acess Port -TAP);
- Controlador TAP;
- Um registo de instruções;
- Um grupo de registos de dados de teste.

Tal como pode ser verificado na Figura 19 a arquitetura lógica de teste utiliza 4 pinos dedicados (TDI, TDO, TMS e TCK) que fazem parte do Test Acess Port (TAP). A definição do funcionamento do teste é garantido através do controlador TAP, este recebe o sinal de TCK e interpreta o sinal TMS para alterar a máquina de estados interna do controlador. O registo de instruções é utilizado para selecionar o teste a ser realizado ou para aceder aos

registos de dados a ser acedido. O grupo de registos de dados de teste deve incluir o *bypass* e o registo *boundary-scan*, opcionalmente poderá conter também o registo identificador do dispositivo e outros registos de dados de teste. As BSC's que se encontram interligadas entre si através de ligação série e aos pinos externos do circuito integrado formam o registo *boundary-scan* (BSR). A seleção entre o caminho alternativo entre o registo de instruções e o registo de dados de teste para os sinais TDI e TDO deve ser realizado através do controlo do controlador TAP.

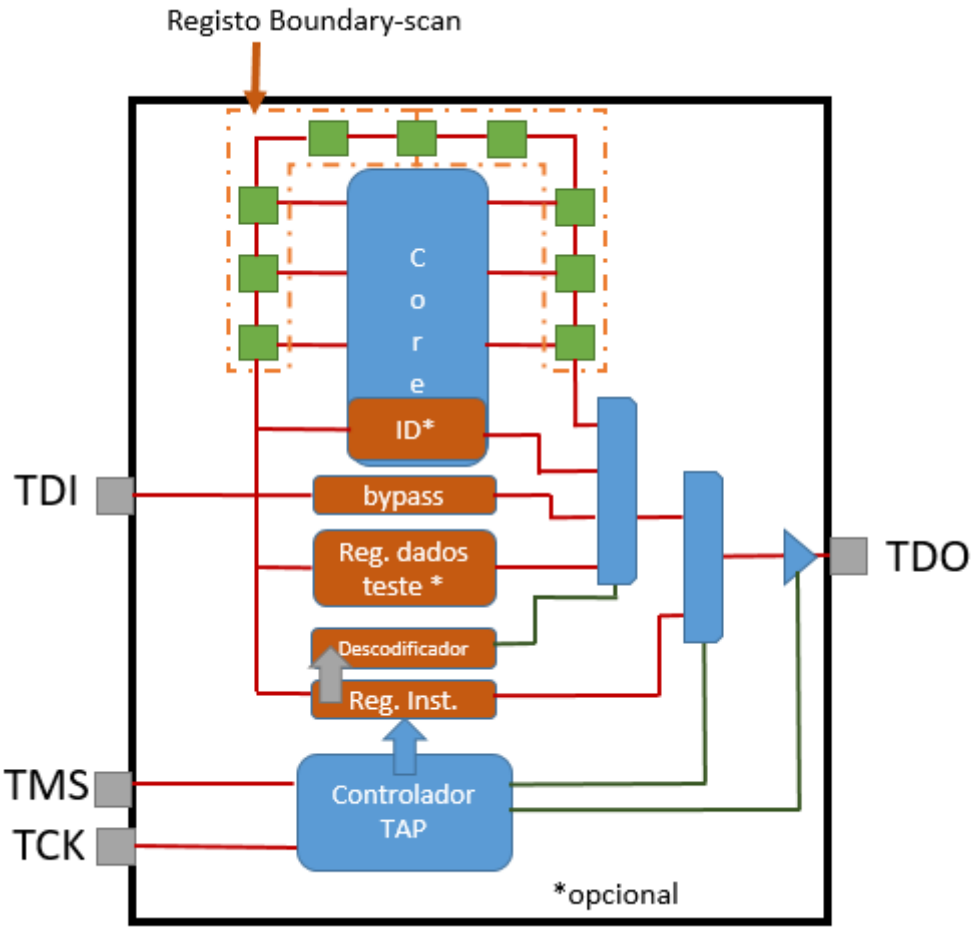


Figura 19 Arquitetura lógica de teste [24]

O controlador TAP é definida como uma máquina de estados finita e síncrona que altera o seu estado de acordo com as alterações dos sinais TMS e TCK, controlando assim a sequência de operação do circuito.

A máquina de estados do controlador TAP encontra-se representada na Figura 20, onde o valor apresentado junto a cada transição do estado representa o sinal presente no TMS, quando existe uma transição ascendente no sinal TCK.

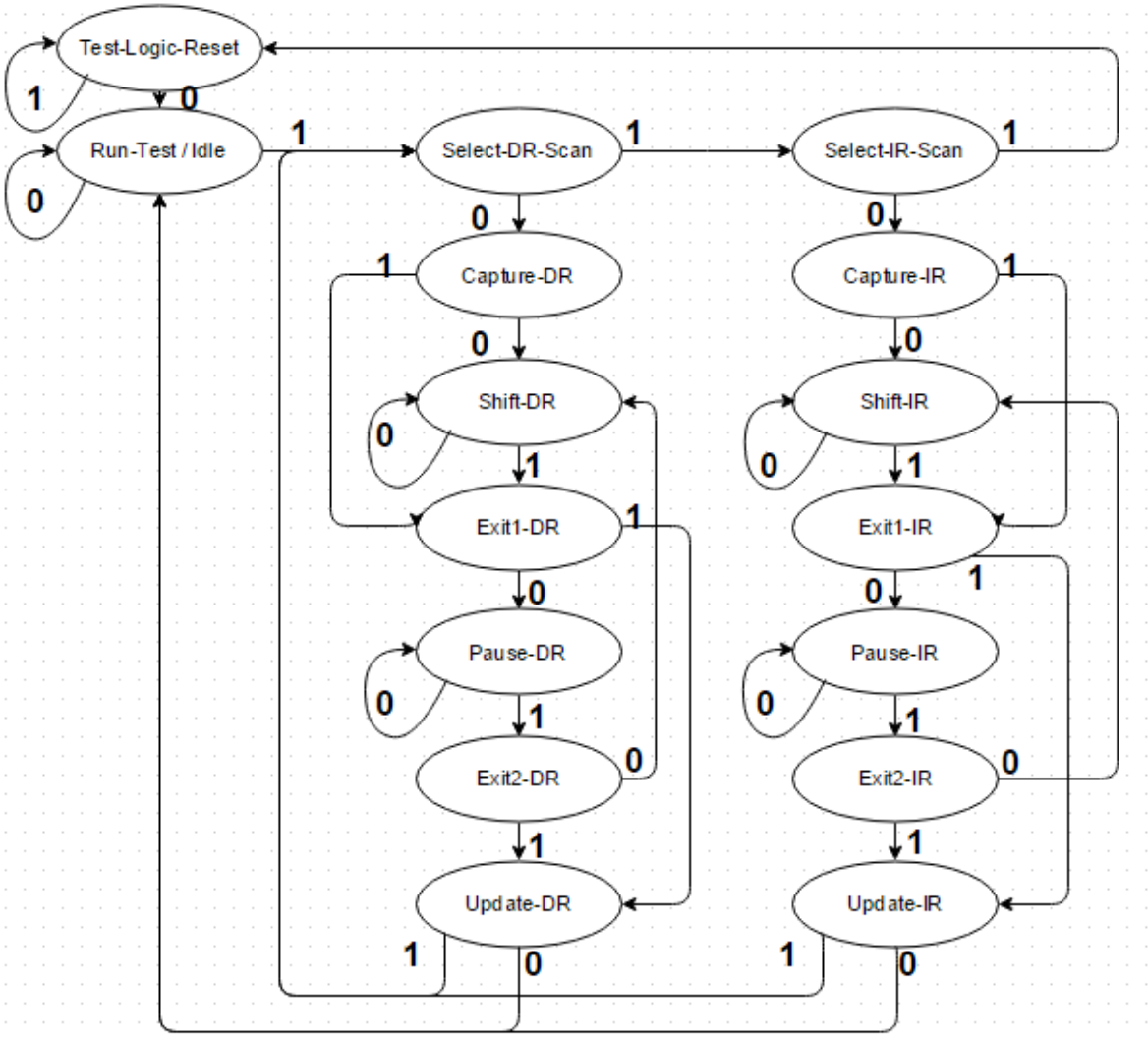


Figura 20 Máquina de estados do controlador TAP

O estado de transição tal como indicado anteriormente acontece sempre quando o sinal TCK possui uma transição ascendente, e a ação lógica do estado ocorre na transição descendente desse mesmo sinal.[17]

O controlador TAP inicializa no estado *Test-Logic-Reset* sempre que é ligada a alimentação do circuito ou sempre que o sinal TRST (se existir) for forçado com o valor 0. Enquanto o sinal TMS se encontra no estado 1 (estado por defeito devido ao *pull-up* existente) a máquina de estados não avança, sendo que neste estado a logica de teste é desabilitada, fazendo com

que a operação normal do sistema core pode funcionar normalmente. Estando neste estado, a norma indica que o estado que deverá ser carregado para o registo de instruções é o *IDCODE* (código que identifica o circuito), no entanto, tal como se verificou anteriormente, se o registo de identificação do dispositivo não existir, será carregada a instrução *BYPASS*. Se por algum motivo existir uma pequena interferência no sinal de TMS causando um flanco descendente do mesmo, a operação lógica garante que ao fim de três flancos ascendentes do sinal TCK o controlador volta para o estado de *reset* sem interferir com o estado do circuito *core*. Independentemente do estado em que esteja, ao fim de 5 flancos ascendentes do sinal TCK e com o valor TMS sempre ativo, a máquina de estados passa para o estado *Test-Logic-Reset*.

O *Run-Test / Idle* é considerado como sendo um estado entre as operações de análise. Enquanto o sinal TMS se encontrar com o valor 0, o controlador irá manter-se sempre no mesmo estado. Neste estado a lógica de teste não se altera, exceto se algumas instruções forem carregadas, tal como a instrução *RUNBIST*. Esta instrução executa um autoteste à lógica interna do circuito. Qualquer instrução que seja para realizar autotestes sem ser *RUNBIST* deve ser executado enquanto o controlador se encontra neste estado. Para as restantes instruções executadas neste estado, não irão afetar a alteração do registo de dados do teste, mantendo o estado anterior (ficará inativo).

Após o estado *Run-Test / Idle*, poderá ser realizada a escolha entre dois ramos, o ramo que contem o conjunto de operações a realizar no registo de dados de teste e o ramo que contem o conjunto de operações a realizar no registo de instruções. O ramo que realiza operações no registo de dados é iniciado pelo estado *Select-DR-Scan*, que é um estado temporário em que todos os registos de dados de teste selecionados pela instrução atual mantêm o estado anterior, o mesmo acontece para o estado *Select-IR-Scan*.

O controlador ao permanecer no estado *Select-DR-Scan* e ao existir uma transição de TCK ascendente com o sinal TMS a 0, a máquina de estados irá passar para o estado Capture-DR. Neste estado os dados paralelos deverão ser carregados para o registo de dados de teste selecionado pela instrução atual. Se o registo de dados de teste selecionado pela instrução atual não possuir entrada paralela, ou se a captura não for necessária para o teste selecionado, o registo retém o estado anterior inalterado.

No estado *Shift-DR,* o registo de dados de teste interligado entre o TDI e o TDO realiza o deslocamento dos dados série entre eles a cada transição do sinal TCK e enquanto o TMS se encontra com o sinal 0. Os registos de dados de teste que são selecionados pela instrução atual mas não se encontram ligados ao caminho série, mantêm o estado anterior inalterado.

O estado *Exit1-DR* é temporário e poderá ser proveniente do estado *Shift-DR* (no caso do sinal TMS ter ficado a 1 nesse estado) ou do estado Capture-DR (no caso do sinal TMS ter ficado a 0 nesse estado). Se o sinal de TMS estiver no estado 1, o controlador irá passar diretamente para o estado *Update-DR* que terminará o processo de observação dos dados, se o sinal de TMS estiver no estado 0 o controlador irá passar para o estado *Pause-DR*. Considera-se que os registos de dados do teste selecionado pela instrução atual retém o estado anteriormente sem alteração.

Permanecendo no estado *Pause-DR* fará com que o deslocamento dos dados, que se encontram no registo de dados de teste entre o TDI e o TDO, seja temporariamente interrompido. Todos os registos de dados selecionados pela instrução atual, retêm o estado anterior sem alteração.

Tal como o estado *Exit1-DR*, o estado *Exit2-DR* é temporário, no entanto se o sinal de TMS for 0 será dada a ordem para que a máquina de estados passe para o estado *Shift-DR* sem haver necessidade de voltar a carregar os registos de dados paralelos para o registo de dados de teste selecionado pela instrução atual (no estado *Capture-DR*).

No caso do sinal TMS se encontrar com o valor 1 no estado *Exit1-DR*, ou no estado *Exit2-DR* irá ser executado o estado *Update-DR*. Este estado é responsável por atualizar o registo de dados de acordo com os bits inseridos durante o estado de deslocação dos dados, deixando os dados disponíveis nas saídas paralelas.

Os restantes estados que se encontram no ramo *Select-IR-Scan* são idênticos aos explicados anteriormente, no entanto os dados serão carregados para o registo de instruções em vez de ser para os registos de dados.

Tal como apresentado anteriormente a norma prevê a utilização de diversos registos, nomeadamente o registo de instruções e o grupo de registos de dados de teste. O registo de instruções permite que uma instrução seja carregada de forma a selecionar o teste a ser executado ou o registo de dados a ser acedido.

O registo de instruções é baseado num registo de deslocamento, e este tem de ter no mínimo duas células, para que seja capaz de suportar instruções mínimas (O registo de instruções deve permitir selecionar o registo *bypass* e deve permitir aceder ao registo *bondary-scan* nas três configurações mínimas-*EXTEST*, *PRELOAD* e *SAMPLE*). As instruções deslocadas no registo de instruções, devem manter-se inalteradas, permitido a sua alteração apenas quando o controlador se encontra no estado *Update-IR* e no estado *Test-Logic-Reset*. As duas células menos significativas do registo de instruções devem ser carregadas com o valor binário *"01"* para o estado do controlador *Capture-IR*. Este valor binário que a norma obriga a carregar serve para para validar que o controlo do TAP está a ser realizado de forma correta. Na Figura 21 encontra-se ilustrada célula de um registo de instruções em detalhe que opera em resposta aos sinais gerados pelo controlador TAP.

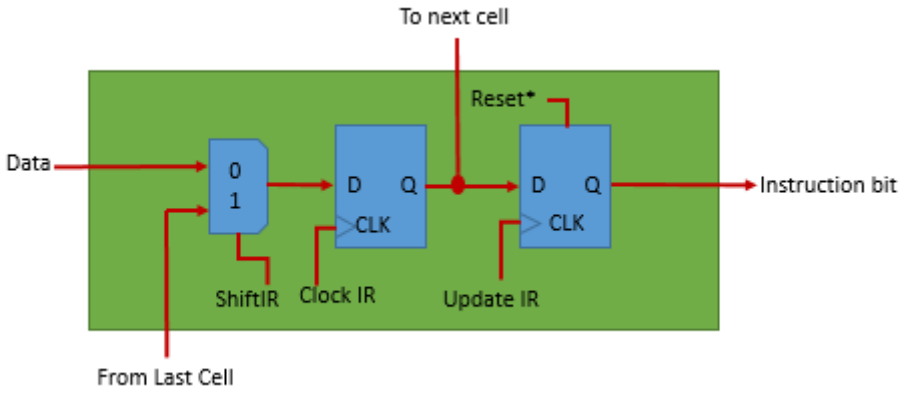


Figura 21 Célula do registo de instruções

A saída paralela (*Instruction bit*) é atualizada quando o controlador se encontra no estado *Update-IR*. Esta saída pode ser reinicializada quando o controlador se encontra no estado *Test-logic-Reset* fazendo com que a entrada *Reset** fique com o estado 0 da célula (este sinal será colocado a 0 após entrar no estado *Test-Logic-Reset* e houver um flanco descendente do sinal TCK). A entrada do relógio (*Clock IR*) para o registo do caminho série é apicado apenas durante os estados do controlador *Capture-IR* e *Shift-IR*.

O registo de instruções permite a utilização de instruções de forma a controlar o teste a ser executado. Cada instrução irá criar um caminho específico do registo de dados de forma a encaminhar os dados entre o TDI e o TDO quando se encontrar no estado *Shift-DR* do controlador. O código binário de uma instrução é definido como sendo uma sequência de bits deslocados em série para o registo de instruções através do pino TDI quando o

controlador se encontra no estado *Shift-IR*. A norma obriga a utilização de um número mínimo de instruções (definidas como instruções públicas), permitindo também a utilização de instruções opcionais (definidas como instruções privadas). As instruções públicas definidas pela norma são: *BYPASS*, *SAMPLE*, *PRELOAD* e *EXTEST*, se o registo de identificação do dispositivo for incluído no componente, a instrução *IDCODE* deverá ser também incluída, assim como a instrução *USERCODE* (se o identificador do dispositivo poder ser programado pelo utilizador). As instruções opcionais definidas pela norma são: *INTEST*, *RUNBIST*, *CLAMP* e *HIGHZ*. As instruções privadas poderão não ser documentadas, podendo ser utilizadas para realizar testes apenas por pessoas que conheçam a utilização dessas instruções, no entanto o fornecedor deve identificar claramente os códigos binários que poderão causar uma operação perigosa (p.ex. se uma instrução colocar as entradas de um componente como sendo saída, poderá danificar o componente se a instrução for carregada quando o componente está rodeado por outros componentes na placa).

O registo *bypass* contem uma única etapa no registo de deslocamento e é utilizado para permitir um caminho série mínimo entre os pinos TDI e TDO de um componente, quando não é necessário realizar um teste operacional ao componente. Isto permite que os dados do teste seja deslocados mais rapidamente aumentando a performance de operação do teste. O código binário para a instrução *BYPASS* deve ser {111…1}. A instrução *BYPASS* deverá selecionar o registo *bypass* de modo a interligar o caminho série dos dados entre o TDI e o TDO no momento em que o estado do controlador for *Shidt-DR* , e estando neste modo, a operação da logica de teste não deverá afetar a operação do sistema lógico *core*.

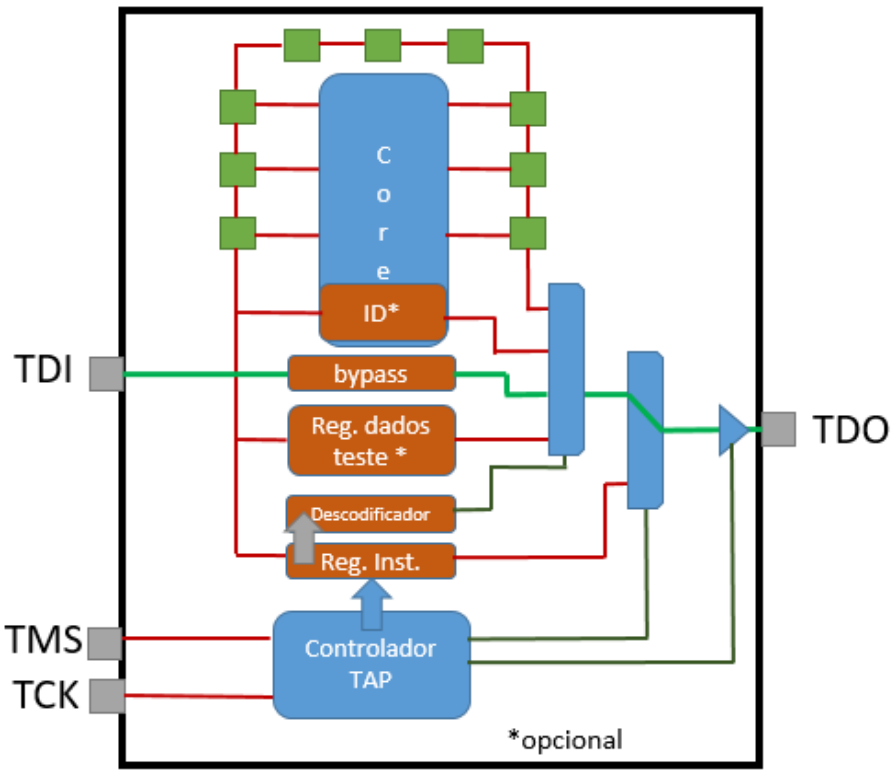


Figura 22 Caminho dos dados quando carregada a instrução *BYPASS*

A instrução *SAMPLE* permite realizar uma amostragem instantânea da operação normal do componente. A instrução deve selecionar apenas o registo *boundary-scan* de modo a interligar o caminho série entre o TDI e o TDO, quando se encontra no estado *Shift-DR* do controlador. Quando a instrução é selecionada, a operação lógica de teste não deverá afetar a operação normal do *core* do circuito integrado, e os estados de todos os sinais que encontram ligados ao *core* deverão ser carregados para os registos *boundary-scan* quando é detetado um flanco ascendente do sinal TCK, encontrando-se o controlador no estado *Capture-DR*. Esta instrução é utilizada para verificar a interação entre os circuitos integrados durante as suas operações normais, assim como para realizar o teste de diagnóstico do próprio circuito integrado sem necessidade de acesso físico a cada um dos pinos. Na Figura 23 encontra-se ilustrado o caminho realizado pelos dados quando é carregada a instrução *SAMPLE*.

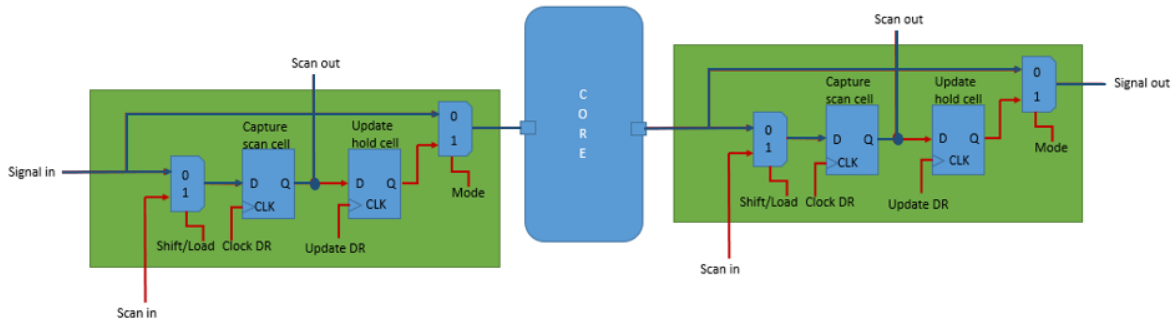


Figura 23 Caminho dos dados quando carregada a instrução *SAMPLE*

Tal como pode ser analisado na figura, a instrução *SAMPLE* não interfere diretamente com o funcionamento normal do *core*. Os dados que são adquiridos pelos pinos de entrada são direcionados para o circuito *core*, assim como os dados que são enviados pelo *core* são transmitidos diretamente para os pinos de saída (isto é possível colocando o sinal *Mode* selecionado a 0). Os sinais de entrada e de saída do *core* são amostrados para o *Scan Out*.

A instrução PRELOAD permite que os dados que se encontram no registo *boundary-scan* sejam carregados para as saídas paralelas. A instrução deverá selecionar apenas o registo *boundary-scan* entre o sinal TDI e TDO quando se encontra no estado *Shift-DR*. Tal como na instrução *SAMPLE*, a operação lógica do teste não deverá afetar a operação normal do *core* do integrado, ou sobre os sinais que são interligados entre os pinos do circuito integrado e o *core*. Ao carregar a instrução, as saídas paralelas incluídas nas células do registo *boundary-scan* devem carregar os dados contidos no *shift-register* (apenas quando existe um flanco descendente no sinal TCK e o estado do controlador se encontra em *Update-DR*). Na Figura 24 encontra-se ilustrado o caminho realizado pelos dados quando é carregada a instrução *PRELOAD*.

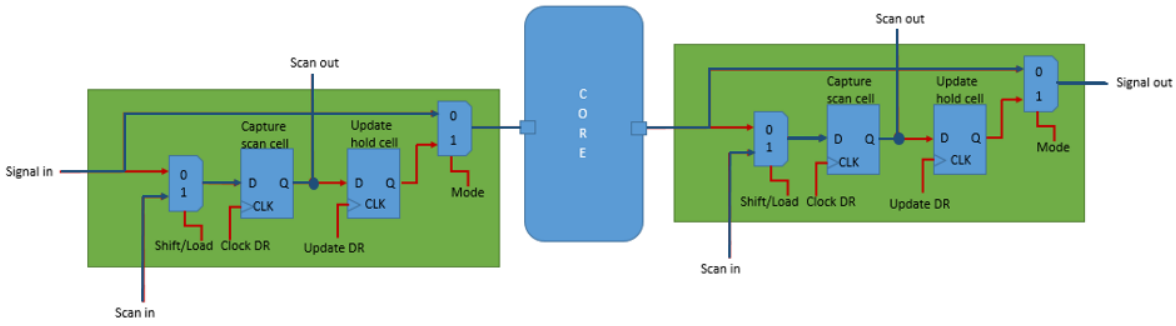


Figura 24 Caminho dos dados quando carregada a instrução *PRELOAD*

Tal como pode ser analisado na figura, a instrução *PRELOAD* não interfere diretamente com o funcionamento normal do *core*. Os dados que são adquiridos pelos pinos de entrada são direcionados para o circuito *core*, assim como os dados que são enviados pelo *core* são transmitidos diretamente para os pinos de saída (isto é possível colocando o sinal *Mode* selecionado a 0).

Como as instruções *SAMPLE e PRELOAD* implementam a funcionalidade um do outro, estes comandos podem ser juntos num só. Enquanto o *SAMPLE* captura os dados provenientes do registo *boundary-scan* permitindo aceder ao dado deslocado através do sinal TDO, o comando *PRELOAD* desloca os dados para o registo *boundary-scan* através do sinal TDI para que possam ser carregados em registos paralelos. Os estados do controlador TAP passam por *Capture-DR, Exit1-DR* e *Update-DR* enquanto a instrução *SAMPLE/PRELOAD* está selecionada.

A instrução *EXTEST* permite realizar o teste ao circuito externo ao circuito integrado (tipicamente às ligações realizadas na placa). As células do registo *boundary-scan* ligadas aos pinos de saída são utilizados para aplicar estímulos de teste, enquanto que os pinos de entrada capturam os sinais de resultados do teste. Normalmente os dados devem ser carregados nas saídas paralelas do registo *boundary-scan* utilizando a instrução *PRELOAD* antes de selecionar a instrução *EXTEST*. Na instrução *EXTEST*, o *core* do circuito integrado deverá manter no estado indeterminado até que seja aplicado o *reset* ao sistema. A instrução deverá selecionar apenas o registo *boundary-scan* para encaminhar os dados entre o sinal TDI e TDO quando se encontra no estado *Shif-DR* do controlador. Quando a instrução *EXTEST* está selecionada, o estado de todos os sinais dos pinos de saída devem ser definidos pelos dados mantidos no registo *boundary-scan* e alterados apenas quando o sinal TCK apresenta um flanco descendente, enquanto o controlador se encontra no estado *Update-DR*. O estado de todos os sinais recebidos nos pinos de entradas do sistema, deverão ser carregados no registo *boundary-scan* quando o sinal TCK apresenta um flanco ascendente enquanto o controlado se encontra no estado Capture-DR. O caminho realizado pelos dados quando carregada a instrução *EXTEST* encontra-se ilustrado na Figura 25.

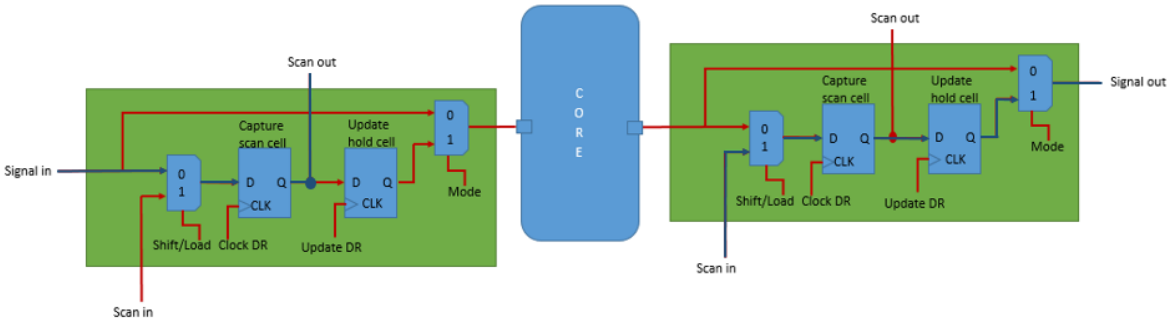


Figura 25 Caminho dos dados quando carregada a instrução *EXTEST*

Os dados que chegam aos pinos de entrada do circuito integrado são capturados e encaminhados para examinação no componente, nos pinos de saída os dados são encaminhados do componente para serem aplicados nas interligações externas. Tipicamente, o primeiro estímulo de teste a ser aplicado (que será utilizado pela instrução *EXTEST*) irá ser deslocado para o registo *boundary scan* quando carregada a instrução *PRELOAD*. Assim, quando o estado do controlador passa para o *Update-IR*, o dado conhecido será conduzido imediatamente do componente para as ligações externas.

A instrução *INTEST* permite realizar o teste lógico do sistema core enquanto o circuito integrado já se encontra em funcionamento na placa. Utilizando esta instrução, são aplicados estímulos de teste e deslocados para o sistema lógico *core*. Os resultados do teste são capturados no registo *boundary-scan* e são examinados pelo deslocamento posterior. Os dados tipicamente deverão ser carregados nas saídas paralelas do registo de deslocamento do *boundary-scan* utilizando a instrução *PRELOAD* antes de utilizar a instrução *INTEST*. A instrução após ser carregada seleciona o registo *boundary scan* para ser conectado entre o TDI e o TDO quando se encontra no estado *Shift-DR* do controlador. As saídas do sistema *core* deverão ser definidas de duas formas distintas: Todos os sinais que são encaminhados para fora do circuito integrado deverão ser os dados mantidos no registo *boundary-scan* e devem apenas alterar o seu estado quando o controlador se encontra no estado *Update-DR* (na existência de um flanco descendente do sinal TCK); ou então todas as saídas dos componentes deverão ficar no estado inativo, ou seja, deverão ficar em alta impedância. A instrução *INTEST* requere que a lógica do sistema *core* possa ser operada no modo passo a passo, onde o circuito executa uma ação por cada deslocamento no registo *boundry-scan*. O caminho realizado pelos dados quando carregada a instrução *INTEST* encontra-se ilustrado na Figura 26.

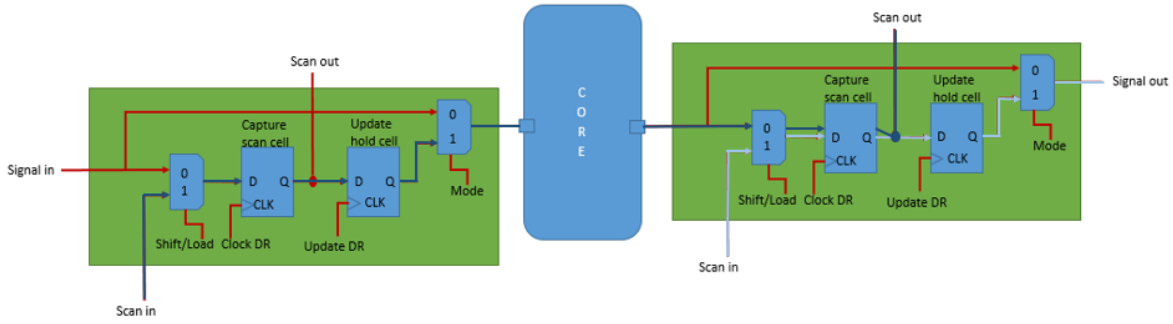


Figura 26 Caminho dos dados quando carregada a instrução INTEST

Tal como pode ser analisado na figura anterior, existem dois caminhos (azul escuro e a azul claro) que representa o fluxo dos dados dependendo da origem dos mesmos. Os dados resultantes da lógica de teste do sistema *core* deverão seguir o caminho indicado com a cor *azul escura*, os dados que serão aplicados fora do circuito integrado correspondem ao fluxo de dados com a cor *azul claro*. Para cada teste, a saída paralela da célula *boundary-scan* é atualizada pelo deslocamento do dado antes do estado do registo de deslocamento ser reescrito com a resposta do teste.

Outra instrução opcional é a *RUNBIST*, que permite realiza o teste autónomo ao componente. Enquanto a instrução *RUNBIST* está selecionada, o estado de todos os pinos de saída são determinados pela logica de teste, estes estados poderão ser dois distintos. O estado do pino pode ser determinado pelo dado mantido no registo *boundary-scan*, que é deslocado para a saída paralela durante cada passo executado, ou então todos os pinos de saída do sistema poderão ser forçado a entrar no estado inativo (alta-impedância). Quando esta instrução é selecionada, o registo de dados de teste onde são carregados os resultados do auto teste, será conectado entre os pinos TDI e TDO quando o controlador entra no estado *Shift-DR*. Esta operação de auto teste só será executada quando o controlador se encontra no estado *Run-Test/Idle*. A sequência de passos necessários para a execução da instrução *RUNBIST* pode ser definida como:

- Opcionalmente, poderá inicializar o registo *boundary-scan* (através da instrução *PRELOAD* por exemplo), apenas será necessário se o estado do pino durante o auto teste, for para ser determinado pelos dados contidos no registo das saídas paralelas.

- Inicializar o auto teste, analisando a instrução *RUNBIST* no registo de instruções;

- Executar o auto teste, garantindo que o controlador TAP permanece no estado *Run-Test/Idle* durante o tempo necessário para que a execução do auto teste seja completada.

- Analisar os resultados do *auto teste* fazendo com que o controlador TAP fique no estado *Shift-DR* e analise os resultados do teste do registo conectado para o TDI e TDO.

A instrução *CLAMP* é considerada pela norma como sendo opcional, esta instrução seleciona o registo *bypass* para se interligar entre o sinal TDI e TDO, permitindo que o estado de todos os sinais conduzidos pelos pinos de saída do sistema sejam definidos pelos dados contidos no registo *boundary-scan*. Esta instrução deverá ser carregada quando o estado do controlador se encontra em *Shift-DR*, uma vez que é utilizado os dados que estão carregados no registo *boundary-scan*, pode-se utilizar a instrução *PRELOAD* para esse efeito. Durante o teste do circuito integrado, ou de um grupo de circuitos integrados, poderá ser necessário aplicar valores estáticos nos sinais que controlam a operação lógica do circuito e para tal, deverá ser utilizada esta instrução.

O uso de registo de indentificadores de dispositivos permite transmitir informação série através do componente com as seguintes informações:

-Identificação do fabricante

-Identificação do *part number*

-O número da versão do dispositivo

Para ter acesso a estas informações, a norma define duas instruções opcionais, a instrução *IDCODE* e a instrução *USERCODE*. A instrução *IDCODE* permite obter a partir do registo de identificação do dispositivo o código identificador do fornecedor, quando o controlador se encontra no estado *Capture-DR*, selecionando o registo entre o TDI e o TDO quando o estado do controlador se encontra no *Shift-DR*. A utilização desta instrução, não deverá afetar a operação do sistema lógico *core*.

A utilização da instrução *USERCODE* seleciona o registo de identificação do dispositivo para ser interligado entre os pinos série TDI e TDO, quando o estado do controlador se encontra no *Shift-DR*. Quando esta instrução está selecionada, é possível obter o código

identificador de 32 bits quando o estado se encontra no *Capture-DR* do controlador. Tal como a instrução anterior, esta não deverá interferir na operação lógica do *core*.

A instrução *HIGZ* coloca o componente num estado em que todas as saídas do sistema lógico são colocados no estado inativo (em alta impedância). Esta instrução deve selecionar o registo *bypass* entre o sinal TDI e TDO quando o controlador se encontra no estado *Shift-DR*. Esta instrução deverá colocar o sistema *core* de forma a não danificar os resultados dos sinais.

3.2. INFRAESTRUTURA IEEE1149.4

Toda a informação presente neste subcapítulo teve como base o documento [25]. A infraestrutura IEEE1149.4 que será a base principal deste trabalho, foi desenvolvida tendo como base a infraestrutura IEEE1149.1. A grande diferença face a infraestrutura anterior é que permite realizar teste a circuitos com sinais mistos (analógicos e digitais). Na Figura 27 encontra-se ilustrada a estrutura de um circuito integrado que se encontra em conformidade com a infraestrutura IEEE1149.4.

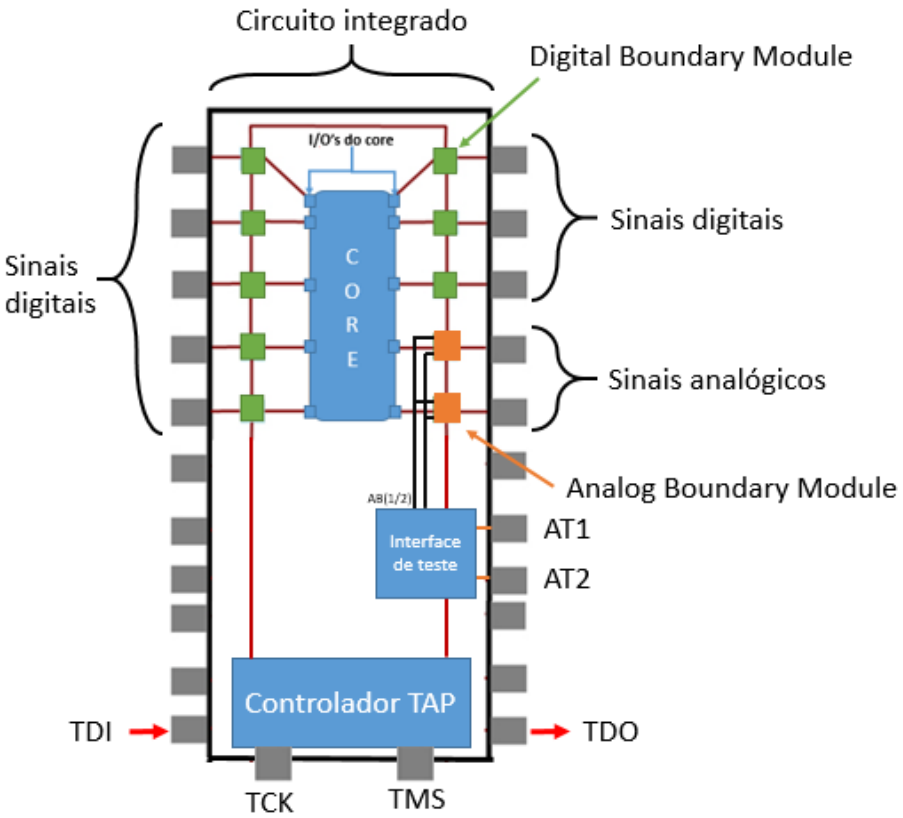


Figura 27 Infraestrutura IEEE1149.4 aplicada num circuito integrado

Tal como se encontra na ilustração anterior, a infraestrutura possui *Analog Boundary Modules* (ABMs) em todos os pinos que possuem a funcionalidade para análise de sinais analógicos. Para acesso aos sinais analógicos de teste, a norma inclui um porto específico, o *analog test acess port* (ATAP). Este porto consiste em dois pinos (AT1 e AT2), uma interface para o barramento de teste analógico (Test Bus Interface Circuit - TBIC) e duas linhas de barramento internas de teste analógico (AB1 e AB2). Os restantes pinos associados aos sinais digitais são idênticos à infraestrutura IEEE1149.1 já explicado anteriormente, no entanto as células utilizadas para realizar a interface entre os pinos externos do circuito integrado e o *core* são denominados por *Digital Boundary Module* (DBM) em vez de *Boundary Scan Cell* como era designado na infraestrutura IEEE1149.1. A máquina de estados e todas as instruções apresentadas anteriormente, encontram-se aplicadas nesta norma, com as funções adicionais para poder realizar testes com a interface analógica. O ABM é o elemento principal para realizar testes a sinais analógicos. Os ABM's encontram-se interligados a cada um dos pinos externos ao circuito integrado que possui a função analógica, sendo o sinal externo transferido para o sistema *core*, no entanto, de forma a possibilitar vários tipos de acesso a testes, a norma obriga que cada ABM tenha a facilidade de se desconectar do sistema *core*. Na Figura 28 encontra-se a estrutura do módulo, que consiste essencialmente por seis switches interligados aos pinos.

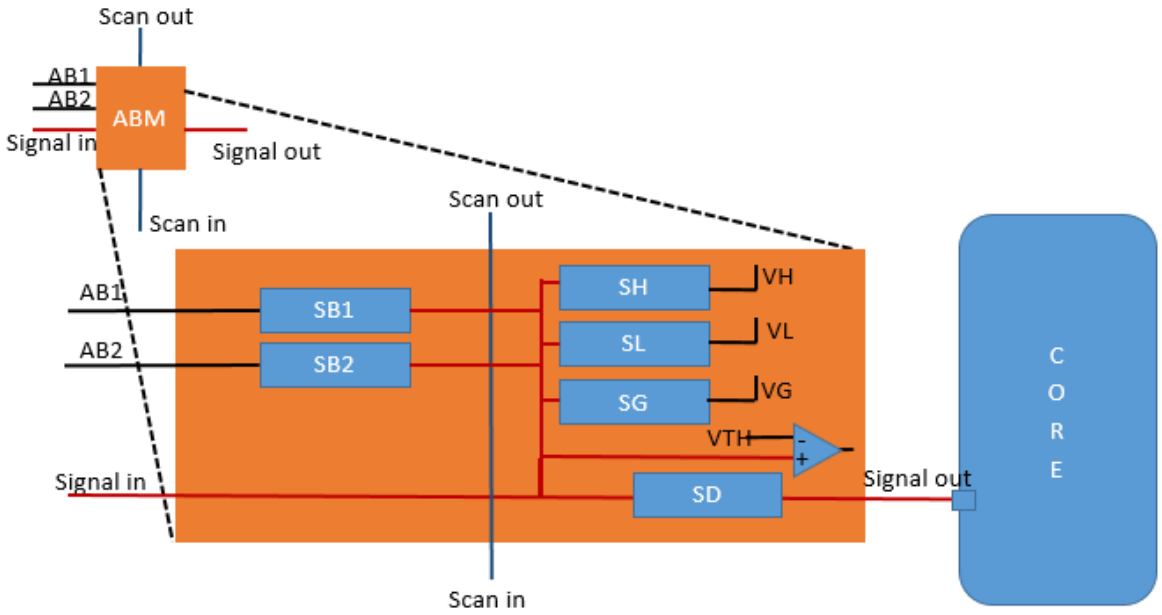


Figura 28 Estrutura de um ABM

Os *switches* SB1 e SB2 permitem que os barramentos analógicos sejam utilizados como pontas de prova virtuais de forma que nos pinos externos ao circuito integrado (AT1 e/ou AT2) se possa aceder aos estados sem ser necessário aceder fisicamente aos pinos.

Os ABM's devem ser capazes de monitorizar o estado de um sinal digital que seja aplicado a um pino analógico. A tensão aplicada no pino será digitalizada em relação a uma tensão Vth (conceitual ou real) cujo valor está na gama

$$\left(\frac{V_H + V_L}{2}\right) \pm \left(\frac{V_H - V_L}{4}\right) \quad \textbf{(1)}$$

Utilizando o comparador entre o sinal aplicado no pino analógico e o sinal Vth, é possível perceber o estado do sinal digital, facilitando a deteção de falhas de interligação.

Para que que o pino seja capaz de fornecer dois níveis de tensão (VH e VL) para o circuito externo, são utilizados dois *switches*, SH e SL respetivamente. Estes níveis de tensão permitem realizar o teste de interligação nos pinos analógicos, de forma a utilizar os mesmos métodos para testes de interligação digital.

Utilizando o *switch* VG, permite-se que o pino seja ligado a um sinal de referência (GND por exemplo).

Para que seja possível abrir a ligação entre o pino externo e o sistema *core*, é utilizado um *switch* SD. O isolamento entre os dois sistemas poderão ser uteis para realizar testes externos ao sistema, fazendo com que o *core* não responda com sinais de ruido, ou sinais inseguros enquanto decorre o teste.

Para realizar a amostra nas ABM's, todas as regras devem de ser satisfeitas de forma com os estados dos interruptores identificados na tabela seguinte.

Tabela 2 Exemplos dos diferentes estados dos *switches* para amostras das ABM's.

Exemplo	Estado dos *switches*						Estado do pino
	SD	SH	SL	SG	SB1	SB2	
P0	0	0	0	0	0	0	Pino completamente isolado

P1	0	0	0	0	0	1	Modo de observação através do AB2
P2	0	0	0	0	1	0	Ligado ao barramento AB1
P3	0	0	0	0	1	1	Ligado ao barramento AB1 e observar pelo barramento AB2
P4	0	0	0	1	0	0	Ligado ao sinal VG
P5	0	0	0	1	0	1	Ligado ao sinal VB e observar pelo barramento AB2
P6	0	0	0	1	1	0	Ligado ao sinal VB e ao barramento AB1
P7	0	0	0	1	1	1	Ligado ao sinal VB, ao barramento AB1 e observar pelo barramento AB2
P8	0	0	1	0	0	0	Ligado ao sinal VL
P9	0	0	1	0	0	1	Ligado ao sinal VL e observar pelo barramento AB2
P10	0	0	1	0	1	0	Ligado ao sinal VL e ao barramento AB1
P11	0	0	1	0	1	1	Ligado ao sinal VL, ao barramento AB1 e observar pelo barramento AB2

P12	0	1	0	0	0	0	Ligado ao sinal VH
P13	0	1	0	0	0	1	Ligado ao sinal VH e observar pelo barramento AB2
P14	0	1	0	0	1	0	Ligado ao sinal VH e ao barramento AB1
P15	0	1	0	0	1	1	Ligado ao sinal VH, ao barramento AB1 e observar pelo barramento AB2
P16	1	0	0	0	0	0	Ligado ao sistema core, e isolado de todo o circuito de teste
P17	1	0	0	0	0	1	Ligado ao sistema core e observar pelo barramento AB2
P18	1	0	0	0	1	0	Ligado ao sistema core e ao barramento AB1
P19	1	0	0	0	1	1	Ligado ao sistema core, ao barramento AB1 e observar pelo barramento AB2

Os caminhos aqui indicados são controlados apenas através da configuração de 4 bits, de acordo com o modo de funcionamento em que se encontram os dispositivos. Na tabela seguinte encontra-se definido os caminhos configurados de acordo com a configuração dos bits e da instrução atualmente definida.

Tabela 3 Código binário para configuração dos caminhos das ABM's.

Código C/D/B1/B2	Instrução		Código C/D/B1/B2	Instrução	
	Extest,Clamp	**Probe, Intest**		**Extest,Clamp**	**Probe, Intest**
0000	P0	P16	**1000**	P8	*
0001	P1	P17	**1001**	P9	*
0010	P2	P18	**1010**	P10	*
0011	P3	P19	**1011**	P11	*
0100	P4	*	**1100**	P12	*
0101	P5	*	**1101**	P13	*
0110	P6	*	**1110**	P14	*
0111	P7	*	**1111**	P15	*

Cada ABM deverá conter um registo de controlo e um registo de atualização. O registo de controlo deve conter quatro estados (*BUS1*, *BUS2*, *CONTROL* e *DATA*) e faz parte do registo *boundary-scan*. O registo de atualização deve carregar os dados paralelamente do registo de controlo e de qualquer logica combinatória em resposta ao sinal TCK (no flanco descendente) enquanto o controlador TAP se encontra no estado *Update-DR*. Na Figura 29 encontra-se ilustrada a estrutura de controlo para um ABM.

O resultado da digitalização do pino de tensão deverá ser capturado no estado *DATA* do registo de controlo quando o sinal TCK encontra um flanco ascendente enquanto o controlador TAP está no estado *Capture-DR.*

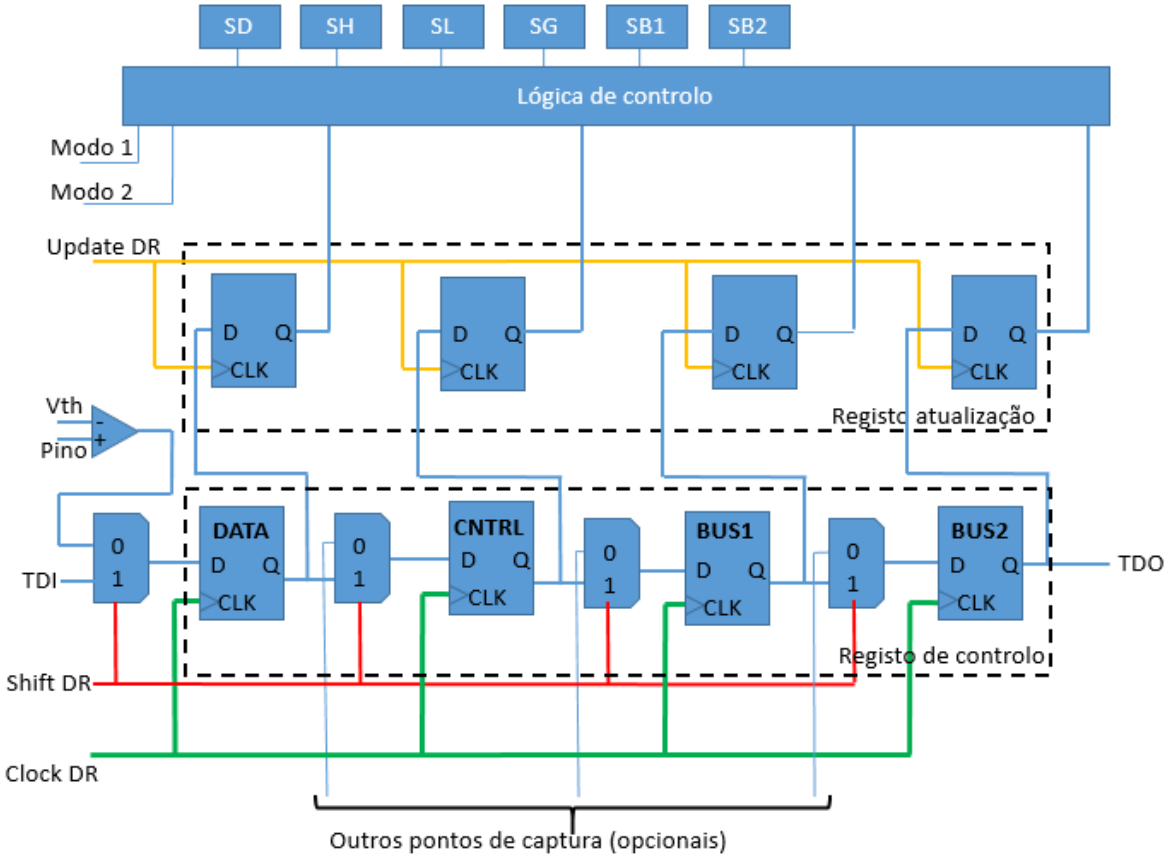


Figura 29 Estrutura do controlo do ABM

O controlo dos sinais para cada *switch* deriva através da lógica de controlo, dependendo do conteúdo existente no registo de atualização e ativação dos sinais modo (Modo1 e Modo2), que deve ser fornecido pelo descodificador de instruções. Para cada *switch*, existe um estado específico que o faz ativar:

- $SD = \overline{M_1}$
- $SH = CDM_1M_2$
- $SL = C\bar{D}M_1M_2$
- $SL = \bar{C}DM_1M_2$
- $SB_1 = B_1M_2$
- $SB_2 = B_2M_2$

De acordo com as expressões anteriores, para ativar o *switch* SD, M1 não pode estar ativo, por exemplo. Os enumeradores C, D, B1, B2, M1 e M2 representam os *flipflops Control*, *Data*, *Bus1* e *B2* assim como o sinal *Modo1* e *Modo2* respetivamente.

O porto ATAP, juntamente com o TBIC e os barramentos de teste analógico fornecem uma estrutura que permite realizar o teste paramétrico. A estrutura TBIC possui dois barramentos de teste analógico (AB1 e AB2), essas linhas são ligadas internamente ao porto ATAP e controlado de acordo com os *switch* existentes no interior do TBIC: Durante a operação normal do sistema *core*, o barramento interno de teste analógico deverá ficar completamente isolado do porto ATAP. A estrutura básica de um TBIC encontra-se ilustrado na Figura 30,

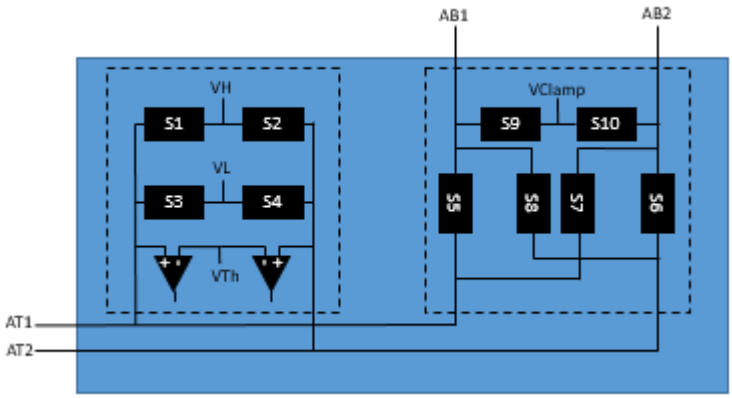


Figura 30 Estrutura básica de um TBIC

A estrutura básica de um TBIC consiste em 10 *switch* e em dois comparadores. Os *switch* permitem realizar a ligação dos pinos ATAP:

-Às tensões VH (Tensão *high*) e VL (tensão *Low*) que agem como valore lógicos para teste de interligações, utilizando os *switch*es S1 até S4.

-Às linhas de barramento interno de teste, utilizando os *switch*es S5 até S8

-À tensão de referência VClamp, utilizando os *switch*es S9 e S10.

A estrutura permite comparar os valores adquiridos nos pinos do porto ATAP com a tensão VTH de forma a obter um sinal de um bit.

A norma define um determinado estado para os *switch*es existentes na configuração do TBIC, os diferentes estados encontram-se definidos na Tabela 4.

Tabela 4 Estado para os interruptores na configuração do TBIC

Exemplo	Estado dos *switches*										Funções
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	
P0	0	0	0	0	0	0	0	0	1	1	AB1 e AB2 são desligados de AT1 e AT2 e ligados à tensão VClamp
P1	0	0	0	0	0	1	0	0	1	0	AB1 desligado de AT1 e ligados à tensão VClamp, AB2 ligado a AT2
P2	0	0	0	0	1	0	0	0	0	1	AB1 ligado a AT1 e AB2 desligado de AT2 e ligado À tensão VClamp
P3	0	0	0	0	1	1	0	0	0	0	AB1 ligado a AT1 e AB2 ligado a AT2
P4	0	0	1	1	0	0	0	0	1	1	AT1 e AT2 ligados à tensão VL, AB1 e AB2 ligado a Vclamp
P5	0	1	1	0	0	0	0	0	1	1	AT1 ligados à tensão VL e AT2 ligados à tensão VH, AB1 e AB2 ligado a Vclamp
P6	1	0	0	1	0	0	0	0	1	1	AT1 ligados à tensão VH e AT2 ligados à tensão VL, AB1 e AB2 ligado a Vclamp
P7	1	1	0	0	0	0	0	0	1	1	AT1 ligados à tensão VH e AT2 ligados à tensão VH, AB1 e AB2 ligado a Vclamp
P8	0	0	0	0	0	1	1	0	1	0	AB1 ligado à tensão VClamp, AB2 ligado a AT1 e AT2
P9	0	0	0	0	1	0	0	1	0	1	AB2 ligado à tensão VClamp, AB1 ligado a AT1 e a AT2

Tal como os ABM's, os *switch*es são controlados através da configuração de 4 bits, tendo também em consideração o modo de funcionamento em que se encontram os dispositivos. Na tabela seguinte encontra-se definido os caminhos configurados de acordo com a configuração dos bits e da instrução atualmente definida.

Tabela 5 Código binário para configuração dos caminhos do TBIC

Código Ca/C0/D1/D2	Instrução		Código Ca/C0/D1/D2	Instrução	
	Extest,Clamp, Runbist	**Probe, Intest**		**Extest,Clamp, Runbist**	**Probe, Intest**
0000	P0	P0	**1000**	P0	*
0001	P1	P1	**1001**	P8	*
0010	P2	P2	**1010**	P9	*
0011	P3	P3	**1011**	*	*
0100	P4	*	**1100**	*	*
0101	P5	*	**1101**	*	*
0110	P6	*	**1110**	*	*
0111	P7	*	**1111**	*	*

Para controlar o estado do TBIC, é utilizado o controlador TBIC através do registo de controlo e do registo de atualização. O registo de controlo consiste em registos de deslocamento com quatro estados identificados como *CALIBRATE*, *CONTROL*, *DATA1* e *DATA2*, este registo deverá fazer parte do registo *boundary-scan*. Os estados de *DATA1* e *DATA2* devem ser capturados do resultado da digitalização da tensão no pino AT1 e AT2 respetivamente (quando o controlador TAP deteta um flanco ascendente no sinal TCK e encontra-se no estado *Capture-DR*). O registo de atualização também consiste em quatro estados e deve ser carregado paralelamente pelo registo de controlo, em resposta ao flanco descendente do sinal TCK enquanto o controlador TAP se encontra no estado *Update-DR*.

Na Figura 31encontra-se ilustrada a estrutura de implementação de um controlador TBIC.

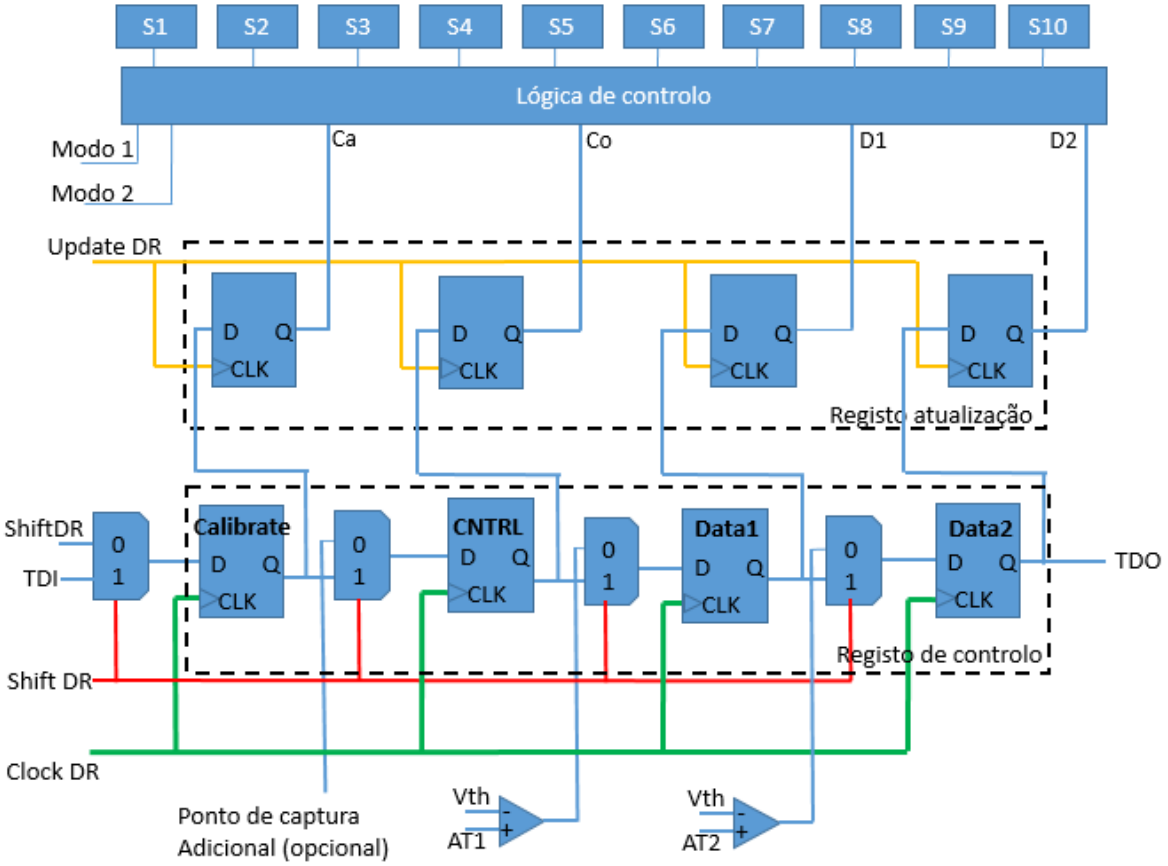


Figura 31 Estrutura de implementação do controlador TBIC.

Tal como se pode verificar na figura anterior, o estado de *Data 1* e de *Data 2* é definido pelo estado digitalizado dos sinais em AT1 e AT2. Os sinais de controlo para cada um dos *switch*es derivam dos valores atuais existentes no registo de atualização e de acordo com os sinais Modo (*Modo 1* e *Modo 2*), tal como é definido na Tabela 6.

Tabela 6 Estado dos sinais "Modo" de acordo com o estado do controlador

Instrução	Modo 1	Modo2
EXTEST, CLAMP, RUNBIST	1	1
PROBE, INTEST	0	1
HIGHZ	1	0
Outras instruções	0	0

O estado dos interruptores são definidos de acordo com as seguintes expressões:

$$
\begin{aligned}
S_1 &= \overline{C_a} C_o D_1 M_1 M_2 \\
S_2 &= \overline{C_a} C_o D_2 M_1 M_2 \\
S_3 &= \overline{C_a} C_o \overline{D_1} M_1 M_2 \\
S_4 &= \overline{C_a} C_o \overline{D_2} M_1 M_2 \\
S_5 &= \overline{C_o} D_1 M_2 (\overline{C_a} + \overline{D_2} M_1) \\
S_6 &= \overline{C_o} D_2 M_2 (\overline{C_a} + \overline{D_1} M_1) \\
S_7 &= C_a \overline{C_o} \overline{D_1} D_2 M_1 M_2 \\
S_8 &= C_a \overline{C_o} D_1 \overline{D_2} M_1 M_2 \\
S_9 &= \overline{S_5} \\
S_{10} &= \overline{S_6}
\end{aligned}
$$

onde Ca é o sinal *CALIBRATE*, Co é o sinal *CONTROL*, D1 é o sinal *DATA1*, D2 o sinal *DATA2*, M1 é o sinal *Modo1* e M2 é o sinal *Modo2*.

A norma define diversas instruções, que já são conhecidas por pertencerem à infraestrutura IEEE1149.1, no entanto são reutilizadas para suportar os testes analógicos. As operações realizadas por qualquer DBM são determinadas de acordo com base nas regras defininas na infraestrutura IEEE1149.1. No entanto as operações realizadas por qualquer ABM é determinado não apenas pelos sinais fornecidos pelo descodificador de instruções (que depende da instrução carregada atualmente no registo de instruções), mas também pelos sinais fornecidos pelo controlador TAP (que depende do estado atual do controlador) e, dependendo da instrução que está selecionada, irá depender do valor contido no registo de controlo de cada ABM.

A norma possui instruções que são consideradas como obrigatórias e instruções opcionais, tal como na infraestrutura IEEE1149.1. Todas as instruções definidas na infraestrutura IEEE1149.1 são utilizadas nesta norma, pelo que foi acrescentada mais uma instrução considerada como sendo obrigatória, a instrução *PROBE*. De seguida serão descritas cada uma das instruções e quais as suas funções.

Quando a instrução *BYPASS* é selecionada, todos os pinos do ATAP devem permanecer isolados do teste analógico interno e de todas as fontes de tensão de teste. Nesta instrução

todos os pinos com função analógica devem ser ligadas ao sistema *core* e deverão ser desligados do barramento interno e externo de teste analógico assim como de todas as fontes de tensão de teste. A instrução *BYPASS* seleciona o registo *bypass* a ser ligado de forma série entre os canais TDI e TDO quando o controlador se encontra no estado *Shift-DR*. A instrução pode ser selecionada colocando o valor lógico 1 em todos os estados do registo de instruções.

A instrução *SAMPLE/PRELOAD* possui duas funções, a *SAMPLE* e a *PRELOAD*, tal como o nome indica. Quando a instrução *SAMPLE/PRELOAD* se encontra selecionada, todos os pinos do porto ATAP devem ficar isolados do barramento de teste analógico interno e de todas as fontes de tensão de teste. Nesta instrução, todos os pinos com a função analógica devem ser interligados ao sistema *core* e deverão ficar isolados do barramento de testes interno e externo analógico assim como de todas as fontes de tensão de teste. A função *SAMPLE* utiliza o registo *boundary scan* de forma a permitir uma amostragem dos dados dos pinos digitais durante a operação normal, essa amostragem é conseguida no instante em que existe um flanco ascendente em TCK e o estado do controlador é *Capture-DR*, podendo ser examinado através do deslocamento do sinal pelo TDO. A amostragem digitalizada dos pinos analógicos é realizada através de um digitalizador de um bit apenas no ABM. A função *PRELOAD* permite que um dado digital seja carregado nas saídas paralelas do *latch* do registo *boundary-scan*. O carregamento do dado é realizado no instante em que ocorre um sinal descendente de TCK e o controlador se encontra no estado *Update-DR*.

A instrução *EXTEST* seleciona cada ABM para que esteja completamente desligado do sistema *core*. Esta instrução serve para realizar testes de interligação simples aplicando níveis lógicos em cada um dos pinos do circuito integrado. Utilizando esta instrução, é possível aplicar sinais analógicos de teste e monitorizar a sua resposta, sendo um dos maiores benefícios para realização de testes de circuitos analógicos sem necessidade de acesso físico ao sistema. Esta instrução seleciona o registo *boundary-scan* para ser interligado de forma série entre o TDI e o TDO quando o estado do controlador é *Shift-DR*. Durante esta instrução, todos os pinos (analógicos e digitais) são desligados do sistema *core*, de modo a não interagir e alterar o sistema, sendo apenas monitorizado o estado de cada um dos pinos de entrada.

A instrução *PROBE* permite que os pinos analógicos sejam monitorizados em AB2 e/ou estimulados através de AB1, enquanto o circuito integrado opera no modo para o qual foi definido (todos os pinos irão estar ligados ao *core* do sistema). A instrução, quando selecionada, irá garantir uma interligação série entre o TDI e o TDO através do registo

boundary-scan quando o estado do controlador for *Shift-DR*. Um, ou os dois pinos do porto ATAP ficarão ligados ao barramento interno de teste correspondente.

A instrução *INTEST* permite realizar o teste ao sistema *core* do componente, mesmo estando este já montado numa placa de circuito impresso. O registo *boundary-scan* é selecionado entre os pinos TDI e TDO para envio de informação série quando o controlador se encontra no estado *Shift-DR*. Na instrução *INTEST*, os pinos digitais encontram-se ligadas às DBM's, o mesmo acontece aos pinos analógicos, que deverão ser ligados às ABM's, permitindo realizar alguns testes ao sistema analógico. Tal como na infraestrutura IEEE1149.1, durante a instrução *INTEST*, os dados que serão enviados para os pinos digitais de saída serão capturados para o registo *boundary-scan* quando o estado do controlador se encontrar no *Capture-DR* e existir um sinal ascendente do TCK. Todos os dados de entrada do sistema *core* digital são fornecidos pelos pinos digitais ou através do sistema *core* analógico, que irá testar os dados fornecidos pelo registo *boundary-scan*. Todos os dados de saída do sistema *core* digital são enviados para os pinos digitais ou para o sistema core analógico que serão capturados pelo registo *boundary-scan*. A norma define que quando esta instrução se encontra ativa, todos os pinos analógicos deverão estar ligados ao sistema *core* analógico, sendo possível interligar qualquer pino analógico ao pino AB1 (para que possa ser estimulado - pinos de entrada) e ao pino AB2 (para que possa ser monitorizado - pinos de saída).

A instrução *IDCODE* é de todo idêntica à já explicada anteriormente na infraestrutura IEEE1149.1, esta instrução permite obter a informação sobre o fabricante do circuito integrado. A instrução deverá ser carregada para o registo de instruções na existência de um flanco descendente do sinal de TCK quando o controlador se encontra no estado *Test-Logic-Reset*, e a instrução será carregada de forma série através dos pinos TDI enquanto o controlador esse encontra no estado *Shift-IR*. A informação será fornecida através do pino TDO quando for detetado um flanco ascendente do sinal de TCK quando o controlador se encontrar no estado *Capture-DR*. Enquanto a instrução está selecionada, os pinos do porto ATAP deverão ficar isolados dos barramentos de teste analógicos, e todos os pinos analógicos deverão estar ligados ao *core*, sendo isolados dos barramentos (interno e externo) de teste analógico e de todas as fontes de tensão de teste existentes no circuito integrado.

A instrução *USERCODE* irá reagir da mesma forma que a instrução *IDCODE*, deixando os pinos analógicos ligados ao sistema *core*, isolando-os dos barramentos, e deixando o porto

ATAP também isolado do barramento de teste. Se a identificação do circuito integrado tiver a possibilidade de ser programável, é utilizada esta instrução para definir um novo número de identificação. O código é carregada no registo de identificação no flanco ascendente do sinal de TCK, enquanto o controlador se encontrar no estado *Capture-DR*, podendo ser lido através do pino TDO.

A instrução *RUNBIST* permite realizar um teste interno do próprio circuito integrado, pelo que a norma recomenda a utilização desta instrução em vez da *INTEST*. Esta instrução irá ser executada quando o controlador se encontrar no estado *Run-Test/Idle*. O resultado do teste é enviado quando o registo de dados é selecionado entre os pinos TDI e TDO (apenas quando o controlador se encontra no estado *Shift-DR*). O resultado do teste é independente dos sinais externos ao circuito integrado. Os pinos de saída analógicos deverão ficar num dos dois estados possíveis, ou ficaram com um dado carregado anteriormente pelo registo *boundary-scan* ou ficarão com o estado de alta impedância.

A instrução *CLAMP* seleciona o registo *bypass* para estar interligado entre o pino TDI e TDO quando o controlador se encontra no estado *Shift-DR*. O estado de todos os pinos de saída (analógicos e digitais) serão definidos pelo conteúdo existente no módulo *boundary*, permitindo impor um estado aos pinos de saída do circuito integrado. Os dados poderão ser carregados previamente através da instrução *SAMPLE/PRELOAD*. Os dados contidos nas ABM's deverão ser carregados para os pinos de saída analógicos e deverão manter-se intactos nos ABM's, o estado dos pinos AT1 e AT2 deverão ser definidos pelo conteúdo existente no registo de controlo TBIC.

A instrução *HIGHZ*, quando selecionada, coloca todos os pinos do circuito integrado em alta impedância, isolando os pinos do porto ATAP das linhas do barramento interno. Quando selecionada esta instrução, o registo *bypass* é selecionado entre o pino TDI e TDO quando o controlador se encontra no estado *Shift-DR*.

3.3. FUNCIONAMENTO DO MÓDULO COM INFRAESTRUTURA IEEE1149.4

A informação presente neste subcapítulo teve como base os documentos [46][47]. Os módulos com infraestrutura IEEE1149.4 não são muito utilizados em aplicações reais, pelo que todos os módulos existentes até ao momento têm vindo a tornar-se obsoletos. No entanto, ainda existe um módulo desenvolvido pela *Texas Instruments*, que se encontra disponível no mercado. O módulo SCAN STA400 detém a funcionalidade de *multiplexer*, sendo este compatível com a infraestrutura IEEE1149.4. O dispositivo permite o acesso até nove pontos analógicos de teste, que podem ser utilizados para observação e controlo dos mesmos. Na Figura 32 encontra-se ilustrado o diagrama de blocos do módulo, sendo possível observar o sistema core (a parte funcional do sistema), assim como a infraestrutura desenvolvida para que esteja de acordo com a norma IEEE1149.4.

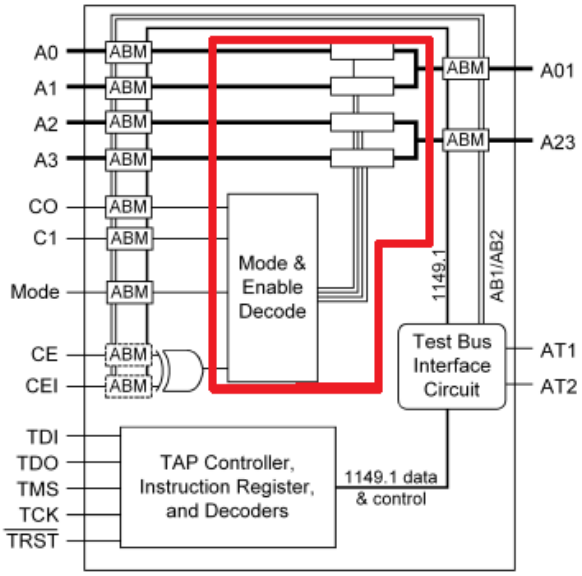


Figura 32 Diagrama de blocos do módulo SCAN STA400

Tal como se pode observar na figura anterior, o circuito funcional (assinalado com a cor vermelha) são dois *multiplexeres*, cada um com duas entradas (pinos A0, A1, A2 e A3) e uma saída analógica (pinos A01 e A02), ou seja, são um *multiplexer* 2:1. Os *multiplexeres* encontram-se interligados a um módulo de controlo, para que possa ser configurado o canal a ser selecionado de cada um dos *multiplexers*, ou para configurar para um único *multiplexer* 4:1, permitindo assim, ter quatro entradas para apenas uma saída analógica. A seleção entre os dois *multiplexeres* 2:1 e um *multiplexer* 4:1 é realizada através do pino Mode. Analisando a Figura 32, é possível verificar que os pinos CE e CEI encontram-se interligados a uma

porta lógica XOR, ou seja o módulo apenas é ativo quando o estado lógico de CE e CEI são diferentes, caso contrário a funcionalidade principal do módulo ficará desativada, sendo apenas possível observar o estado dos pinos utilizando a infraestrutura IEEE1149.4. Para realizar a configuração dos *multiplexers* (relação entre pinos de entrada a saída), são utilizados os pinos C0 e C1, sendo o estado final dos *multiplexeres* definidos de acordo com a tabela de verdade seguinte (ver Tabela 7).

Tabela 7 Tabela de verdade de configuração dos *multiplexers*

CE	Mode	C1	C0	A01	A23
Not equal to CEI	0	0	0	A0	A2
	0	0	1	A1	A2
	0	1	0	A0	A3
	0	1	1	A1	A3
	1	0	0	A0	Not connected
	1	0	1	A1	Not connected
	1	1	0	Not connected	A2
	1	1	1	Not connected	A3
Equal to CEI	x	x	x	Not connected	Not connected

Uma vez que o dispositivo é compatível com as infraestruturas IEEE1149.1 e IEEE1149.4, toda a interface externa no dispositivo é interligada ao core através de onze ABM's (quatro pinos de entrada, e dois de saída do *multiplexer*, assim como cinco para o controlo dos mesmos), sendo estes considerados os pontos analógicos de observação e de controlo individual de cada pino. Para comunicação com a infraestrutura, o módulo possui vários pontos externos do TAP, tais como TDI, TDO, TMS TCK e TRST para interface com o controlador TAP, assim como os pinos ATAP AT1 e AT2 para realizar a interface com o TBIC. Tal como é indicado na norma, o pino TRST é opcional, permitindo fazer reset à máquina de estados do controlador TAP.

No *datasheet* do módulo, encontra-se definido os bits de controlo dos ABM's e do TBIC, no entanto estes estados deverão ser definidos de acordo com as tabelas apresentadas na norma IEEE1149.4. A descrição da funcionalidade dos ABM's e do TBIC podem ser mal interpretados, pelo que para uma boa implementação, toda a informação deverá ser consultada através da norma (consultar a Tabela 3 e a Tabela 5).

Para controlo do estado dos onze ABM's e do TBIC, é utilizado o registo *Boundary Scan*. O controlo de cada ABM, assim como do TBIC, é realizado através de 4 bits, logo o registo *Boundary Scan* é constituído por 48 bits (4bits*12). O *datasheet* do módulo identifica como

se encontra organizado o registo *boundary scan*, na Figura 33 encontra-se ilustrado o registo e a ordem que os bits deverão ser transmitidos entre o TDI e o TDO.

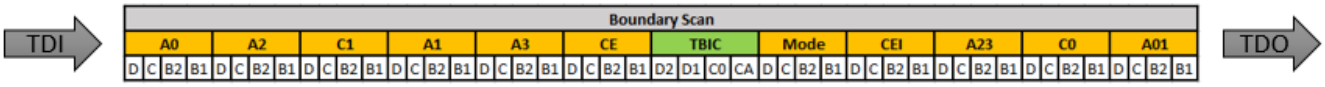


Figura 33 Registo Boundary Scan

Na figura anterior, os ABM's encontra-se ilustrados com a cor laranja e o TBIC encontra-se com a cor verde. Tal como se pode verificar na figura, o bit B1 do ABM A01 deverá ser enviado para o registo em primeiro lugar e por último será enviado o bit D do ABM A0. É de notar que a ordem dos bits neste circuito integrado não se encontra de acordo com o sugerido pela norma, no entanto, para realização do controlo, deverá ser feita uma ligação com a mesma visto que a nomenclatura se mantem igual. Tanto o TBIC como os ABM's deverão ser configurados de acordo com a Tabela 5 e a Tabela 3, no entanto a ordem de envio dos bits deverá ser feito de acordo com a ordem do registo Boundary Scan definido na Figura 33.

Para controlo do controlador TAP é utilizado o registo de instruções. No *datasheet* do módulo, o registo de instruções encontra-se definido por um conjunto de 20 bits, contendo no total 10 instruções possíveis de utilização. No conjunto de instruções, encontra-se a possibilidade de utilizar 5 instruções obrigatórias (na prática são 4 instruções, sendo uma repetida) e 5 instruções opcionais. Na Tabela 8 encontra-se definidas as instruções possíveis de utilização no controlador TAP, assim como o respetivo código binário.

Tabela 8 Instruções do controlador TAP

Instrução	Código binário	Descrição
Bypass	11111111111111111111	Bypass no boundary-scan registor, contendo apenas 1 bit entre TDI-TDO
EXTEST	00000000000000000000	Todos os pinos controlados pelo registo boundary scan
EXTEST	11111111111111101000	Código binário alternativo para o EXTEST
SAMPLE	01111111111111111000	Acesso ao registo boundary-scan, mas não atualizado
FASTACC	11111111111111001011	Acesso aos registos e suprime o Capture-DR
INTESTD	11111111111111110000	Idêntico à instrução INTEST da infraestrutura IEEE1149.1
INTESTA	11111111111111111000	Idêntico à instrução INTESTD, no entanto todos os pinos ligados ao sistema core
PROBE	11111111111111111000	Instrução aplicada apenas na infraestrutura IEEE1149.4
HIGHZ	01111111111111001111	Todos os pinos ficam em alta impedância
CLAMP	11111111111111101111	Os pinos mantêm o estado lógico anterior

É de notar que as instruções referidas pela infraestrutura como sendo obrigatórias encontra-se com a cor amarela, as que foram acrescentadas no módulo por opção do fabricante, encontram-se com a cor verde. O bit menos significativo deverá ser enviado em primeiro lugar para o registo de instruções.

3.4. FERRAMENTA EDUCACIONAL COM INFRAESTRUTURA IEEE1149.1

Para realizar a observação e o controlo do módulo SCAN STA400 será necessário uma aplicação que implemente os protocolos definidos para comunicação com o TAP. Para além da aplicação, também será necessário *hardware* que faça a interface entre o dispositivo e o computador. São diversas as aplicações e o *hardware* desenvolvido a nível comercial e a nível educacional, que implementam funções específicas, de acordo com as necessidades de cada teste. Existem equipamentos que permitem apenas o controlo e observação dos pinos dos dispositivos e outros que fornecem mais opções, tais como realizar a configuração dos mesmos, tal como as FPGA's.

Uma vez que o custo dos equipamentos, utilizados para realizar a interface entre as aplicações e os módulos, é muito elevado foi necessário perceber o funcionamento a baixo nível para realizar a comunicação entre a aplicação e o módulo SCAN STA400. Para ajudar a perceber como deverá ser implementada a comunicação, a *Texas Instruments* desenvolveu uma aplicação (*Scan Educator*) que permite perceber o funcionamento da infraestrutura *Boundary Scan*. O *software* foca-se em dispositivos que implementam a infraestrutura IEEE1149.1, no entanto após se perceber o funcionamento do controlo dessa infraestrutura, transpor para a infraestrutura IEEE1149.4 torna-se mais facilitada. O *Scan Educator* foi desenvolvido para correr em máquinas com o sistema operativo *Windows XP*, pelo que tornou-se necessário a utilização de uma máquina virtual com *Windows XP* para experimentação do mesmo. Na figura seguinte encontra-se os menus apresentados na aplicação *Scan Educator*. Para além de informação sobre a infraestrutura IEEE1149.1, esta também permite realizar o teste do controlador TAP, controlando interruptores para gerar sinais no TDI, TMS e TCK e obter informação do estado atual do controlador TAP, do estado do registo de instruções, do registo *Boundary Scan* e do estado de TDO. De forma bastante intuitiva, o *software* permite analisar como os dados são transmitidos entre os registos e qual o deslocamento dos mesmos.

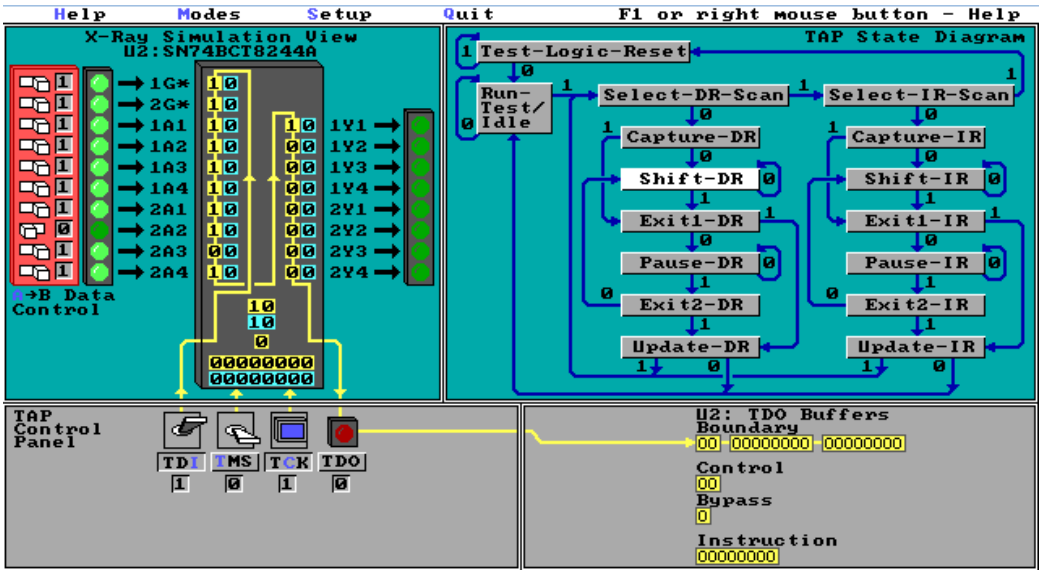


Figura 34 Aplicação *Scan Educator* com controlo digital

Para além do teste do controlador TAP, o *software* de simulação, permite realizar o modo *SCAN control* que é capaz de simular a interação entre dois módulos ligados em série através do controlador TAP. Nesta simulação, é possível definir os registos *boundary scan* e o registo de instruções de cada módulo através de código binário, sendo gerado os sinais TDI TMS e TCK automaticamente dependendo da ação que se pretende executar (escrever para o registo de instruções, para o registo *boundary scan* ou fazer *reset* ao teste).

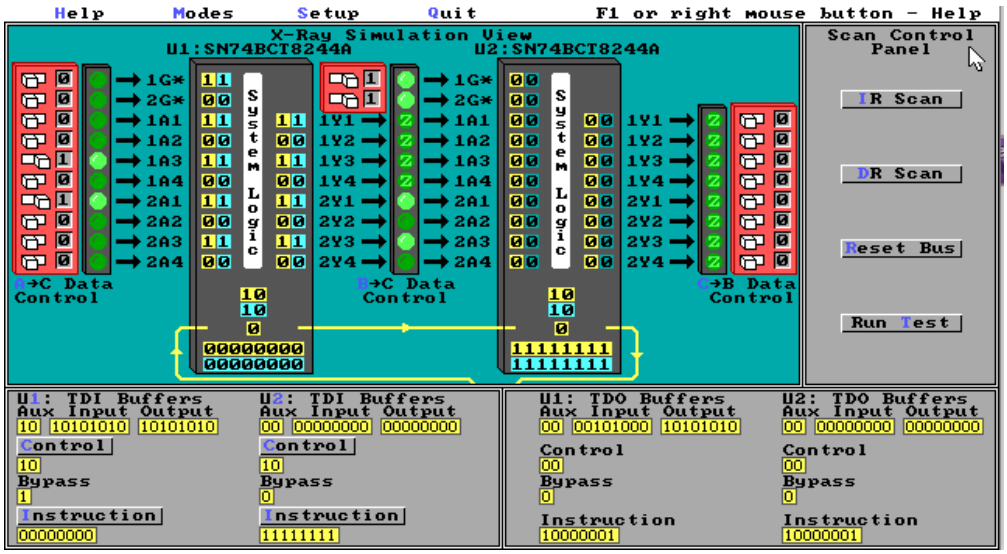


Figura 35 Aplicação *Scan Educator* com interação de múltiplos dispositivos

Utilizando esta ferramenta educacional torna-se muito mais simples, perceber como deverá ser feito o controlo dos sinais para permitir testar o circuito de missão.

3.5. DISPOSITIVOS LÓGICOS CONFIGURÁVEIS

Os dispositivos lógicos são muito utilizados na área da eletrónica porque fornecem funções específicas como realizar a interface entre diferentes dispositivos, realizar comunicação de dados, processar sinais, exibir dados, realizar operações, entre outras funções. Os dispositivos lógicos podem ser classificados em duas categorias distintas os dispositivos fixos e os configuráveis . Tal como o nome sugere, os circuitos que possuem dispositivos de lógica fixa são permanentes, ou seja, realizam apenas uma função ou um grupo de funções e após o seu fabrico, não é possível alterar as suas características. O tempo necessário para desenvolver dispositivos de logica fixa (desde o desenho, protótipos e fabrico final) pode levar vários meses ou até anos dependendo da complexidade dos dispositivos, e se for necessário alterar algum requisito poderá levar a custos muito elevados ou mesmo a realização de um projeto novo.

Os dispositivos de lógica configurável oferecem uma grande variedade de funções, podendo ser alteradas a qualquer momento para realizar qualquer tipo de função (desde que disponível). Com estes dispositivos o desenho dos mesmos é realizado através da utilização de *softwares* que permite não só o desenvolvimento, como realizar simulações e testes dos equipamentos. De forma muito simples e rápida, é possível desenhar um circuito, configurá-lo e testá-lo num dispositivo físico. O custo de desenvolvimento torna-se muito menor face aos dispositivos lógicos fixos, podendo sempre alterar o circuito sempre que for necessário alterar algum requisito, bastando reconfigurar o dispositivo. [26]

Até ao momento, houve uma grande evolução nos dispositivos digitais reconfiguráveis, no entanto no domínio da eletrónica analógica reconfigurável, têm-se notado um avanço mais lento, contudo este encontra-se em evolução e será o principal tema deste trabalho.

No domínio digital, os dispositivos programáveis mais conhecidos são as *Field-programmable gat arrays* (FPGA's) que são chips de silício. A primeira FPGA comercial foi desenvolvida em 1985 por Ross Freeman, no entanto muitos estudos já tinham sido iniciados em 1969.[27] As FPGA's são dispositivos semicondutores baseados numa matriz de blocos lógicos configuráveis (*computational logic blocks* - CLBs) ligados entre si de forma configurável. [28]

Os circuitos reconfiguráveis no domínio analógico, demoraram mais tempo a aparecer no mercado, sendo só em meados de 1996 que começaram a surgir algumas ofertas. Estes já tinham sido divulgados em 1980 como *Field Programmable Analog Arrays* (FPAA).

A primeira FPAA disponível no mercado foi desenvolvida pela Zetex e foi denominada de *Totally Reconfigurable Analog Circuit* (TRAC). O desenvolvimento das TRACs surgiu de um trabalho académico pelo Professor David Grundy, que defende uma abordagem computacional estruturada para o desenho analógico. Desde então, apareceram diversas empresas a desenvolver FPAA's, tais como a Latice Semiconductor, a Motorola que lançou a família MPA1000, mas que no entanto em 2000 foi vendida surgindo o fabricante Anadigm (mais conhecido atualmente).[28][29]

A utilização de uma FPAA permite uma implementação rápida de sistemas analógicos em *hardware* real, apenas num único integrado. Utilizando as FPAA's é possível:

- Diminuir o tempo de fornecimento do dispositivo no mercado, reduzindo a complexidade de desenho analógico;
- Criar uma reconfiguração dinâmica do dispositivo, podendo adaptar-se ao ambiente em que se encontra a trabalhar;
- Desenhar produtos que realizem diferentes modos de operação;
- Reduzir o circuito, criando dispositivos mais pequenos.

As FPAA são utilizadas para uma vasta gama de soluções. Estas conseguem realizar funções de condicionamento de sinal, amplificação, filtragem, analisador de sinal (deteção de pico e sample and hold por exemplo). Poderão existir algumas necessidades da parte do utilizador, em que a FPAA não consiga responder, sendo necessário recorrer a *hardware* externo ligado à FPAA para responder a essas necessidades.

O processo de conceito de produto, desenho e fabricação de circuitos integrados que combinem milhares de transístores e interligá-los num único *chip* é denominado de *Very-Large-Scale Integration* (VLSI) [30]. Este processo por norma é muito demorado até que o produto esteja disponível para o utilizador final, podendo levar anos. Utilizando uma FPAA em vez do processo VLSI, torna-se muito mais vantajoso no tempo de conceção, tal como se pode analisar na Figura 36. No entanto a utilização de uma FPAA irá provocar algumas

ineficiências relativamente ao processo VLSI, tais como a diminuição da largura de banda e o aumento do consumo energético. [31]

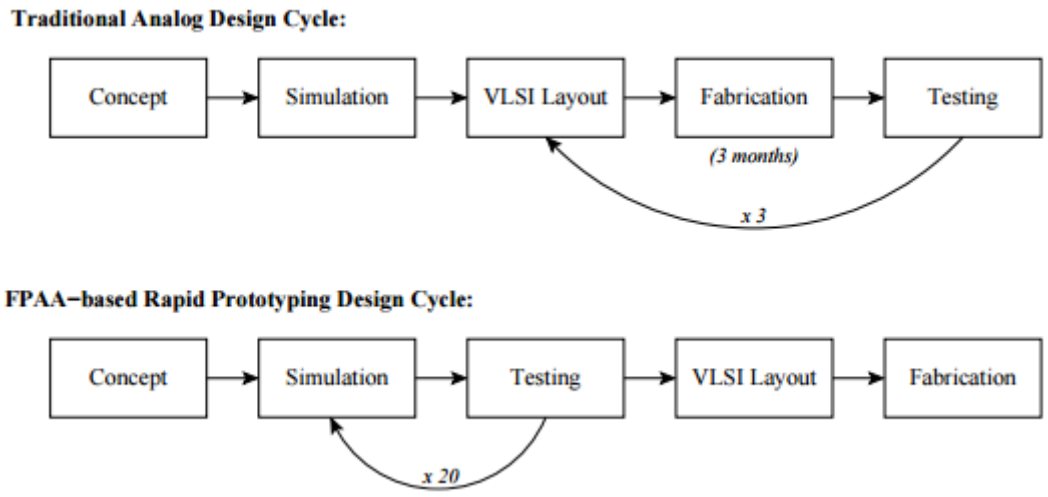


Figura 36 Comparação de utilização VLSI e FPAA em projetos [31]

Na Figura 36 é possível verificar que a utilização de uma FPAA para desenvolvimento de um projeto analógico, torna-se muito mais vantajoso perante o método tradicional, podendo realizar vários testes práticos (juntamente com simulação do circuito) antes da fabricação do circuito. No método VLSI, o teste apenas é possível após estar fabricado o circuito integrado. Se for necessário realizar algum ajuste após a realização dos teste, no método VLSI é necessário desenvolver um novo desenho, esperar o tempo de fabrico e de seguida realizar novamente os testes, no método de utilização de FPAA à medida que necessário realizar alguma alteração no produto basta implementar uma nova configuração na FPAA. O pré-desenvolvimento e o pré-teste com a utilização de FPAAs reduzem a implementação de circuitos analógicos de meses para dias. Embora as desvantagens que a FPAA possui, tal como falado anteriormente, as vantagens de flexibilidade, versatilidade, e adaptação de funcionalidades são compensadas para o desenvolvimento de produtos.

Os elementos mais importantes de uma FPAA são os blocos analógicos configuráveis (CAB – *Configurable Analogue Blocks*) e a interconexão da rede programável. Para implementar as funções analógicas desejadas, é necessário carregar no *shift-register* da FPAA uma determinada configuração de bits. Os blocos analógicos têm parâmetros que podem ser configurados para se adaptar à aplicação a desenvolver. A interconexão de rede é programada de forma a alternar as ligações dos sinais e dos blocos. Na Figura 37 encontra-se ilustrada a estrutura básica de uma FPAA.

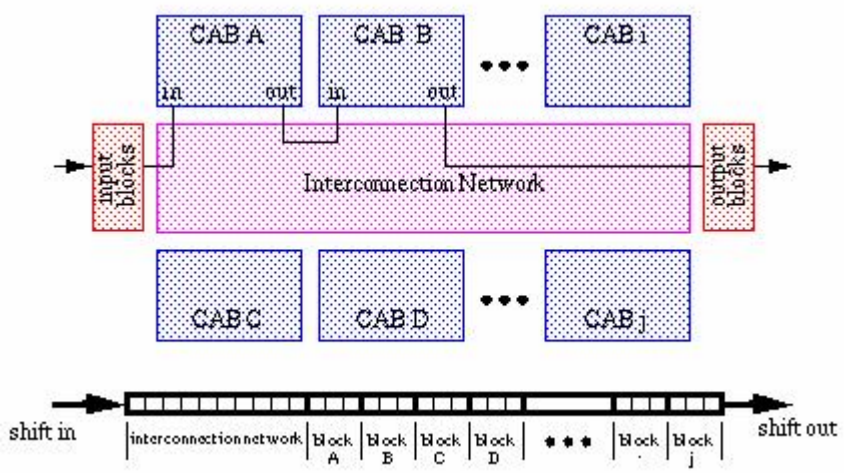


Figura 37 Estrutura básica de uma FPAA [35]

Tal como se pode verificar, na Figura 37, os blocos de entrada e de saída são ligados através das ligações realizadas pela rede de interligações, passando por cada CAB, para realizar uma determinada ação na aplicação. Todas as configurações são realizadas através do *shift-register* tal como apresentado na figura anterior, sendo que alguns bits são específicos para definir a forma da rede de interligações e os restantes bits para definir a função de cada CAB. Cada CAB pode implementar várias funções de processamento de sinal analógico, tal como amplificador de sinal, integrador, diferenciador, logarítmico, multiplicador, comparador, entre outros e deverão ser configurados de acordo com a aplicação desejada. As FPAAs podem ser classificadas em duas categorias, podem ser dispositivos de tempo contínuo ou dispositivos de tempo discreto.

As FPAAs de tempo discreto são desenvolvidas com a tecnologia de condensadores comutados ou comutação de corrente. Na Figura 38, encontra-se ilustrada a tecnologia de condensadores comutados, que têm como objetivo emular uma resistência variável, utilizando um condensador e a comutação de dois interruptores, de modo a controlar o fluxo de carga do condensador (sendo que o fluxo de carga controla a corrente) e por sua vez controlar a resistência.

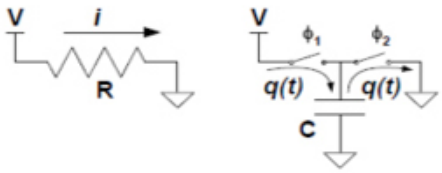


Figura 38 Tecnologia de condensadores comutados [33]

Na Figura 38 encontra-se ilustrado do lado direito a tecnologia de condensadores comutados e no lado esquerdo o circuito equivalente dessa tecnologia. Quando o interruptor Ø1 se encontra fechado e o interruptor Ø2 se encontra aberto, o condensador é carregado com a tensão V. A carga armazenada no condensador pode ser definida pela seguinte equação:

$$q=C.V$$

Quando o interruptor Ø1 se encontra aberto e o interruptor Ø2 se encontra fechado, o condensador é descarregado. Se a frequência de comutação dos interruptores for dada por Fs, a corrente que flui através do circuito pode ser determinada pela seguinte equação:

$$I=q/t=q.Fs= C.V. Fs$$

Sendo a frequência de comutação (Fs) bastante elevada, é possível considerar o processo como contínuo definindo uma resistência equivalente (R). O valor da resistência equivalente é definida pela seguinte equação:

$$R=V/I=V/(C.V.Fs)=1/(C.Fs)$$

Conclui-se então que para variar o valor da resistência basta variar a frequência de comutação dos interruptores. [32][33] Especialmente para filtrar sinais, esta tecnologia é bastante utilizada devido à exatidão nas respostas em frequência, assim como devido à boa gama dinâmica e linearidade.

Uma vez que esta tecnologia realiza amostragem de sinais contínuos, o teorema de Nyquist é bastante relevante para o sistema. O teorema de Nyquist define a relação entre a taxa de amostragem e a frequência do sinal medido, sendo que a taxa de amostragem deverá ser duas vezes superior à frequência máxima do sinal medido. De modo a evitar a sobreposição espectral (conhecido como *aliasing*) entre réplicas do espectro na banda base, a largura de banda do sinal deve ser limitada a menos de metade da frequência de amostragem. Assim sendo, esta tecnologia, fica limitada pela frequência de *clock*, para que consiga representar um sinal no domínio discreto. A frequência de *clock* mais comum nas FPAA's baseadas nos condensadores comutados é de 1MHz, assim sendo a FPAA ficará limitada a uma frequência de amostragem de sinais de entrada de 500kHz, para que os sinais não se sobreponham. [33] [36]

As FPAAs de tempo contínuo incorporam uma matriz de componentes fixos, tais como amplificadores operacionais e/ou transístores, e estes são interligados entre eles através de

comutadores. Para realizar a configuração dos comutadores, é utilizado um controlador externo, que envia dados para registos digitais de modo a implementar diferentes funções na FPAA. As FPAAs de tempo contínuo ultrapassam as desvantagens encontradas nas que utilizam a tecnologia de tempo discreto. Uma vez que o sinal analisado não é por amostragem, a frequência não fica limitada pelo teorema de Nyquist, não sendo necessário implementar o filtro de *anti-alising*, permitindo trabalhar com sinais de largura de banda maior. Mas tal como qualquer tipo de tecnologia, este também possui algumas desvantagens. A rede de comutadores, introduz impedâncias parasitas no sinal o que irá limitar a largura de banda e adicionar ruido ao sistema. [31]

3.6. FUNCIONAMENTO DE FPAA

A informação presente neste subcapítulo teve como base os documentos [41] [45]. A Anadigm, tal como falado anteriormente, é uma empresa que iniciou em 2000 com apoio da Motorola, pelo que desenvolve *hardware* e *software* pré-qualificado que permite que circuitos analógicos complexos possam ser implementado em processadores de sinal analógico programáveis. Para além do desenvolvimento de FPAAs, a Anadigm desenvolveu o seu produto, os *Dynamically Programmable Analog Signal Processors* (dpASP), estes são baseados na tecnologia de comutadores de condensadores.

A grande diferença entre os dispositivos FPAA e os dpASP nota-se no tipo de configuração do dispositivo. As FPAA são consideradas como reconfiguração estática enquanto os dpASP são de reconfiguração dinâmica. Nas FPAA's, é necessário fazer reset ao dispositivo para carregar uma nova configuração, enquanto que os dpASP, podem ser reconfigurados durante o seu funcionamento.

Os dpASP são dispositivos de configuração dinâmica em tempo real que permitem que as suas funcionalidades sejam reconfiguráveis nos sistemas através do *software* de configuração ou através de um microcontrolador/processador. Uma dpASP pode ser configurada para implementar múltiplas funções analógicas ou adaptar-se durante o seu funcionamento de modo a manter a sua precisão de operação necessária. São muito utilizados para aplicações que necessitam de um processamento analógico rigoroso em condicionamento de sinal, filtragem, aquisição de dados e controlos de malha fechada.

Para realizar a configuração da dpASP, a Anadigm disponibiliza um *software* próprio, o AnadigmDesigner2 EDA. Utilizando o *software*, o projetista pode construir funções complexas utilizando módulos analógicos configuráveis (CAMs) como blocos. Utilizando este *software*, realizar o desenho do sistema poderá ser feito em poucos minutos (dependendo da sua complexidade) permitindo completar sistemas analógicos de forma rápida, simulando-os de forma a perceber o estado final do sistema.

Para facilitar a introdução ao desenvolvimento de dispositivos, a Anadigm fornece *Kits* de desenvolvimento que utilizam a FPAA AN231E04 (dpASP). Na Figura 39 encontra-se a placa de desenvolvimento (AN231K04-DVLP3) utilizada para o desenvolvimento deste trabalho.

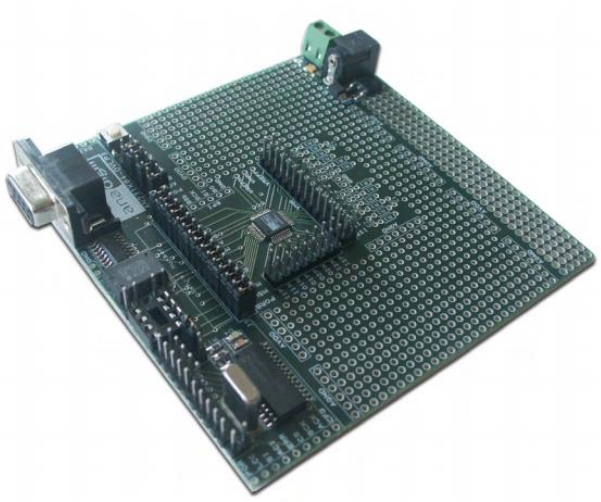


Figura 39 Anadigm 5Volts *Kit* de desenvolvimento

Tal como indicado anteriormente, esta FPAA pode ser considerada de operação dinâmica, pelo que permite que seja reprogramados alguns parâmetros durante a execução do código, permitindo realizar ajustes de diversos parâmetros para que o equipamento se adapte ao objetivo final pelo qual foi desenvolvido. As alterações possíveis de realizar *on-the-fly* poderão ser simples, tais como, o ajuste de ganhos ou frequências de corte, mas também poderão ser mais complexas, tal como a alteração do comportamento global do sistema. Para que a FPAA consiga ser configurada durante o seu funcionamento, torna-se necessário que este seja configurada como *Slave*, existindo um equipamento *Master* (um microcontrolador, por exemplo) capaz de realizar a configuração do mesmo utilizando comunicação SPI para

o configurar. Na Figura 40 encontra-se um exemplo de configuração para que o dpASP funcione através de operação dinâmica.

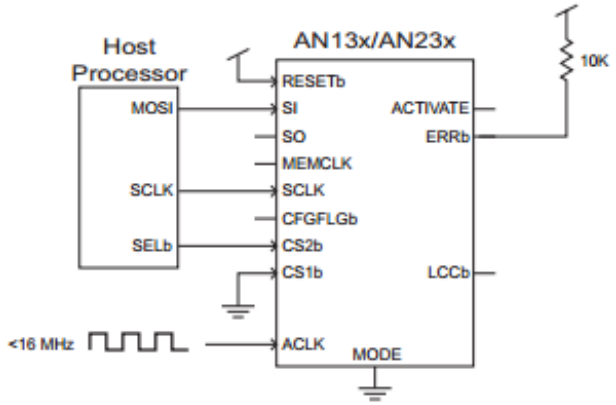


Figura 40 Configuração para operação dinâmica

Na sistema anteriorente apresentado, após a FPAA ser alimentada, ficará a aguardar que seja recebido uma sequência de configuração enviada através do microcontrolador (denominado de *Host Processor* na figura anterior). Para o sistema iniciar a configuração do dispositivo, é necessário que o microcontrolador selecione o dispositivo, para isso utiliza o pino CS2b (colocando a linha a 0V) para indicar que vai iniciar a comunicação, ficando a fpaa à escuta de dados provenientes da porta série (MOSI). Após a configuração estar realizada,o dpASP ficará disponível para processar os sinais analógicos sem necessidade de reeniciar o sistema.

Estes dispositivos tambem podem operar em modo estático, sendo configurados apenas uma única vez, sem que o código seja ajustado em caso de necessidade. Para tal, o dpASP, deverá ser ligado ao uma memória, com configuração através de SPI. Na Figura 41 é possível verificar a configuração para operação estática do dpASP.

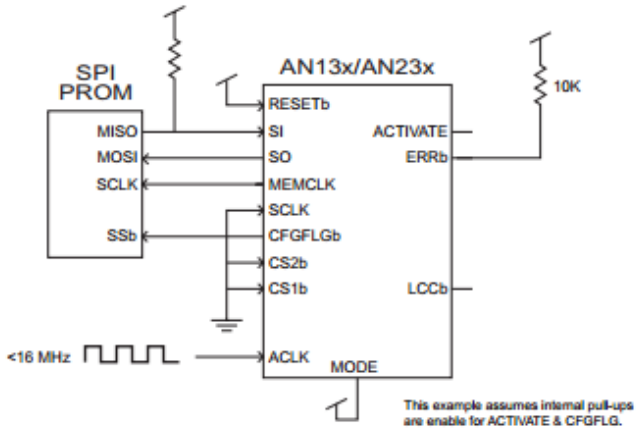


Figura 41 Configuração para operação estática

Após o sistema ser alimentado, a dpASP irá iniciar a leitura da configuração na memória *programmable read-only memory* (PROM) utilizando a comunicação *Serial Peripheral Interface* (SPI) . Estando o sistema alimentado e pronto para ser configurado, o pino CFGFLGb ficará a 0V, que fará com que a memória PROM fique selecionada. Após a selecção da memória, o dpASP envia um comando fixo de leitura da memória através do SO, obtendo a resposta da configuração através do SI. Após a configuração estar devidamente executada, o sistema irá iniciar o processamento do sinal analógico sem necessidade de reeniciar o dispositivo.

Uma vez que o dpASP, possui uma memória volátil, quando se retira a alimentação do mesmo, é necessário realizar novamente a programação do dispositivo. No caso dos sistemas com operação dinámica, o microcontroaldor, deverá programar (autonomamente ou através de configurações enviadas para o microcontrolador) o dispositivo, no caso dos sistemas com operação estática, o dpASP irá configurar-se, lendo sempre a configuração escrita na memória PROM.

A placa de desenvolvimento (AN231K04-DVLP3) está preparada para ser configurada para o modo de operação dinámico e estático. Por defeito toda a configuração realizada é executada dinamicamente, sendo utilizado um microcontrolador PIC16F876A. O microcontrolador comunica com a FPAA através de SPI, sendo o microcontrolador o responsável por realizar a programação da FPAA. Para que seja possível realizar uma configuração da FPAA através do *software* da Anadigm, será necessário ligar o computador ao dispositivo através de um conversor USB-RS232. Existindo um protocolo definido pelo

fabricante da placa, o *software* irá enviar as configurações realizadas no software para o microcontrolador, ficando este responsável por iniciar a comunicação por SPI de forma a programar a FPAA.

A funcionalidade de programação que a placa de desenvolvimento fornece, permite uma rápida e fácil configuração da FPAA, no entanto, para um sistema com multiplas FPAA's não se torna viável ter um sistema com diversos microcontroladores, quando na verdade é necessário apenas um dispositivo que seja capaz de programar cada uma da FPAA. Para desenvolvimento de um produto final, a utilização de um *KIT*, não se torna vantajoso, sendo necessário perceber como é feita a programação da FPAA através do microcontrolador. Analisando todo o sistema percebe-se facilmente que o microcontroldor presente no *KIT* está a realizar a interface entre o computador e a FPAA, sendo esta programada através de comunicação SPI. Assim sendo, de forma a substituir o microcontrolador *Programmable Intelligent Computer* (PIC) para realizar a programação, é possível realizar a programação do dpASP direatamente através de comunicação SPI utilizando um outro microcontroldor externo à placa de desenvolvimento. Contudo, estes métodos são meramente dinámicos, havendo a necessidade de um módulo externo iniciar a configuração. Para que o sistema seja capaz de ser configurado estaticamente (utilizando uma memória PROM), a placa de desenolvimento possui um *socket* pronto para incluir uma memória, sendo necessário inicialmente programar a memória com a configuração da FPAA, sendo a FPAA completamente autonoma (integrada com a memória) na realização da configuração sempre que reeniciada a alimentação.

Para definição do tipo de configuração, a placa de desenolvimento possui alguns *jumpers* que selecionam o tipo de configuração que será selecionado (configuração estática, dinámica através do microcontrolador PIC interno ou dinámica através de um microcontrolador externo).

4. PROPOSTA DE SOLUÇÃO

Face às dificuldades de acesso que as FPAA's possuem para que o teste aos sistemas seja realizado de forma simplista, no presente capítulo serão fornecidas diversas propostas de solução de forma a responder aos problemas. Será dada uma posta de verificação funcional de circuitos analógicos flexíveis, uma proposta de solução para a dificuldade de desenvolvimento de projeto com uma FPAA, assim como será dada a resposta para que o acesso aos testes do sistema não seja incapacitado, devido à falta de acesso ao *hardware*.

4.1. PROPOSTA DE VERIFICAÇÃO FUNCIONAL DE CIRCUITOS ANALÓGICOS FLEXÍVEIS

Nos circuitos flexíveis, para além da grande capacidade de alterar o circuito através de configuração, torna-se também relevante ter a capacidade validar que o circuito desenvolvido está a funcionar para a missão pretendida. De momento, as FPGA's têm a capacidade de controlar e observar os nós do circuito de missão através da infraestruturas IEEE1149.1, tornando-se assim uma grande vantagem para ajudar na realização da depuração do circuito. Esta funcionalidade é extremamente relevante para circuitos que se encontram interligados com outros dispositivos, sem capacidade de isola-los individualmente, e para dispositivos que tenham o acesso externo (pinos) completamente isolados, ou muito pequenos, onde impossibilita o acesso direto. Até à data da realização deste trabalho, as FPAA's não adotaram nenhuma estratégia de acesso para observação e controlo dos nós do circuito de missão de forma indireta tal como ilustrado na Figura 42 (sistema atual). Para tal, este trabalho será focado no desenvolvimento de uma aplicação que permita tornar uma FPAA com as mesmas características de apoio à depuração tal como uma FPGA, no entanto com utilização da infraestrutura IEEE1149.4 com capacidade de controlo e observação de sinais analógicos.

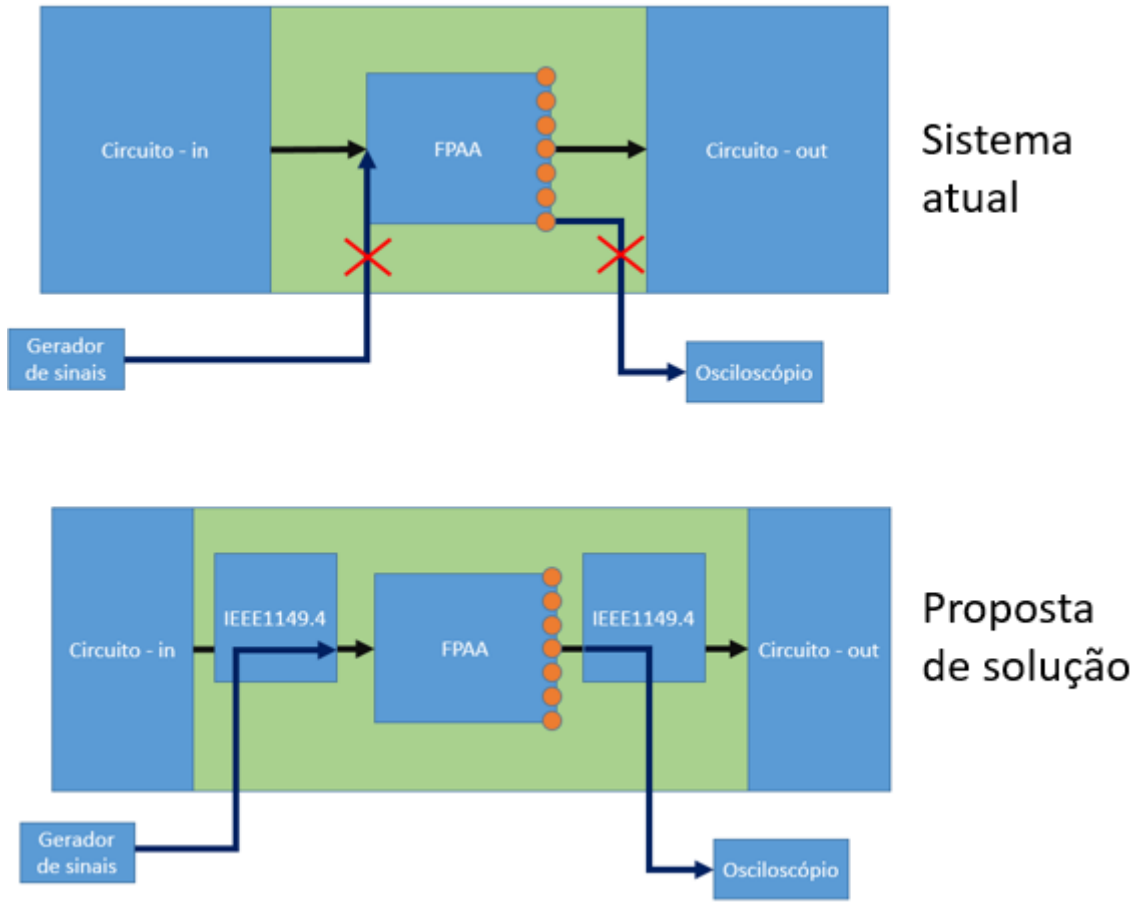


Figura 42 Sistema atual vs Proposta de solução

Após uma análise das soluções existentes no mercado de aplicações e soluções capazes de controlar os módulos de interface IEEE1149.4, verificou-se que os seus custos são elevados, o que levou a um estudo de uma nova solução capaz de realizar o controlo do módulo utilizando um equipamento de controlo. Não sendo apenas o custo um dos obstáculos, também existia a necessidade de desenvolver uma aplicação que tivesse interface com o *software* de controlo da infraestrutura, o que poucas das aplicações desenvolvidas até ao momento teriam disponível uma interface de controlo do dispositivo. Face às necessidades, foi utilizado um dispositivo de aquisição e controlo, capaz de implementar a comunicação para interface com os módulos STA400. O dispositivo escolhido para o controlo foi o NI USB 6008 e foi desenvolvido pela National Instuments. Sendo este um dispositivo multifuncional, com capacidade de 8 entradas analógicas digitais, 2 saídas analógicas e 12 entradas/saídas digitais, tendo a possibilidade de utilizar como contador de 32 bits. Utilizando os pinos digitais e analógicos do módulo, é possível ter controlo dos pinos TAP

e ATAP dos módulos SCAN STA400, permitindo assim implementar o protocolo para realizar o controlo e observação dos módulos.

Tal como foi analisado anteriormente, existem diversas formas de realizar a programação da FPAA que se encontra na placa de desenvolvimento. Esta poderá ser programada através do microcontrolador PIC existente na placa de desenvolvimento, realizando uma comunicação por RS232 à PIC, ou poderá ser programada através de comunicação SPI utilizando um módulo externo. Os modos de configuração indicados anteriormente são considerados como dinâmicos, pelo que a configuração através de acesso a uma memória PROM é um modo de operação estático, não sendo esta a configuração a ser utilizada neste projeto, havendo a necessidade de configurar e testar o dispositivo.

Uma vez que um dos objectivos deste projecto passa pela execução de um *software* único para configuração e teste do dispositivo, será necessário interligar um sistema que implementa-se a comunicação através de SPI para programação do dispositivo, evitando assim a utilização do softwre da ANADIGM para realizar a configuração da FPAA. Uma vez que a placa de desevolvimento se encontra configurada de fábrica para realizar a configuração da FPAA através do microcontroaldor PIC, será necessário alterar a configuração dos *jumpers* para que permita realizar a configuração através de comunicação SPI externa.

Uma das necessidades para realizar a configuração do dispositivo através do módulo SPI externo, passa por criação do projecto no *software* Anadigm, podendo simular o circuito para o qual está a ser desenhado. Feita toda a configuração, o *software* permite exportar o código hexadécimal da mesma, sendo assim esse o código transmitido para a FPAA através do módulo externo por comunicação SPI. Idealmente, tal como os dispositivos existentes no mercado, a programação da FPAA deveria ser realizada utilizando toda a infraestrutura IEEE1149.1, sendo necessário implementar a configuração de SPI através da infraestrutura, no entanto este trabalho não tem como objetivo realizar esta ação, podendo deixar esta opção para desenvolvimento futuro.

Desta forma o controlo, observação e programação da FPAA será realizada através de um *software* único permitindo assim, realizar a verificação funcional do circuito flexível.

4.1.1. ARQUITETURA DO HARDWARE DO SISTEMA FINAL

Após ser feita a análise de requisitos do sistema e de acordo com o equipamento disponível, foi desenhada a arquitetura do *hardware*, de modo a responder a todos os requisitos definidos. Uma vez que o objetivo principal passa pela reutilização da infraestrutura IEEE1149.4 para apoiar as tarefas de apoio à depuração em FPAAs, levou que o sistema fosse não só analisado à saída do dispositivo mas também à entrada de modo a perceber a relação entre ambas. Para tal, serão utilizados dois dispositivos com infraestrutura IEEE1149.4 (os STA400) ligados a uma FPAA. De momento, os dispositivos atuais não permitem configurar os seus pinos de forma a separa-lo do circuito atual, para que possa ser injetados sinais e observados à saída do dispositivo de forma a validar o sistema. Utilizando as infraestruturas IEEE1149.4, o circuito terá a capacidade de funcionar de acordo com a sua missão, mas também terá a capacidade de se comutar do circuito atual de forma a apoiar à depuração. Para que seja possível programar a FPAA de forma dinâmica, será utilizado um microcontrolador com capacidade de comunicação com o computador, implementando um protocolo específico de configuração da FPAA através de SPI. Para implementação do controlo do STA400, foi idealizado utilizar um sistema de aquisição e controlo. Na Figura 43 encontra-se ilustrada a arquitetura do *hardware*.

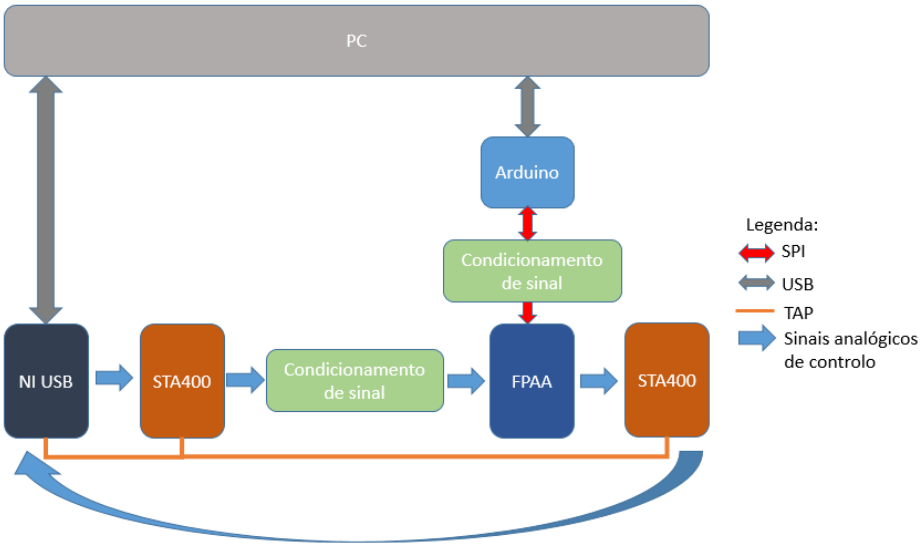


Figura 43 Arquitetura do hardware

Tal como pode ser analisado na figura anterior, utilizando o computador, será possível programar, controlar e observar o estado da FPAA. Para tal o computador será ligado ao Arduino através de uma porta universal serial bus (USB) e internamente tratará a mensagem

enviada pelo computador para programar a FPAA via SPI. O sistema de aquisição e controlo também é controlado por computador, utilizando um USB para interligação dos dois sistemas. O sistema de aquisição e controlo (NI-USB 6008) terá a capacidade de implementar o protocolo para controlo do TAP de modo a controlar o estado dos STA400, dando a possibilidade de funcionamento normal do sistema, de realizar um controlo específico do mesmo, ou até de observar o estado atual de cada pino da FPAA. O módulo da National Instrumments (NI) também será capaz de injetar sinais analógicos para o STA400 de forma a coloca-los nos pinos da FPAA. Após a FPAA processar todos os sinais serão enviados para a STA400 à saída, sendo os sinais observados através do módulo de aquisição e controlo. Considerando que o sistema da FPAA não tem os pinos acessíveis, torna-se necessário obter o estado e controlar o estado de cada pino da FPAA, de forma a validar que o sistema se encontra a funcionar para o qual foi configurada. Para tal, os módulo STA400 permitem, através da interface ATAP, realizar essas ações utilizando para controlo e observação o módulo NI. Visto que a FPAA, é o único sistema que utiliza níveis de tensão LVTTL, torna-se necessário implementar dois sistemas de condicionamento de sinal sobre os sinais provenientes no módulo STA400 e sobre a comunicação SPI proveniente do Arduino.

4.1.2. ARQUITETURA DO *SOFTWARE*

Para implementação do sistema de teste e configuração, será desenvolvida uma aplicação em *Labview*. Sendo o *Labview* um ambiente gráfico de desenvolvimento criado especificamente para acelerar a produtividade de engenharia, este encontra-se muito utilizado no sector industrial. [37] Sendo este um sistema de programação gráfico, torna-se simples o desenvolvimento do código para sistemas mais complexos de engenharia, ajudando a reduzir os tempos dos testes oferecendo informação relevante com os dados adquiridos. São diversas as empresas que utilizam os sistemas de teste, de controlo e aquisição através do *Labview*, empresas essas tais como a National Grid UK, a Subaru ou a AIRBUS [38]. A Subaru, de modo a garantir o nível de controlo de qualidade no seu primeiro veículo híbrido, criou condições de teste extremas utilizando o *Labview*. Sendo o tempo um fator bastante importante para a produção, nos mercados em que a competitividade é bastante elevada, a Subaru conseguiu passar os seus sistemas de simulações ambientais para os testes rigorosos físicos poupando muito tempo de desenvolvimento com a utilização da programação em *Labview*. Para além do *Labview*, a National Instrumments disponibiliza um

software de gestão de testes, o *TestStand*. O *TestStand* permite gerir os testes, sequenciando o código de forma a registar os resultados de cada passo do teste, gerando no final do mesmo um relatório para cada produto [39]. A integração do *TestStand* com o *Labview* permite realizar passos de testes simples tais como deteção de tensões de alimentação, validação de consumos, testes de interfaces com o utilizador, entre outros. O *TestStand* é muito utilizado em diversos sectores, tais como no sector militar, aeroespacial, automóvel, médico, entre outros. A grande vantagem deste *software* é a facilidade de reutilização de código para produtos diferentes, permitindo realizar não só execuções sequenciais como também testes paralelos para maximizar a produção [40].

Considerando estes os ambientes de programação mais comuns para o desenvolvimento de testes de sistemas, a utilização do *Labview* será a base deste trabalho. Sendo o módulo de aquisição e de controlo um equipamento da National Instrumments, a utilização de *Labview* torna-se obrigatória, no entanto, a utilização deste equipamento de aquisição e controlo não se torna uma obrigação, podendo ser facilmente substituído por um outro qualquer sistema de aquisição e controlo, utilizando por exemplo um microcontrolador.

O *software* a ser desenvolvido terá as funcionalidades apresentadas na Figura 44.

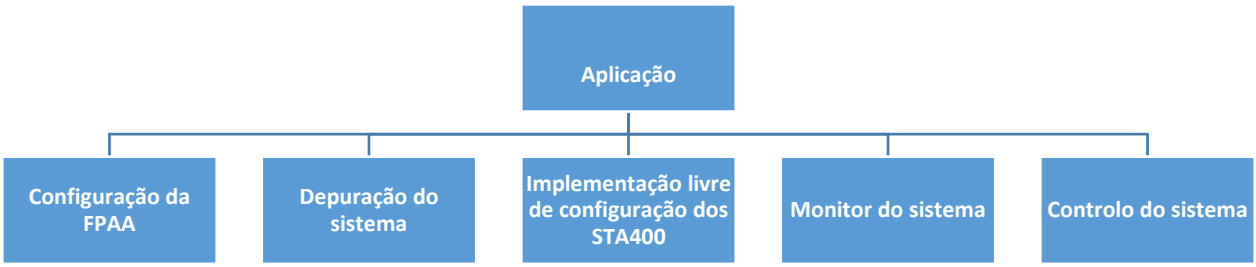


Figura 44 Funcionalidades do *software* principal

Tal como ilustrado anteriormente, o *software* terá a possibilidade de realizar a configuração da FPAA, sendo esta configuração realizada através da comunicação com o módulo que implementa a comunicação por SPI (o Arduino). A configuração da FPAA será gerada utilizando o carregamento direto do ficheiro gerado pelo *software* Anadigm de simulação e

de configuração da FPAA. Após a FPAA estar devidamente configurada para realizar a execução pretendida, torna-se importante realizar a depuração do sistema, pelo qual é possível analisar o estado de cada pino (na entrada e na saída) da FPAA, assim como força um estado conhecido em cada um dos pinos, de modo a validar que o sistema está configurado devidamente tal como foi previsto em simulação. O módulo de controlo do sistema permite aplicar sinais analógicos em pinos específicos, assim como definir sinais digitais, configurando o estado dos *multiplexeres* (sistema core dos STA400). O monitor do sistema servirá para adquiridos em tempo real sinais gerados no sistema, podendo ter maior perceção a nível gráfico do estado do mesmo. De modo a tornar o *software* mais abrangente, será implementado um módulo que dá a possibilidade do utilizador implementar a configuração dos módulos STA400 utilizando o protocolo definido na norma.

4.2. PROPOSTA PARA DETALHE DO PROJETO DE CONFIGURAÇÃO DA FPAA

Sendo as FPAA's bastante vocacionado para engenharia puramente analógica, por vezes existe alguma dificuldade para realizar a configuração da mesma sem que seja necessária a utilização de um *KIT* de desenvolvimento visto que a infraestrutura de programação já se encontra bem definida. No entanto, por vezes torna-se necessário isolar a FPAA do KIT de desenvolvimento de forma a desenvolver um circuito proprietário, pois tal como foi falado anteriormente, as FPAA's por norma incluem eletrónica externa para poderem realizar as funcionalidades necessárias. Para além da necessidade de isolamento, poderá existir também a necessidade de interligar mais do que uma FPAA, no entanto com utilização de um *KIT* de desenvolvimento seria obrigatório ter uma interface de programação por cada FPAA incluída no projeto, o que na verdade apenas uma será necessária para realizar a configuração, fazendo com que o sistema se torne demasiado grande.

De forma a tornar o desenvolvimento do projeto com FPAA's mais facilitado, propõe-se neste trabalho, desenvolver um guia de trabalho educacional, para que múltiplas FPAA's ou apenas uma FPAA possa ser configurada individualmente através de comunicação SPI (indiretamente) sem utilização do *KIT* de desenvolvimento.

4.3. PROPOSTA DE EXPERIMENTAÇÃO REMOTA

Dado que este trabalho, poderá ajudar a nível educacional, a perceber o funcionamento das infraestruturas IEEE1149.1 e IEEE1149.4, assim como permitir na ótica do utilizador configurar um circuito para a FPAA (de forma a conhecer melhor o conceito de circuitos analógicos flexíveis) validando o funcionamento do mesmo, torna-se relevante tornar o sistema facilmente acessível para qualquer utilizador. Para tal será desenvolvido um sistema, capaz de realizar a validação de circuitos analógicos flexíveis a partir de uma interface com ligação à internet. Permitindo assim que os utilizadores possam realizar testes ao sistema sem restrições de acesso físico aos dispositivos.

5. IMPLEMENTAÇÃO DA SOLUÇÃO

De forma a tornar possível a implementação do sistema de verificação funcional de circuitos analógicos flexíveis, foi necessário subdividir o sistema por diversas tarefas. Inicialmente foi analisado o modo de controlo do módulo com infraestruturas IEEE1149.4 (SCAN-STA400) utilizando o módulo da NI. De seguida foi analisado o modo de configuração da FPAA, utilizando o sistema por defeito (programação realizada pela PIC), de forma a validar como é que é realizada a configuração da FPAA através do *software* da Anadigm. Após a obtenção do conhecimento de configuração da FPAA através da configuração por defeito, torna-se importante desenvolver o sistema de configuração da FPAA através de um dispositivo externo, para tal será explicada a implementação detalhada do projeto de configuração da FPAA. Estando os dois sistemas completamente controláveis para realizar as ações pretendidas para o trabalho, interligaram-se os dois sistemas num único, de forma a torná-lo capaz de realizar a verificação funcional. Para finalizar serão apresentados os meios de acesso remoto desenvolvidos para permitir realizar o controlo, observação e configuração do sistema sem que haja restrições de acesso físico.

5.1. IMPLEMENTAÇÃO DE CONTROLADOR DO MÓDULO COM INFRAESTRUTURA IEEE1149.4

Para validação do funcionamento do módulo com infraestrutura IEEE1149.4, tornou-se necessário perceber como deveria este módulo ser interligado para que fosse possível colocá-lo em funcionamento, podendo validar o seu funcionamento do sistema core através da infraestrutura desenvolvida. Para validação do módulo STA400, foi decidido que se iria iniciar com um sistema apenas com um módulo STA400, avançando para um sistema com 2 STA400. Após a validação do funcionamento com 2 STA400, a aplicação de mais módulos, torna-se praticamente transparente, permitido realizar o controlo e observação de qualquer módulo através da infraestrutura IEEE1149.4. Assim sendo, para o primeiro

sistema, foi decidido que o módulo STA400 seria ligado diretamente ao módulo da NI, tal como ilustrado na Figura 45.

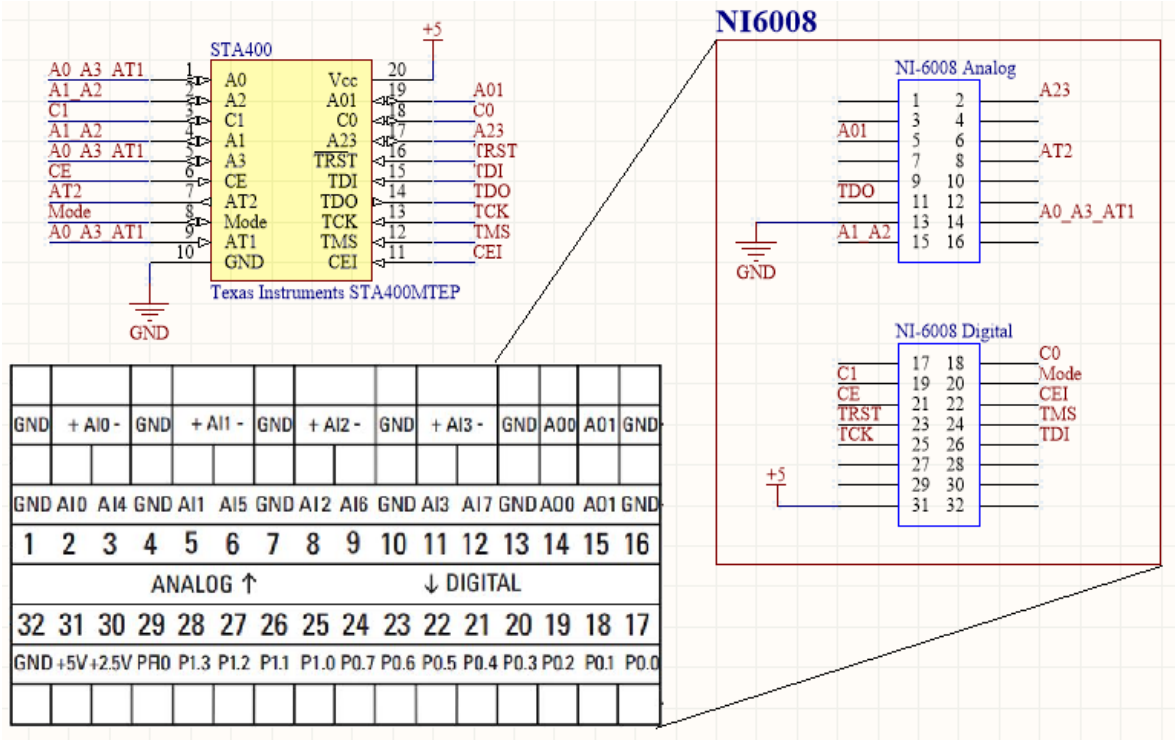


Figura 45 Esquema de ligação de um módulo STA400

Tal como pode ser analisado na figura anterior, o módulo STA400, encontra-se ligado diretamente ao módulo da NI, deste modo todo o controlo será realizado por esse módulo. Uma vez que o módulo da NI possui apenas 2 saídas analógicas, tornou-se necessário reutilizar esses sinais entre diversos pinos do módulo. Visto que o controlo deste será muito controlado, esta interligação entre os pinos do módulo não será prejudicial. Os pinos A01, A23, AT2 e TDO são ligados aos pinos analógicos de entrada do módulo NI, sendo possível perceber em qualquer instância o estado de cada um, tendo também um pequeno histórico dessa informação. Os pinos A0, A1, A2, A3 e AT1 serão ligados aos pinos analógicos de saída do módulo NI, permitindo aplicar valores entre 0 e 5V, dependendo do tipo de teste que se pretende realizar. Para controlo do sistema core do módulo STA400, foram utilizados os pinos C0, C1, CE, CEI e MODE. Uma vez que o estado do sistema core altera de acordo com os diferentes estado destes pinos (ver Tabela 7), torna-se necessário configurar os pinos digitais como digitais de saída para que estes possam ser controlados. Visto que o controlo TAP é realizado através dos pinos TDI, TMS, TCK e TRST, estes também serão ligado ao módulo da NI, através dos pinos digitais, estando estes configurados como saída.

Estando o sistema todo montado passou-se ao desenvolvimento do *software* para comunicar com o dispositivo. Tal como visto anteriormente, o controlador TAP é definido como uma máquina de estados finita e síncrona que altera o seu estado de acordo com as alterações dos sinais TMS e TCK, controlando assim a sequência de operação do circuito. Dependendo do estado de operação do circuito, poderá ser necessário definir o valor em alguns registos, para executar uma tarefa em específico, para tal utiliza-se o sinal de TDI juntamente com o TCK para que o controlador TAP adquira essa informação e a trate.

O processo de geração destes dados será realizado utilizando os pinos digitais do módulo da NI, onde se encontram ligados os sinais TMS, TCK e TDI. Visto que o essencial é validar o funcionamento do módulo através da execução deste teste, foi desenvolvida uma função que processa os dados que o utilizador pretende escrever para o respetivo canal (TDI ou TMS). Deste modo, o utilizador terá acesso a uma lista que poderá preencher de acordo com as necessidades tal como ilustrado na Figura 46.

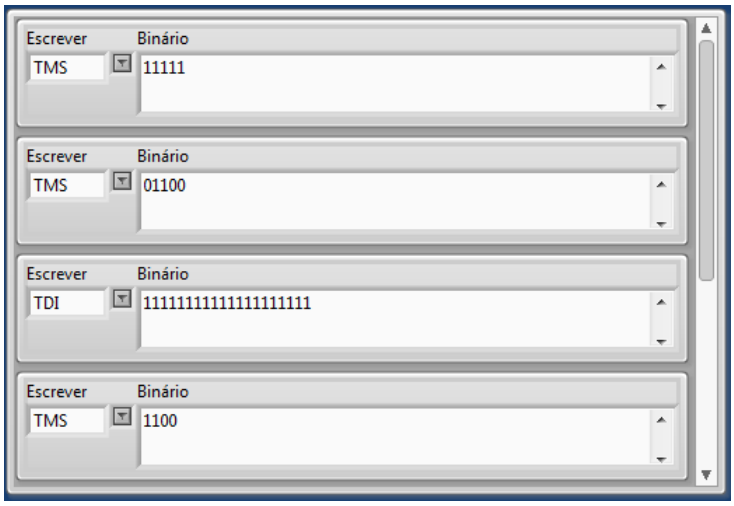


Figura 46 Lista de execuções pretendidas

Tal como se pode analisar na figura anterior, o utilizador terá acesso a uma lista onde deverá indicar o sinal para o qual pretende escrever e o código binário correspondente que irá escrever para esse sinal. Obtendo essa informação, a função irá processa-la gerando para o módulo da NI os sinais digitais correspondentes. O controlador TAP lê o dado presente no pino TMS e TDI quando existe um flanco ascendente no pino TCK, e considera-o válido quando existe um flanco descente. Assim, a função desenvolvida analisou o tipo de sinal a escrever e por cada dado binário irá gerar um flanco ascendente e um flanco descendente, mantendo o sinal de TMS ou de TDI dependendo da ação que se pretende executar.

Na Figura 47 encontra-se ilustrado os parâmetros de entrada e de saída que a função desenvolvida. A entrada *Escrever* indica se é TMS ou TDI e *Binário* é o dado que se vai escrever para o sinal *Escrever*. A saída da função *Dados(TMS-TCK-TDI)* corresponde à matriz dos diversos sinais a escrever. A ordem de escrita dos sinais para o módulo será executado de cima para baixo, sendo o sinal mais a esquerda o sinal de TMS, o do centro TCK e o da direita TDI (os três sinais são escritos ao mesmo tempo para o módulo).

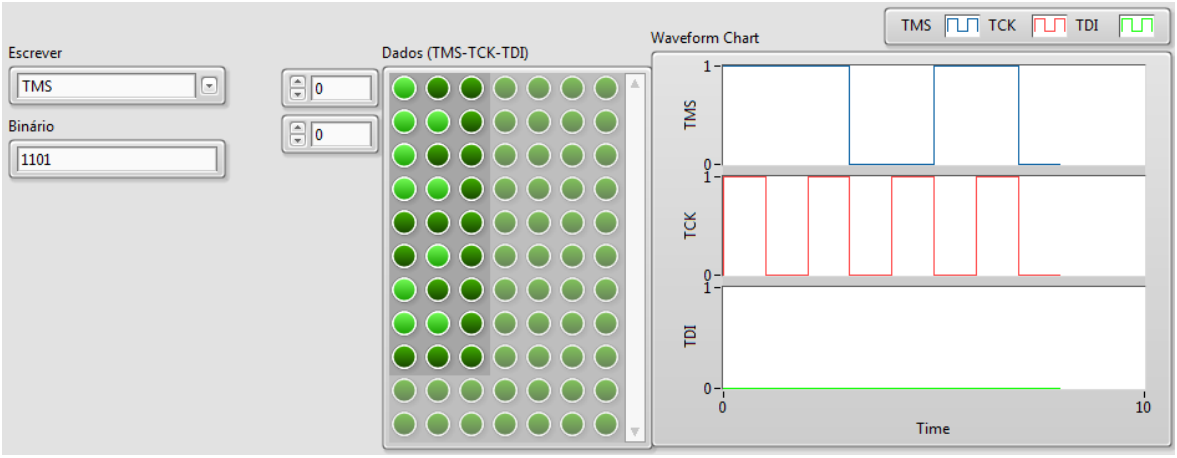


Figura 47 Entradas e saídas da função de gerador de sinais TMS-TCK-TDI

No exemplo apresentado pode-se verificar que o sinal que se pretende escrever é o TMS, com o valor binário 1101, onde irá gerar o deslocamento da máquina de estados para um outro estado qualquer, dependendo do estado onde se encontra antes da execução desta função. Após ser executada esta função, pode-se verificar que independentemente do momento, o sinal de TDI encontra-se a 0, sendo alterado apenas o sinal de TMS e de TCK. Visto que o sinal é entendido pelo controlador, quando há um impulso de TCK, para cada valor binário escrito é dado um impulso ao sinal de TCK. Sempre que o sinal de TCK gera um flanco descendente, o sinal de TMS altera para o valor seguinte a enviar. Nas figuras seguintes encontram-se o código desenvolvido para executar esta função. Tal como se pode verificar nas imagens seguintes, tanto a entrada *Escrever* como o *Binário* são strings, pelo que existe necessidade de converter valor binário em valor numérico. Para tal, utiliza-se uma função que converte o valor binário num *array* de *bytes* que traduz cada dado escrito em texto num valor decimal. Após obter o *array* de bytes é subtraído o valor decimal 48 (porque em *ascii*, o valor "0" corresponde ao valor decimal 48), gerando um novo *array* de dados com o valor em *double* (linha laranja nas figuras seguintes). Após ter a informação em *array* torna-se necessário perceber para que tipo de sinal se vai escrever, para tal, é colocado uma

estrutura *case* que recebe como parâmetro de entrada o dado *Escrever* onde irá executar os diferentes processos dependendo do tipo de sinal (Cada figura abaixo corresponde ao *case* específico).

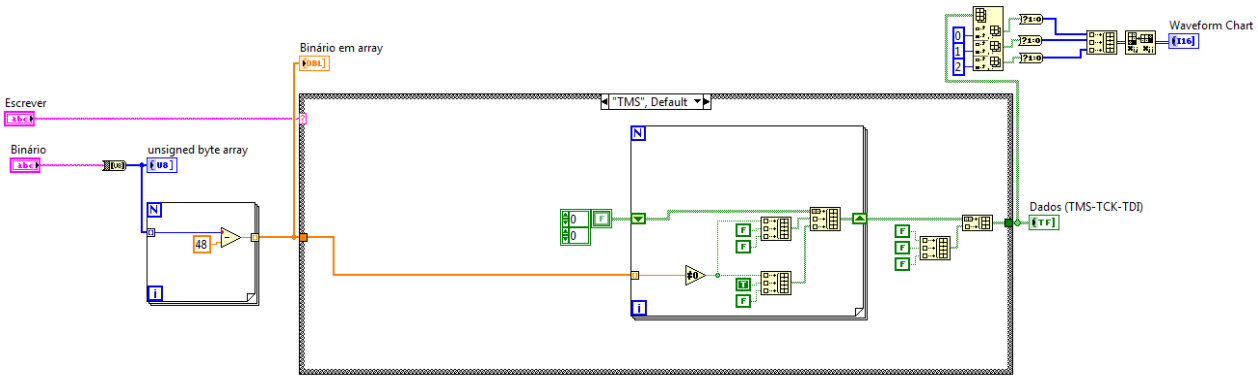


Figura 48 Função de geração de sinais para controlo do TAP-Caso TMS

No caso da função ser de escrita para o TMS, os dados do *array* são percorridos, e analisados se são iguais ou diferentes de 0 (no caso de serem diferentes de zero coloca o valor *True* na saída). Visto que para cada sinal gerado é necessário gerar um flanco descendente e ascendente, é criado uma matriz onde será escrito o valor de TMS e forçado o sinal de TCK a ter o valor lógico *Falso* no início e o valor *True* de seguida (note que para cada dado do *array* serão gerados estes sinais, forçando sempre o sinal de TDI a *False*). Para finalizar, é escrito o valor lógico *False* a todos os sinais. Neste momento, os dados já se encontram preenchidos na saída *Dados (TMS-TCK-TDI)*, sendo de seguida gerado o gráfico correspondente a cada sinal.

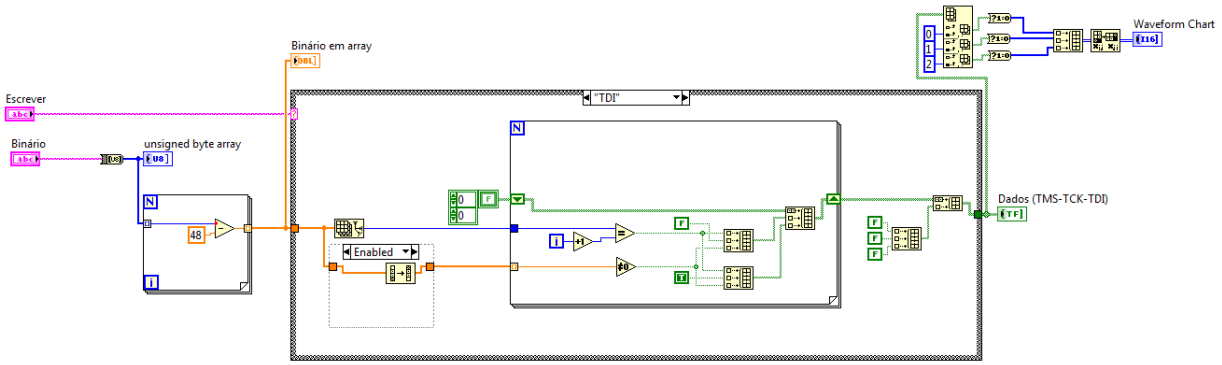


Figura 49 Função de geração de sinais para controlo do TAP-Caso TDI

No caso da função ser de escrita para o TDI, os dados do *array* são processados de igual forma ao caso TDI. No entanto quando os dados já se encontram no formato *double* em *array* (linha cor de laranja), estes são invertidos, uma vez que o sinal mais significativo deverá ser escrito em último no TDI. O dado no controlador será processado tal como se escreve, tendo a necessidade de realizar esta ação, antes de se escrever qualquer valor para o módulo. Tal como no caso anterior, o sinal de TDI era forçado a *False*, uma vez que se pretendia escrever no sinal TMS, neste caso o sinal de TDI irá variar, sendo forçado o sinal de TMS a falso (Exceto quando se pretende terminar o processo de escrita, visto que o sinal de TMS deverá ficara 1 para o controlador altere o estado da máquina de estados). Para cada sinal de TDI será também gerado um flanco descendente e ascendente, para tal é criado uma matriz onde será escrito o valor de TDI e forçado o sinal de TCK a ter o valor lógico *False* no início e o valor *True* de seguida. Para finalizar, é escrito o valor lógico *False* a todos os sinais.

Esta função será a base de todo este trabalho para permitir controlar o estado do módulo, permitindo assim ter acesso a todos os comandos da norma. Para além desta função, este *software* de teste possui diversos processos, que não serão apresentado neste relatório visto que o código é todo ele gráfico (todo o código será colocado em anexo num DVD entregue juntamente com este documento), tornando este trabalho com demasiadas imagens, sendo apenas representado uma imagem dos processos que são executados com a devida explicação.

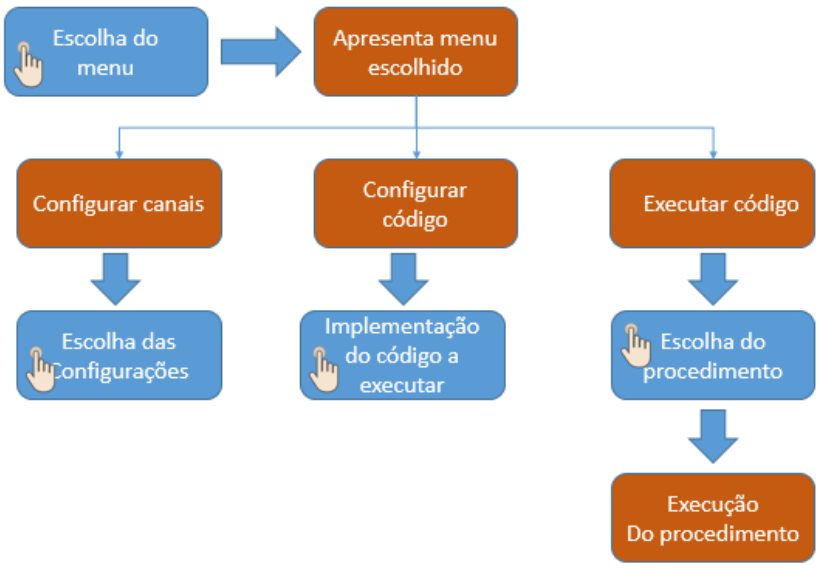


Figura 50 Processos de execução do *software* de teste STA400

O *software* desenvolvido permite que o utilizador escolha entre 3 menus, o menu de configurar canais, o menu de configurar código e o menu de execução do código. No menu de configuração de canais, tal como o nome indica, o utilizador poderá definir quais são os canais da NI que se encontram ligados aos canais do módulo STA400. No menu de configurar código, o utilizador terá a possibilidade de criar, editar ou eliminar procedimentos de teste. Os procedimentos desenvolvidos neste menu terão como base a lista apresentada anteriormente, onde o utilizador define os passos de execução dos sinais TDI e TMS, gravando esse procedimento com um nome. Todos os procedimentos serão apresentados numa lista para que, quando o utilizador escolher aceder ao menu *Executar código* tenha acesso a essa lista e escolha qual o procedimento que pretende executar.

O *software* desenvolvido tem o aspeto ilustrado nas figuras seguintes:

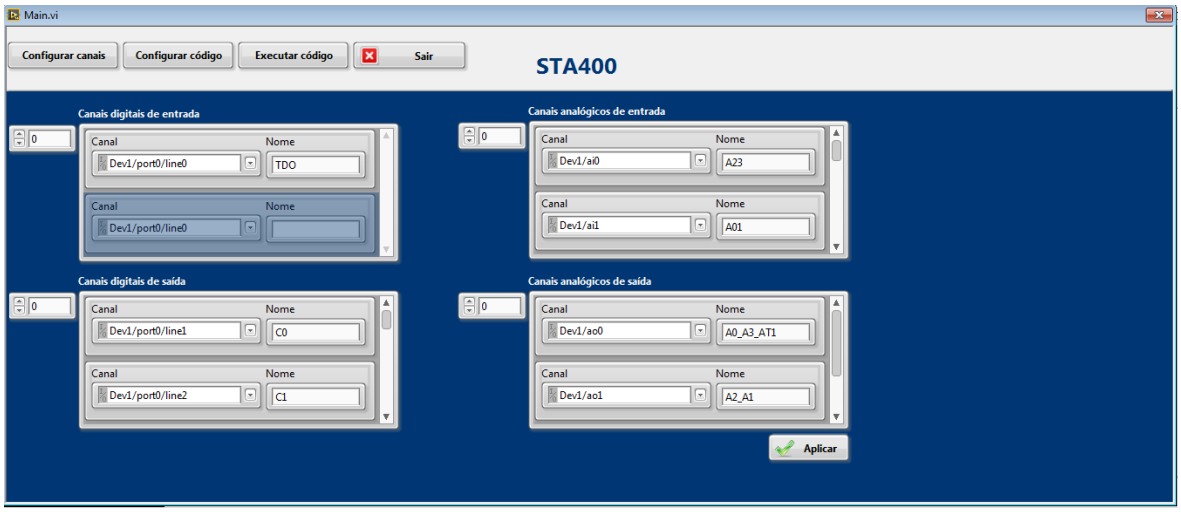


Figura 51 Menu configurar canais, *software* preliminar

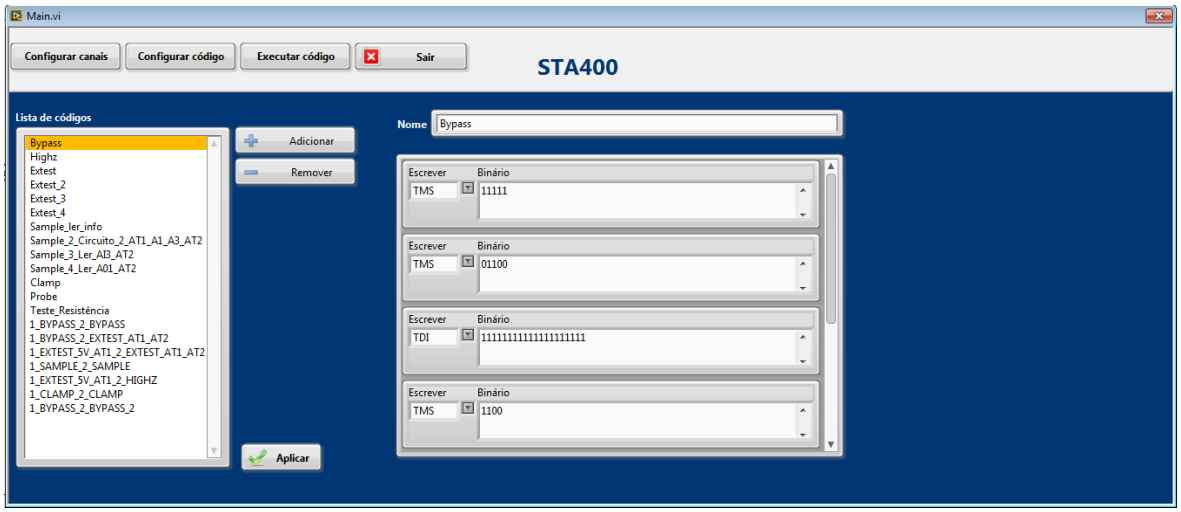


Figura 52 Menu configurar código, *software* preliminar

Após o utilizador dar início à execução do procedimento de teste, será apresentado um submenu idêntico ao da figura seguinte. Neste menu teremos a informação da máquina de estados do controlador, informação do estado dos sinais digitais TCK, TMS e TDI em tempo real assim como os dados adquiridos em A23, A01, AT2 e em TDO. De forma a poder realizar um controlo durante a execução do teste, também existe a possibilidade de controlar os sinais digitais C0, C1, Mode, CE, CEI, TRST e os sinais analógicos A0, A1, A2, A3 e AT1.

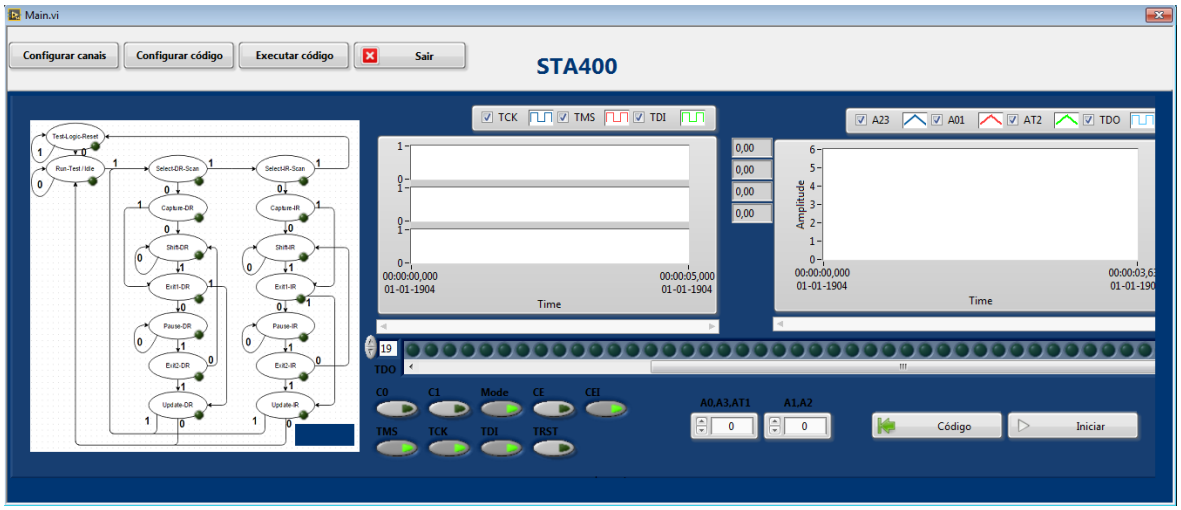


Figura 53 Menu Executar código, *software* preliminar

Para validar o funcionamento do sistema, foram executados váriso testes, de acordo com os comandos disponíveis no módulo STA400. Para tal foram criado os procedimentos indicados na tabela seguinte.

Tabela 9 Procedimentos desenvolvido para testes preliminares

Procedimento	Explicação
Bypass	Ligar o TDI ao TDO e analisar em TDO a transição de bits aplicados em TDI
Highz	Coloca o STA com os pinos em alta impedância
Extest	Coloca o A01 com a saída a 5V
Extest_2	Idêntico ao *Extest*, mas com a diferença de utilizar o 2º comando de E*xtest* definido pelo módulo STA400, colocando a saída a 0V
Extest_3	Idêntico ao *Extest_2*, no entanto liga AT1 ao AT2 através do AB2
Extest_4	Idêntico ao *Extest_3*, com a diferença de que liga AT2 a A01, obrigando a A01 ter 5V
Sample_ler_info	Adquirir no TDO, o estado de todos os canais
Sample3_ler_A3_AT2	Ler o valor que está a ser injetado em A3, em AT2
Sample_4_Ler_A01_AT2	Ler o valor de A01 em AT2. Não é possível porque o circuito fica desligado dos pinos. Pode-se forçar um valor na saída
Clamp	Lê o estado lógico dos pinos e mantem-nos

Probe	Idêntico ao *Sample_4*, no entanto já se torna possível ler o A01 uma vez que o circuito já está ligado aos pinos.

Os procedimentos gerados servem para realizar alguns dos testes que o STA400 permite. De seguida será apresentada a explicação de um dos procedimentos executados, pelo que os outros seguem o mesmo raciocínio, podendo ser analisados posteriormente no código entregue no DVD. O procedimento *Probe* será um dos procedimentos utilizados no trabalho final para analisar o estado de um pino específico através do pino AT2, para tal será apresentado e explicado o procedimento para cada dado enviado para o STA400.

Tabela 10 Explicação do procedimentos *probe*

Sinal	Dado
TMS	11111
TMS	01100
TDI	1111111111111111000
TMS	1100
TDI	00000000000000000000000100000000000000000000010
TMS	10000000000000000000000000000000

O primeiro sinal a ser enviado, é para o TMS com o dado 11111, isto irá fazer com que o sistema entre em *reset*. Independentemente do estado onde o sistema se encontra, o envio de cinco 1's para o pino do TMS, fará com que a máquina de estados passe para o *Test-Logic-Reset*.

Conhecendo o estado atual do sistema, é possível enviar o comando para seleção do registo *probe*. Para tal será necessário movimentar o controlador para o estado *Shift-IR* de modo a escrever o comando *probe* no registo de instruções. O dado 01100 enviado para o pino TMS irá fazer com que o controlador se movimente para o estado *Shift-IR*.

Neste momento, o registo de instruções é selecionado e deverá ser enviada a instrução *Probe* para que esta seja executada. Analisando o *datasheet* do módulo STA400, é possível verificar que o código binário correspondente à instrução *probe* é 1111111111111111000 (ver Tabela 8).

Tendo a instrução *Probe* selecionada, o passo seguinte será selecionar os ABM's e o TBIC para selecionar o pino A01 ao AT2 do TBIC. Desse modo, será necessário aceder ao estado *Shift-DR* para que possa adquirir essa informação. Uma vez que no final do envio do último dado para o registo de instruções é enviado para o TMS o bit 1, o estado do controlador será o *Exit1-IR*, desse modo, para aceder ao estado *Shift-DR* será necessário enviar para o TMS o valor binário 1100. Após o envio desses dados, o sistema encontra-se apto para receber a configuração dos ABMS e do TBIC.

Analisando a Figura 33, é possível observar como se encontra organizado o registo *boundary scan* do STA400. Visto que a infraestrutura IEEE1149.4 define quais são os dados a serem preenchidos para seleção do TBIC e dos ABM's, será necessário analisar essa informação para que se possa preencher o registo *boundary scan* de forma a seleciona o pino A01 ligado ao AT2. Anteriormente, foi analisado que para colocar sinal de AT2 ligado a um ABM, seria necessário liga-lo a AB1 ou a AB2. No caso aqui apresentado, foi considerado que AT2 seria ligado a AB2. Analisando a Tabela 4 é possível verificar que existem diversas configurações possíveis para ligar AT2 a AB2, no entanto, não querendo alterar nenhum estado de AT1, nem realizar a interligação de AT2 a AT1, foi selecionado o caminho P1, que seleciona o interruptor 6 e 9 (ver Figura 54), ficando AB1 ligado à tensão interna VClamp e AB1 em alta impedância.

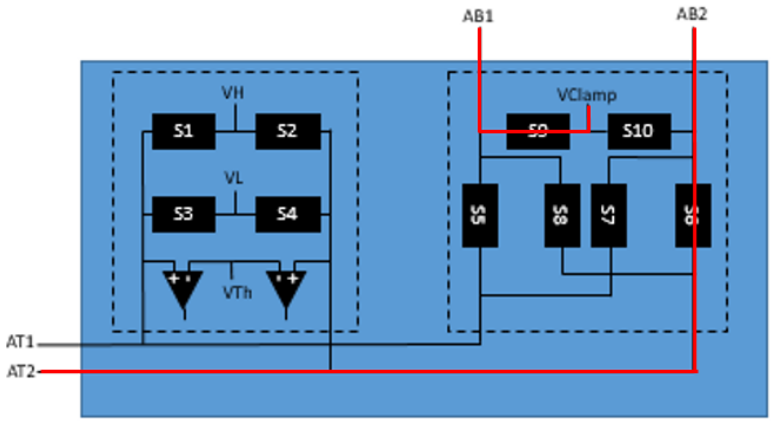


Figura 54 Configuração do TBIC AT2-AB2

De acordo com a Tabela 5 a configuração a escrever no *boundary scan* que se traduz à ativação do caminho P1 do TBIC é 0001 (para Ca, C0, D1, e D2 respetivamente). Assim sendo no registo *boundary scan*, como este se encontra ao contrário (ou seja D2, D1, C0 e Ca) o valor a ser preenchido será 1000.

Após se ter definido que o sistema iria ligar o AT2 ao AB2, será necessário ligar o AB2 a A01 para que seja possível adquirir o sinal de A01 em AT2. Para tal, foi consultada a Tabela 2, onde se definiu que os interruptores a selecionar são o SD e SB2 de forma a ligar o pino analógico ao barramento AB2, deixando o sistema no seu funcionamento normal. O caminho correspondente à ativação destes interruptores é o P17, que irá configurar os interruptores tal como ilustrado na Figura 55.

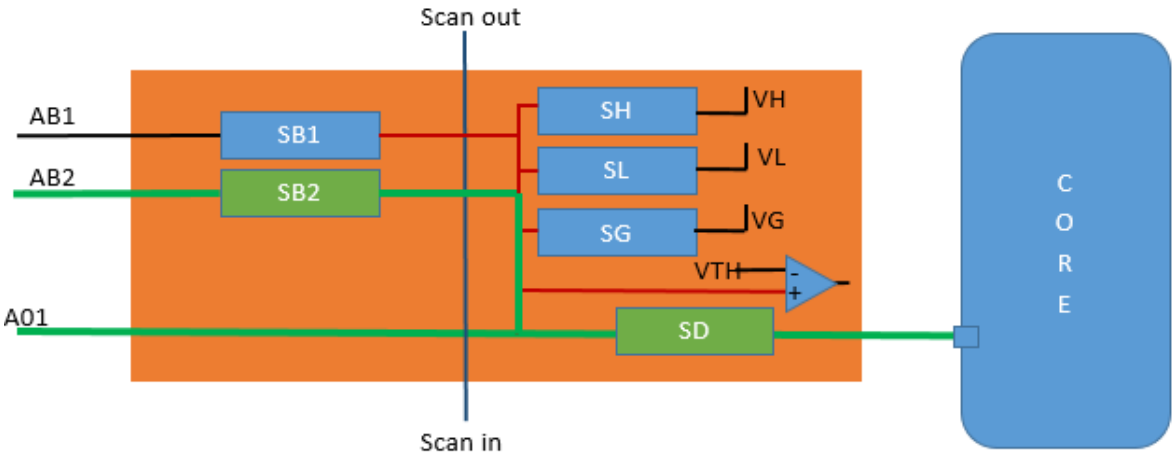


Figura 55 Configuração do ABM A01- AB2

De acordo com a Tabela 3 a configuração a escrever no *boundary scan* que se traduz à ativação do caminho P17 do ABM é 0001 (para C, D, B1, e B2 respetivamente). Visto que o registo *boundary scan*, se encontra organizado de forma diferente (ou seja D, C, B2 e B1), será necessário enviar o valor 0010 para o registo. É necessário ter em atenção que para o mesmo valor binário existem duas opções P1 e P17, no entanto como se pretende utilizar o output do sistema, estando este ligado ao sistema *core* (de forma a não alterar o funcionamento normal do dispositivo), torna-se necessário utilizar a instrução *Probe* em vez da instrução *Extest* (daí a necessidade de se ter escolhido a instrução *probe* anteriormente).

Neste momento é possível identificar claramente o valor binário a ser enviado para o registo para configuração dos interruptores, na Figura 56 encontra-se ilustrada a configuração enviada.

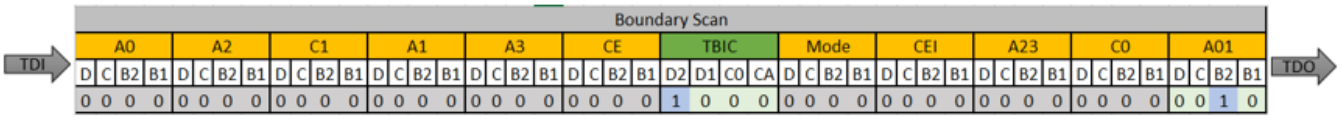


Figura 56 Configuração do registo *boundary scan*

Tal como se pode verificar na figura anterior, o sinal 0 (LSB) de A01 deverá ser escrito em primeiro lugar. Uma vez que no *software*, este tratamento já se encontra a ser realizado, o valor escrito no software deverá ser igual ao descrito na figura.

Após o envio do último bit (MSB de A0), o controlador ficará no estado *Exit-DR*. Para que o sistema entre em funcionamento normal, o controlador deverá ser colocado no estado *Run-Test/Idle*, assim sendo é necessário enviar o valor binário 10 para o TMS. Na configuração realizada, após o valor binário 10 encontram-se mais zeros à direita, fazendo com que o *software* não termine, mantendo-se no estado Run-Test/Idle.

Finalizado o teste de controlo e observação de um módulo STA400, passou-se ao desenvolvimento de um teste de controlo e aquisição, utilizando dois módulos STA400, para validar o conceito de controlo e observabilidade através de comunicação série. Na Figura 57 encontra-se ilustrado o esquema de ligação dos dois módulos STA400 ao módulo da NI.

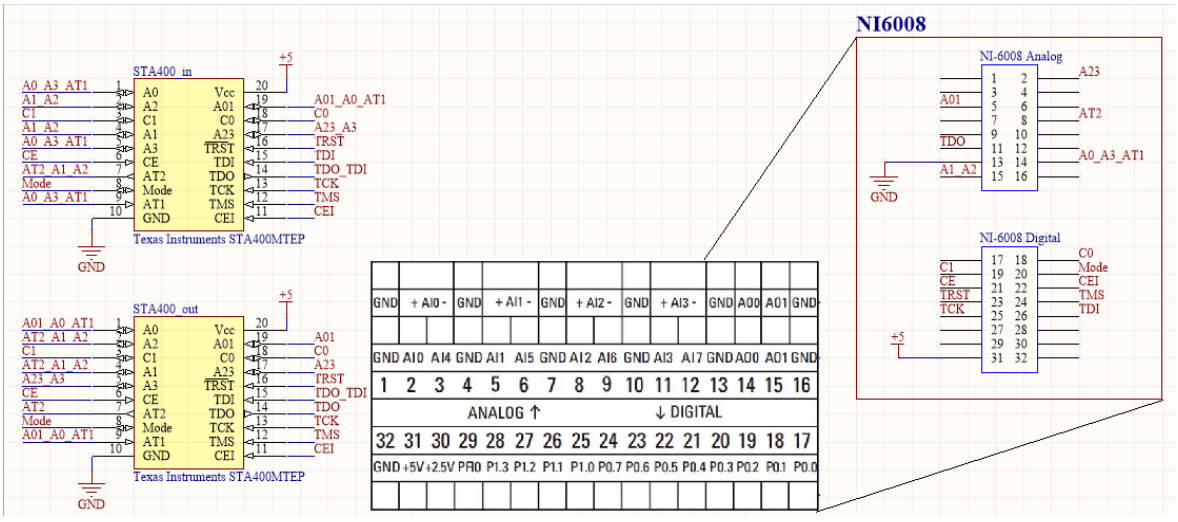


Figura 57 Esquema de dois módulos STA400

Comparando o esquema da Figura 45 com o da Figura 57 é possível verificar que os sinais ligados ao módulo da NI são iguais. As saídas do módulo *STA400 in* irá ligar às saídas do módulo *STA400 out*, e o caminho série será realizado entre os dois sistemas ligando o TDO do *STA400 in* ao TDI do *STA400 out*. Sendo os sinais de interface ao módulo da NI todos iguais, o software a utilizar será o mesmo, sendo a grande diferença deste sistema os comandos a enviar para os controladores do STA400.

Tal como no sistema anterior, foram realizados vários testes que permitiram perceber as necessidades para que os dois módulos sejam interligados e que comuniquem entre sí para realizar o teste final. Desta forma, foram criados os seguintes procedimentos

Tabela 11 Procedimentos criados para interface entre STA400's

Procedimento	Explicação
1_Bypass_2_Bypass	O módulo 1 é colocado em *bypass*, assim como o módulo 2. Sendo possível aplicar um sinal em A0 do módulo *STA400_in* e dependendo do estado do *mux*, colocar o valor em A01, sendo este sinal transmitido para A0 e para AT1 do módulo *STA400_out*. Visto que os módulos têm o mesmo controlo do *mux*, para que o sinal aplicado em A0 de *STA400_in* apareça no A01 de *STA400_out*, o *mux* deverá estar configurado de tal forma a ligar A0 a A01. (Exemplo: CE=0;0 CEI=1; Mode=0; C1=0; C0=0, ver Tabela 7).
1_Bypass_2_Extest_AT1_AT2	O módulo 1 ficará em *Bypass* e o módulo 2 irá executar a instrução *Extest*, permitindo ligar o sinal de AT1 em AT2. Para além desta configuração, também será realizado um teste de validação onde será escrita a palavra 111110000011111000011111 na

	saída de TDO no final do teste. Para tal, a palavra deverá ser escrita antes de escrever a configuração final dos módulos.
1_Extest_5V_AT1_2_Extest_AT1_AT2	Este teste será igual ao teste anterior (*1_Bypass_2_Extest_AT1_AT2*) no entanto, em vez de colocar o módulo *STA400_in* no estado *Bypass*, forçará o pino de A01 a estar a 5V, obtendo no módulo *STA400_out* o sinal de 5V em AT2.
1_Sample_2_Sample	O teste realizado verifica o estado dos pinos dos módulos obtendo a informação desse estado em TDO. A mensagem 111110000011111000011111 será escrita no final do envio do estado dos módulos.
1_Extest_5V_AT1_2_Highz	Aplicando a configuração de *Extest* no módulo *STA400_in* forçando a saída de A01 a estar a 5V, tal como no teste anterior (*1_Extest_5V_AT1_2_Extest_AT1_AT2*), no entanto coloca-se o módulo *STA_out* no estado *Highz*, forçando que este fique com os pinos do módulo em alta impedância. Para validação do teste, é necessário medir (utilizando um multímetro) o sinal à entrada do pino 19 do módulo *STA400_in* e validar que este está a 5V.

Estando os procedimentos de teste gerados, passou-se à validação de cada um. De seguida será dada a explicação do procedimento de teste "*1_Extest_5V_AT1_2_Extest_AT1_AT2*" visto que será realizado um controlo específico para cada um dos módulos. Obtendo esta informação, facilmente se percebe como criar os restantes procedimentos (os procedimentos já se encontram criados no software desenvolvido e que se seguem em anexo no DVD entregue). Na tabela seguinte encontra-se a configuração realizada do procedimento.

Tabela 12 Explicação do procedimento *1_Extest_5V_AT1_2_Extest_AT1_AT2*

Sinal	Dado
TMS	11111
TMS	01100
TDI	00000000000000000000111111111111111101000
TMS	1100
TDI	00000000000000000000000000000000000000001100 00000000000000000000001001000000000000000000 11111000001111100000111111
TMS	10

Não conhecendo o estado atual dos módulos torna-se imprescindivel, colocar os controlador num estado conhecido para que seja possível realizar o controlo dos mesmos de forma correta. Para tal, é enviado o sinal 11111 para TMS, colocando o controlador no estado *Test-logic-Reset*.

Estando os dois módulos no estado inicial, será necessário enviar o comando a executar, para tal os módulos deverão encontrar-se no estado *Shift-IR*. Dessa forma, será enviada a informação 01100 através do sinal TMS de forma a colocar o controlador no estado de configuração de comandos.

Sendo a ideia colocar os dois módulos a executar o comando *Extest*, deverá ser enviada a informação através do TDI para o controlador. Analisando a Tabela 8, é possível verificar que a instrução *Extest* pode ser enviada de duas formas, ou tudo a 0's ou 1111111111111101000, para validar que as duas instruções funcionam sem problemas, foi enviado para o módulo 2 a mensagem 1111111111111101000 e para o módulo *STA400_in* a palavra 00000000000000000000. Sabendo que o módulo *STA400_in* tem ligado o TDO no TDI de *STA400_out*, a informação a ser enviada primeira deverá ser a do *STA400_out*.

Uma vez que em *software*, o envio da mensagem já se encontra a ser tratada, a mensagem final a ser enviada para TDI deverá ser 00000000000000000000011111111111111101000.

Estado as instruções definidas para cada um dos módulos, torna-se necessário escrever a configuração do registo *boundary scan*, de modo a configurar os interruptores de ambos os módulos de modo a ter 5V na saída A01 do módulo *STA400_in* e esse mesmo sinal que irá estar disponível em AT1 do módulo *STA400_out* ser redirecionado através do pino AT2 desse mesmo módulo. Para escrever a configuração do registo *boundary scan* é necessário que o estado do controlador seja *Shift-DR*, para tal será necessário enviar o código binário 1100 para o pino TMS.

Após o controlador se encontrar no estado pretendido, passa-se à configuração do registo *boundary scan*. Para realizar a configuração de cada módulo foi necessário ter em conta as configurações apresentadas na Tabela 2. Visto que o objetivo final é obter uma tensão de 5V no pino A01 do módulo STA400, foi necessário analisar uma configuração capaz de ligar o pino analógico a uma tensão interna de 5V. Uma vez que VH é uma tensão DC interna do módulo que correspondente ao estado lógico 1 (5V), será possível utilizar este sinal para o ligar no pino A01. Observando a Tabela 2, a configuração que configura o pino lógico de saída analógica para se ligar a VH, é através do caminho P12, que irá ativar o *switch* SH. A configuração lógica correspondente ao caminho P12 utilizando a instrução *Extest* pode ser observada através da Tabela 3 e corresponde ao código binário 1100 (C, D, B1, B2).

Após a definição da configuração do módulo de entrada, será necessário configurar o módulo de saída. Sendo a configuração pretendida, ligar o AT1 a AT2, deverá ser analisada a Tabela 4 para avaliar que interruptores deverão ser ligados para realizar esta configuração. Existem dois caminhos possíveis para ligar apenas o AT1 a AT2, ou através do caminho P8 ou através do caminho P9. Sendo indiferente a escolha dos caminhos, definiu-se que se iria utilizar o caminho P8, analisando a Tabela 5, verifica-se que para definir o caminho P8, deverá ser configurado através do código binário 1001 (Ca, C0, D1, D2).

Uma vez que neste teste, para além da definição da configuração, também seria expectável obter os dados 111110000011111000001111 1 no caminho TDO após o término das configurações dos módulos, foi necessário acrescentar essa palavra no envio da trama para o pino TDI. Dessa forma a configuração final da mensagem a enviar será a seguinte.

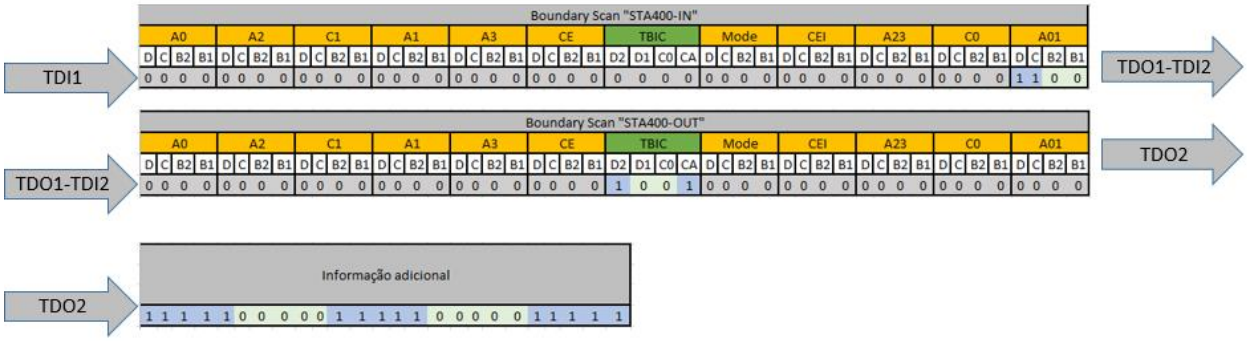


Figura 58 Configuração da mensagem

Os primeiros dados a enviar deverá ser a *informação adicional*, sendo toda a informação transmitida pelo sinal TDI, configurando o dispositivo *STA400-IN* a fornecer 5V no A01, e ligado o AT1 ao AT2 do *STA400-OUT*, obtendo em TDO a informação adicional. É de notar que a informação é transmitida apenas para o *STA400-in*, sendo a menssagem transmitida em série através da ligação existente entre o TDO de *STA400-IN* e TDI de *STA400-OUT*.

Finalizada a configuração, o estado do controlador é *Exit1-DR*, pelo que deverá ser enviado o código binário 10 para TMS, para que este fique no estado *Run-Test/Idle*. Neste momento o sinal analisado em AT2 do módulo *STA400-Out* será de 5V, sinal este que é transmitido pelo módulo *STA400-In*.

5.2. CONFIGURAÇÃO DE FPAA UTILIZANDO O MÉTODO POR OMISSÃO

A validação de funcionamento da FPAA teve como base dois objetivos, tentar configurar o sistema utilizando o *software* da Anadigm através da PIC existente na placa e o segundo objetivo é realizar a configuração da FPAA utilizando a comunicação SPI. Sendo um dos objetivos do trabalho detalhar o projeto para realização de programação da FPAA, nesse subcapítulo (5.3) será explicado de forma clara, como programar uma FPAA através do método por SPI, sendo o presente capítulo dada a explicação da implementação da configuração, utilizando o sistema base de configuração da FPAA. Tal como visto anteriormente no subcapítulo 3.6, a FPAA pode ser configurada dinamicamente ou estaticamente. Por defeito, a configuração dinâmica está ativa, podendo programar a FPAA utilizando a PIC existente na placa de desenvolvimento. Sendo esta a configuração por defeito, a posição dos *jumpers* da placa de desenvolvimento (os *jumpers* permitem realizar diferentes configurações, dependendo das necessidades do utilizador) também deverão estar

na configuração por defeito. Na figura seguinte encontra-se ilustrada a configuração da placa da Anadigm, assim como deverá estar ligada a placa de desenvolvimento para a realização dos testes.

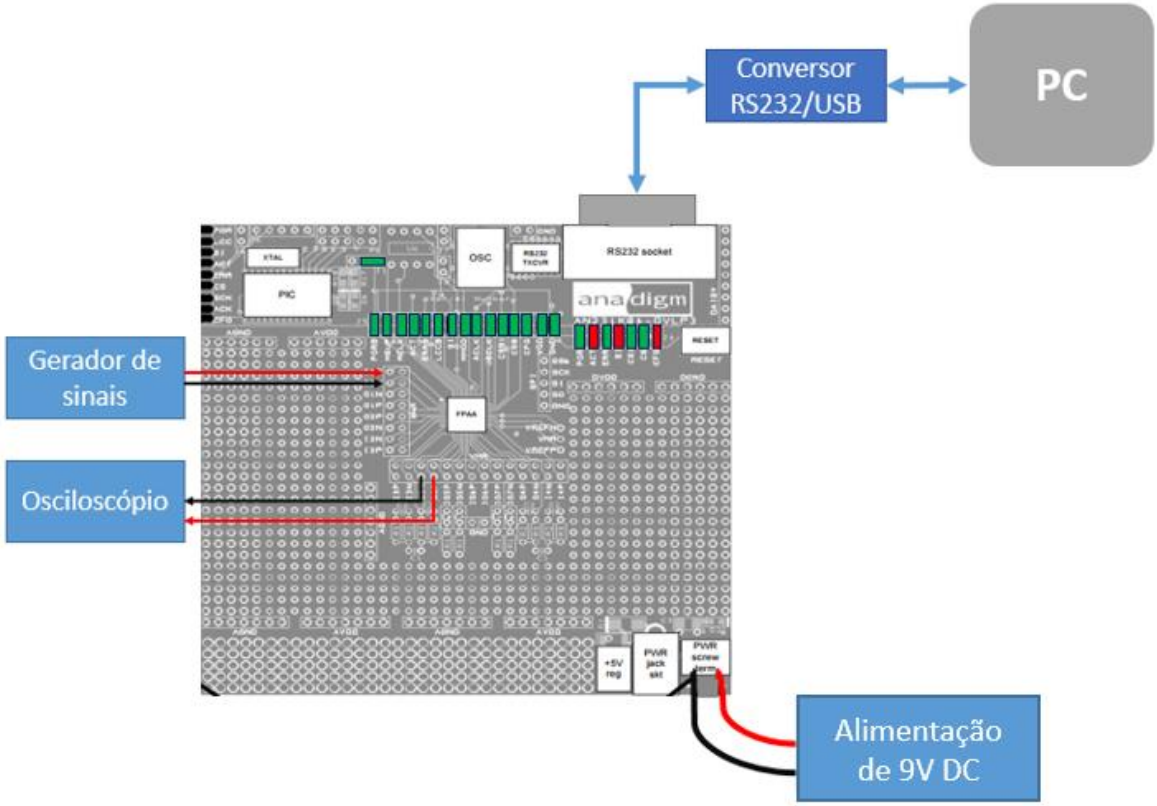


Figura 59 Ligações da FPAA através de configuração por defeito

O teste inicial, passa por injetar um sinal na entrada I1 da FPAA proveniente de um gerador de sinal, analisando o mesmo sinal na saída O4 da FPAA, analisando-o através de um osciloscópio. A placa de desenvolvimento pode ser alimentada através do *jack* de *power*, ou através de terminal de parafusos. A placa de desenvolvimento permite ser alimentada com tensões entre 4V e 12V, sendo que a recomendação dada pela *Anadigm* passa pelo fornecimento de tensão de alimentação entre 6V a 9V. Desta forma, foi utilizada uma fonte de alimentação de 9V ligado ao terminal de parafusos, estando dentro dos valores recomendados. Para realizar a configuração da FPAA, os *jumpers* deverão ser colocados tal como apresentado na figura anterior, sendo desta forma o microcontrolador PIC o responsável por programar a FPAA de acordo com a configuração enviada através do computador. Para interligar a placa de desenvolvimento com o computador, foi ligado um conversor RS232 para USB, sendo a informação enviada em série para o microcontrolador.

Após a placa de desenvolvimento estar pronta a nível de *hardware*, a alimentação da mesma poderá ser ligada, um led indicador verde na placa irá ficar ativo, indicando que a alimentação da placa está correta. Após a alimentação do sistema, a FPAA ficará apta para receber a configuração proveniente do computador. Para realizar a configuração, recorre-se ao *software* da Anadig (AnadigmDesigner 2), sendo necessário configurar a FPAA que será utilizada para realizar a programação.

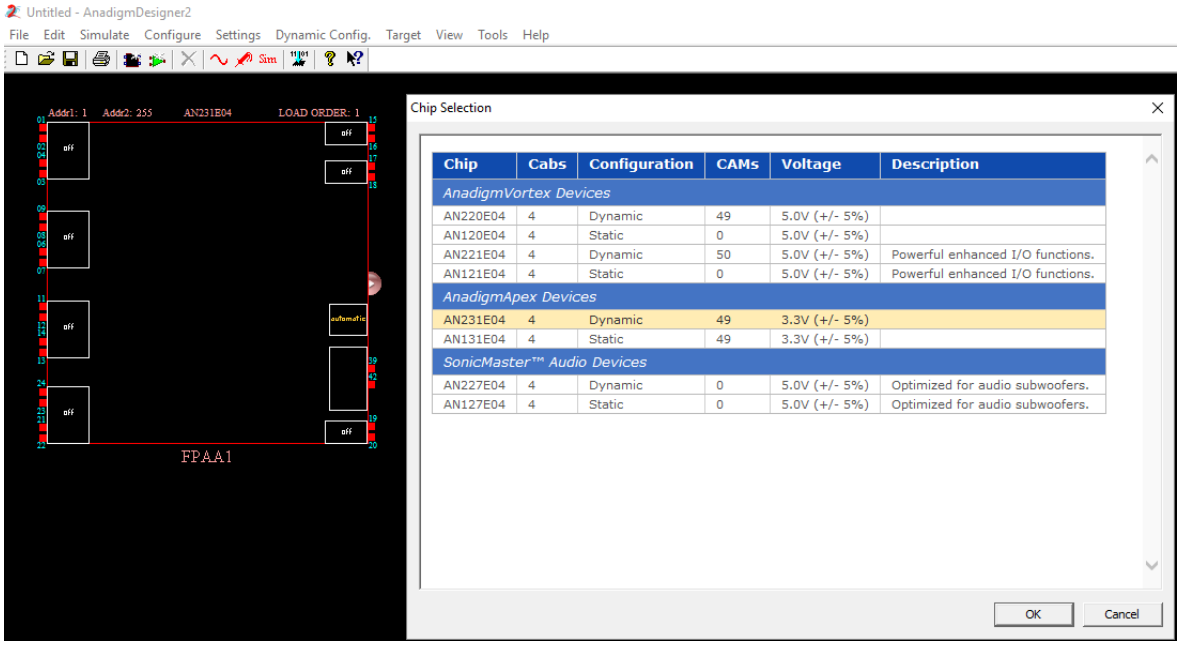


Figura 60 Configuração da FPAA no *software* AnadigmDesigner

Tal como ilustrado na figura anterior, a FPAA que deverá ser selecionada é AN231E04, que é o dispositivo que se encontra na placa de desenvolvimento. Esta FPAA possui 4 CABs, e 49 CAMs permitindo realizar a configuração dinâmica da mesma, sendo apenas possível aplicar e fornecer sinais analógicos de 3.3V. Após selecionar a FPAA, será necessário configurar a placa de forma a ligar o *input* I1 ao *output* O3. Para configurar a FPAA, é necessário selecionar a CAM específica e escolher o modo de funcionamento da CAM. Para a CAM1 é utilizado o modo de *input* e o tipo de entrada como *Bypass*, para a CAM3 é utilizado o modo de *output* e o tipo de saída como *Bypass*, na figura seguinte é possível verificar a CAM1 e a CAM3 configuradas, assim como a configuração específica da CAM1.

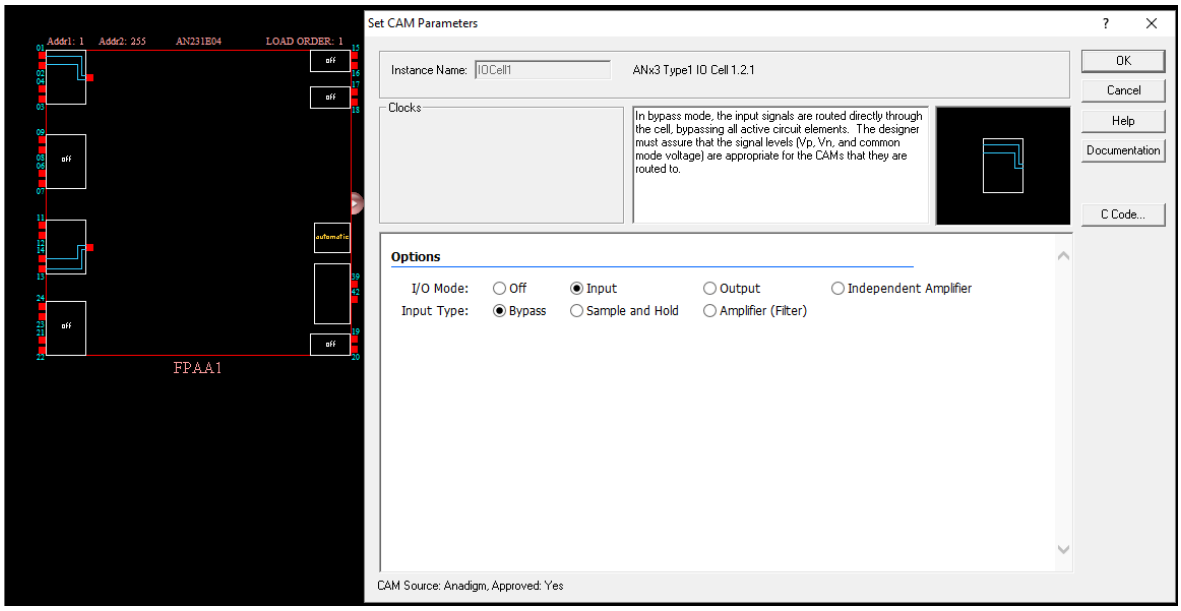


Figura 61 Configuração da CAM para o teste inicial

Configuradas as CAM's da entrada e da saída da FPAA utilizada para o teste inicial, será necessário realizar a interligação interna da FPAA para realizar o processamento do sinal pretendido. Para o processamento do sinal, poderiam ser utilizadas novas CAM's com diversas operações, tais como detetor de picos, osciladores, filtros, entre outros, no entanto para o teste inicial, pretende-se ligar a saída à entrada do sistema, desta forma bastará interligar as CAM's tal como ilustrado na figura seguinte.

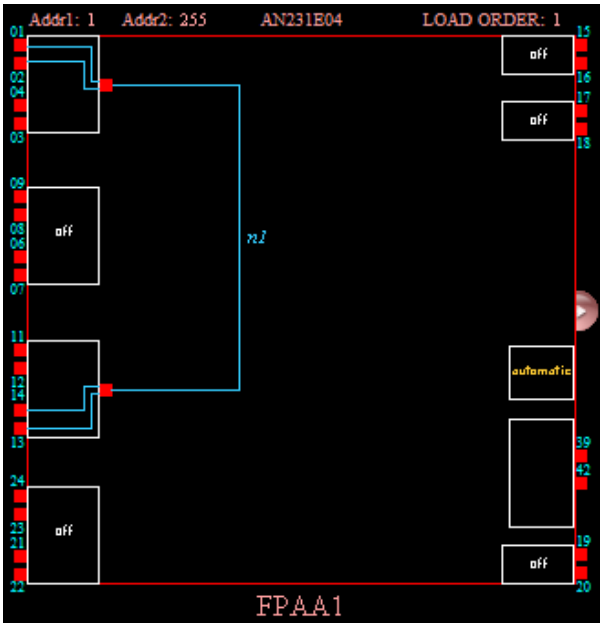


Figura 62 Configuração da FPAA para o teste inicial

Para validação do sistema, poderá ser iniciada uma simulação ao sistema configurado, sendo aplicado um gerador de sinais na entrada e configurando-o para gerar uma onda sinusoidal, podendo aplicar pontas de prova ao longo do sistema. Após correr a simulação é possível analisar os diferentes sinais gerados na entrada e o observado na saída tal como ilustrado na figura seguinte.

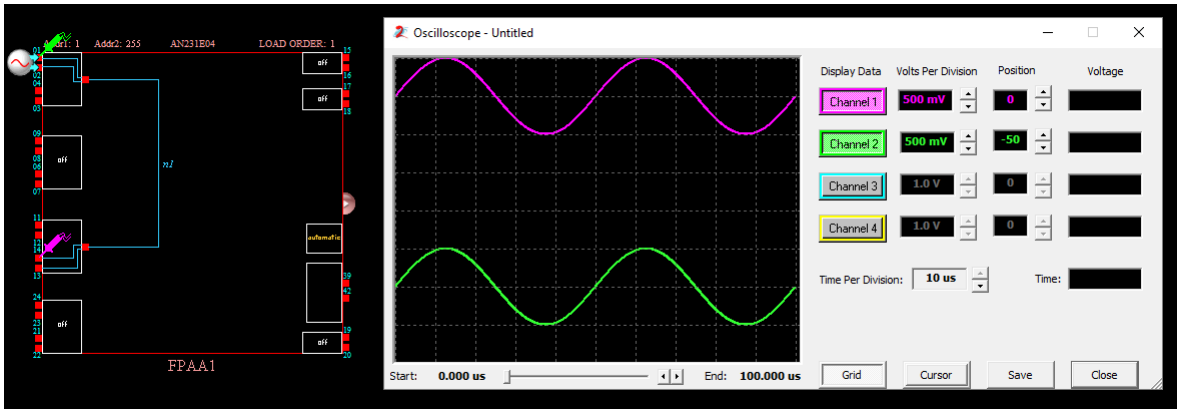


Figura 63 Simulação do teste inicial

Validado o conceito do teste através da simulação, o próximo passo a realizar é transferir a configuração para a FPAA de forma a poder validar o sistema fisicamente. Para transferir o a configuração para a FPAA deverá ser configurado através do menu *Settings->Preferences…*, a porta COM utilizada assim como o *boud rate*. Configurado a porta COM que define o conversor USB-RS232 será dado um estado da porta com a indicação se esta está disponível para iniciar a programação ou se existem problemas na configuração da mesma. Finalizada a configuração da porta COM, basta aceder ao menu *Configure* e selecionar *Write Configuration Data to Serial Port*, sendo dada a ordem de transferência de informação entre o computador e a PIC, deixando esta no estado capaz de realizar a configuração da FPAA através de SPI. Após a PIC terminar a configuração da FPAA, irá colocar o LED indicador junto da PIC com a cor verde ou com a cor vermelha em caso de sucesso ou de falha respetivamente, e o *software* também terá a capacidade de apresentar essa informação.

Estando a configuração realizada corretamente, o sistema não poderá ser desligado, visto que a FPAA possui memória volátil. Para validação do teste inicial, aplicou-se diversos sinais no gerador de sinais no pino I1P, observando a resposta da FPAA através de um osciloscópio na saída O3P, sendo o resultado obtido em O3P, o sinal injetado de I1P, estando assim validada a configuração.

5.3. DETALHE DO PROJETO DE CONFIGURAÇÃO DA FPAA

Sendo o sistema final idealizado, para utilizar um sistema único de programação, de controlo e observabilidade do sistema, o *software* da Anadigm não poderia ser utilizado para realizar a programação do mesmo. Para além deste objetivo, também se torna relevante, ter a capacidade de realizar a programação da FPAA sem a obrigatoriedade de utilizar um microcontrolador PIC no sistema. Sendo a comunicação entre o microcontrolador PIC e o computador impercetível para o desenvolvedor, torna-se importante perceber outra alternativa para realizar a programação da FPAA através de uma outra interface. Para tal será necessário ter o controlo total do sistema de configuração da FPAA, deixando de parte a interface entre a PIC e a FPAA para realizar a programação.

Tal como analisado anteriormente, a realização da configuração da FPAA é feita através de comunicação SPI para tal, será utilizado um microcontrolador que possua interface por SPI. O microcontrolador a utilizar poderia ser uma PIC, um ATMEGA, entre outros. Sendo este trabalho não vocacionado para o tipo de implementação de comunicação por SPI, foi utilizado um *Arduino* de forma a tornar este trabalho mais simplificado a nível de *hardware*. O *Arduino* tem como base um atmega 2560, dando a possibilidade de implementação da comunicação SPI e de comunicação universal asynchronous receiver/transmitter (UART) para comunicar com o computador.

A configuração por defeito da placa de desenvolvimento da FPAA está direcionada para que a programação da FPAA seja transmitida através da PIC, dessa forma torna-se relevante alterar a configuração da placa de desenvolvimento de forma a ter a possibilidade de realizar a configuração da FPAA utilizando um dispositivo externo com comunicação SPI, isolando o microcontrolador do KIT de desenvolvimento. Analisando o esquema elétrico [41] da placa de desenvolvimento, verifica-se a funcionalidade de cada *jumper*. De forma a facilitar a perceção da troca de sinais necessárias para iniciar a programação da FPAA, foi analisada a transmissão de dados [42] e o estado dos pinos da FPAA, tornando o desenvolvimento mais percetível.

Na tabela seguinte encontra-se uma descrição de cada *jumper* de acordo com a analise realizada através do esquema elétrico e das mensagens transmitidas dos dispositivos.

Tabela 13 Descrição de *Jumpers* da placa de desenvolvimento

Jumper	Descrição
J5-PorB (44-Resetb)	O *jumper* liga a FPAA ao clear da PIC. Esta linha é utilizada como *reset* e é ativa no estado lógico falso. Visto que deverá manter-se no estado *on*, deverá ser utilizado um *pull-up* na placa de desenvolvimento que será analisado mais a frente. Esta linha tem interface direta ao botão de *reset* da placa de desenvolvimento. Estando o *jumper* montado, clicando no botão de *reset* os dois dispositivos (FPAA e PIC entram no estado de *reset*)
J5-MemSetup_SO (43-SO)	O *jumper* liga a saída de dados série da FPAA ao módulo da PROM. Quando o pino *MODE* está no nível lógico *true*, a FPAA quando ligada envia o comando de leitura através desta linha para a PROM
J5-MEMCLK (42)	Este *jumper* tem como função principal ligar a linha da FPAA que irá gerar o *clock* dos dados série.
J5-ACTIVATE (41)	O pino *ACTIVATE* da *FPAA* é um *open-drain* e é colocado a *GND* quando a configuração primária está completa. O *jumper* utilizado serve para informar a PIC que a operação de configuração está completa.

J5-ERR_B (40)	O pino *ERRB_B* serve para indicar que a configuração está errada, dando a possibilidade à PIC ter essa informação utilizando o *jumper*
J5-LCC_B (39)	Este pino é colocado no estado lógico *true* durante o arranque da FPAA, mantendo-se nesse estado até a configuração local estar completa.
J5-SI (38)	A utilização deste *jumper* terá como função permitir que o microcontrolador PIC envie os dados de configuração para a FPAA
J5-MEMMODE (35-MODE)	O estado deste pino é lido pela FPAA durante a sequência de arranque. Se o pino estiver no estado lógico *false*, a FPAA ficará à espera que um dispositivo inicie a configuração, se o pino estiver no estado lógico *false*, esta irá enviar o pedido de leitura através do SO para a PROM para que envie a configuração através do canal SI.
J5-ACLK (34)	É o *clock* principal para o funcionamento da FPAA. Será a esta frequência que os condensadores comutados irão funcionar. O *jumper* servirá para ligar o *clock* gerado ao pino da FPAA
J5-SCLK (33)	Este *jumper* servirá para fazer a interface com a PROM no caso de o pino *MODE* se encontrar no estado lógico *false*, sendo a FPAA a gerar o *clock*.

J5-CS_B1 (32)	Este pino serve para selecionar a FPAA quando se pretende escrever uma configuração nova para a mesma através de comunicação SPI.
J5-CS_B (31)	Idêntico ao pino anterior (*J5-CS_B1*)
J5-CFGFLG (30)	Este pino é utilizado para interligar múltiplas FPAA's. o que neste trabalho não irá ser realizado.
J5-+3V	Pino de 3V
J5-GND	Pino de *GND*
J4-POR	Permite ligar um *Pull-up* de *POR*
J4-ACT	Permite ligar um *Pull-up* de *ACTIVATE*
J4-ERR	Permite ligar um *Pull-up* de *ERR*
J4-SI	Permite ligar um *Pull-down* de *SI*
J4-CS1	Permite ligar um o *Chip select* ao *GND*
J4-CB	Permite ligar um *Pull-down de Chip select 2*
J4-CF0	Permite ligar um *Pull-up* de *config flag*
J1	Permite desabilitar o microcontrolador PIC
J8	Liga o pino *MODE* a *GND* ou a 3.3V

Após a análise de cada uma das funcionalidades dos *jumpers*, foi possível chegar à conclusão que seria necessário desligar alguns dos pinos que ligavam ao microcontrolador de forma a isolar a FPAA. Desta forma a configuração final da placa de desenvolvimento encontra-se ilustrada na Figura 64. Note que os *jumpers* com cor verde encontram-se ligados e os de cor vermelha desligados. Para isolar o sistema será necessário ter o *clock* analógico (*ACLK*)

ligado à FPAA, o pino *MODE* ligado ao *GND* (utilizando o *jumper* J8 e o *jumper* J5-*MEMMODE*) para que a FPAA fique à espera de um sistema externo para a configurar, o pino *VDD* e *GND* ligado à FPAA para alimenta-la com 3.3V. Para além destes pinos, deixou-se a configuração dos *pull-ups* por defeito, no entanto nem todos seriam necessário estarem ligados ao sistema. O *PORb* é obrigatório, para que a FPAA não entre em *reset* (apenas se carregar no botão de *reset*), o *jumper* J4 CB também se deixou ligado por defeito, visto que se pretende uma ligação ponto a ponto com o sistema de programação (Arduino), no caso de se incluir mais do que uma FPAA, o pino *CS_B* de cada FPAA (pino da FPAA que liga ao *jumper* J4 CB) deverá ser ligado ao microcontrolador individualmente. O *jumper* J1 deixou-se ligado, apenas para que se possa verificar que estando a PIC ligada, esta não influencia na programação da FPAA.

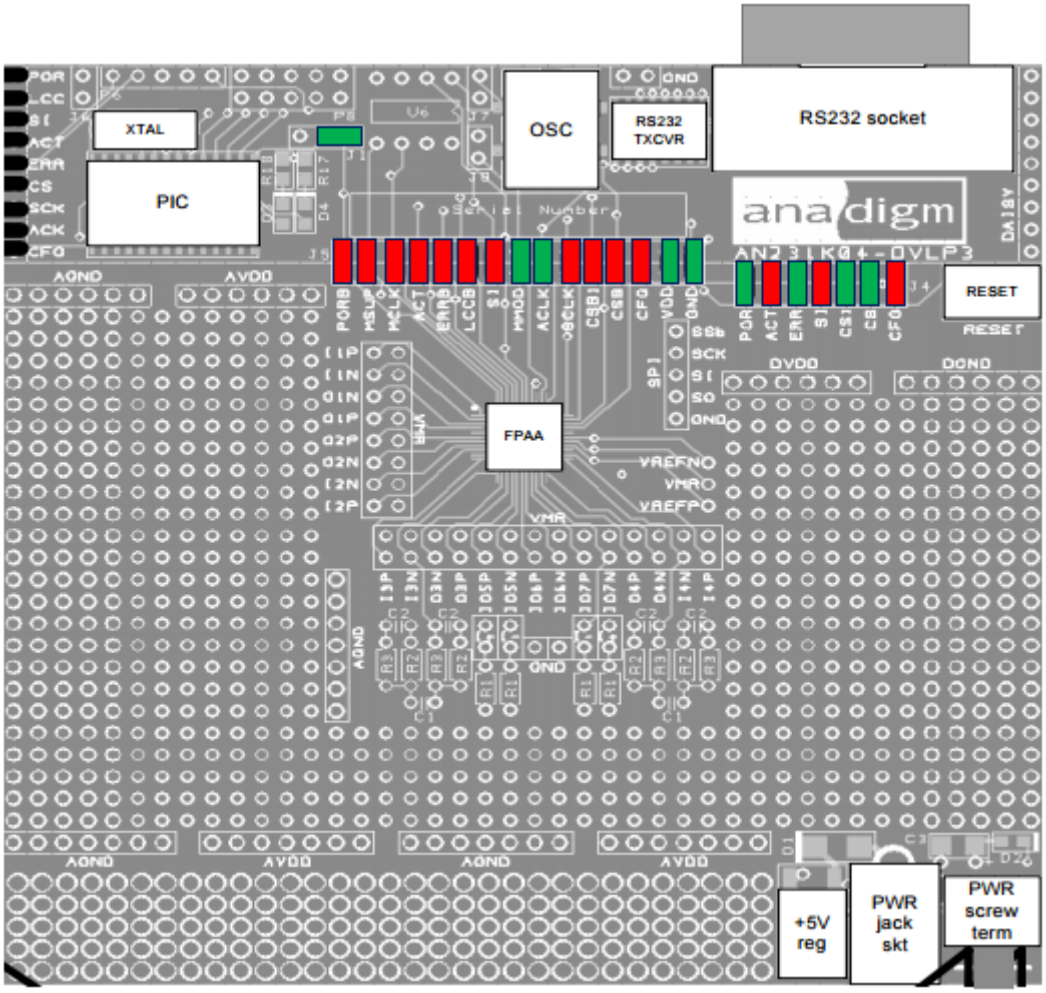


Figura 64 Configuração dos *jumpers* na placa de desenvolvimento

Visto que a FPAA trabalha com níveis de tensão de 3.3V, e o microcontrolador utilizado (Arduino) trabalha com níveis de tensão de 5V, é necessário aplicar um divisor de tensão

para passar os sinais de 5V do SPI para os 3.3V. Os pinos do Arduino mega que têm a função de comunicação de SPI são D50 (MISO), D51 (MOSI), D52 (SCK). Sendo o Arduino o dispositivo *master* e a FPAA a *slave*, será necessário ligar apenas o MOSI (*Master Out-Slave In*) do Arduino ao SI da FPAA, o MISO (Master In- Slave Out) não será utilizado uma vez que a FPAA encontra-se configurada como operação dinâmica (através do pino *MODE*), não enviando nenhuma informação para o Arduino. Sendo o Arduino ligado diretamente à FPAA e sendo este o único dispositivo de comunicação, o *Chip Select* da FPAA será ligada sempre ao *GND* (tal como se verificou anteriormente, ligou-se o *jumper* J4-CB para o efeito), dando a possibilidade de ser sempre iniciar uma comunicação através do Arduino. O *Chip Select* na comunicação SPI, permite selecionar o dispositivo que irá ficar à escuta para troca de mensagens. Tal como pode ser analisado na Figura 65 para que o dispositivo *Master* possa comunicar com qualquer um dos *Slaves*, será necessário utilizar 3 pinos (SCLK, MOSI e MISO) em comum, e um *Chip Select* individual para cada um dos dispositivos (SS). No momento em que o *Master* inicia uma transmissão de dados com um *Slave*, obrigatoriamente necessita de colocar a linha de *Chip Select* do respetivo dispositivo *Slave* ligado ao GND.

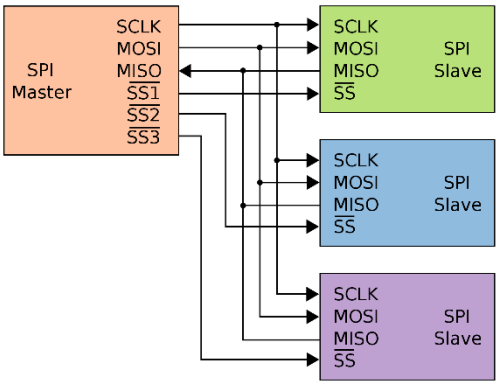


Figura 65 Configuração de comunicação SPI entre múltiplos *Slaves* [48]

No sistema atual, a comunicação SPI é realizada entre um *Master* (Arduino) e um *Slave* (FPAA), não sendo necessário ter o controlo através do *Chip Select*, deixando este pino ligado sempre à massa possibilitando sempre a comunicação entre os dois dispositivos. No caso de um sistema com múltiplas FPAA's seria necessário ligar vários pinos digitais (o mesmo número de FPAA's) a cada dispositivo, realizando uma comutação de estados entre comunicações com os diferentes dispositivos. Sempre que iniciada uma comunicação com uma FPAA específica, seria colocado o pino digital (*Chip Select*) ligado ao *GND*, e os restantes ligados a 3.3V, após a realização da programação da FPAA selecionada, o pino

digital deverá ser colocado a 3.3V, permitindo assim selecionar um novo *chip select* de uma outra FPAA. Na figura Figura 66 encontra-se ilustradas as ligações efetuadas para realizar a programação da FPAA através de SPI, para o sistema atual (*Master* comunica diretamente com *Slave*).

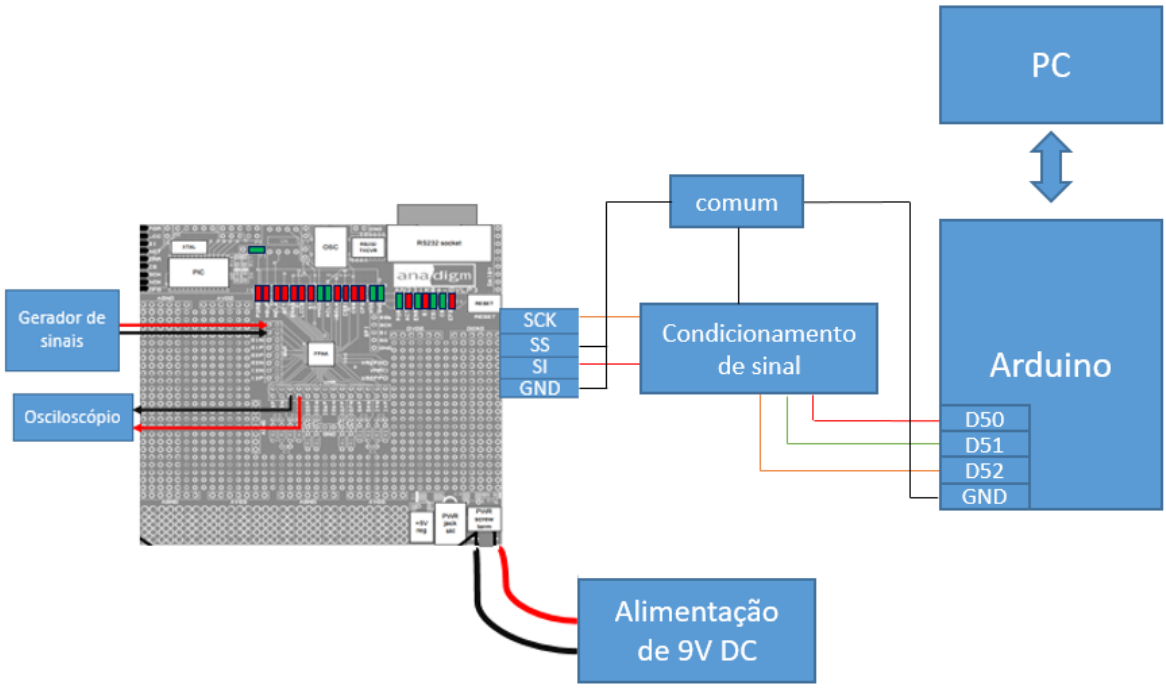


Figura 66 Interligação da FPAA com o Arduino

Tal como ilustrado na figura anterior, o computador liga-se ao Arduino através de porta USB, alimentando-o tendo acesso a uma porta UART para comunicar entre os dois dispositivos. É de notar que o pino D51 (MOSI) do Arduino está ligado ao condicionamento de sinal. O condicionamento de sinal aqui representado, é um divisor de tensão de forma a passar os sinais de 5 para 3.3V através da expressão (2). Note que *Rsup* da expressão abaixo, corresponde à resistência que liga ao pino D51 do Arduino e a SI da FPAA e que *Rinf* corresponde à resistência que liga entre a resistência *Rsup* e o comum (*GND*).

$$V_{out} = \frac{R_{inf}}{R_{inf} + R_{sup}} \times V_{in} = \frac{20k}{20k + 10k} \times 5 = 3.333V \qquad (2)$$

O sinal após passar pelo divisor de tensão está a ser ligado ao pino SI da FPAA assim como ao pino D50 (MISO) do Arduino para validar que a mensagem está a ser corretamente escrita (apenas utilizado durante a fase de testes de desenvolvimento). Pelo mesmo motivo do

anterior, o sinal do pino D52 (SCK) do Arduino está a passar pelo divisor de tensão antes de ser ligado à FPAA. De forma a tornar o sistema interligado entre sí, é estritamente necessário colocar as massas do sistema comuns.

Após a realização da interligação dos dois sistemas, passou-se ao desenvolvimento de *software* tanto para a aplicação do computador, como para o *firmware* do Arduino. Tal como apresentado anteriormente, o *software* da Anadigm permite exportar a informação criada no desenvolvimento gráfico de configuração para um ficheiro hexadecimal pronto a ser enviado via SPI para o dispositivo.

Para exportar a configuração de desenvolvimento para um ficheiro hexadecimal tendo a configuração base criada, acedendo ao menu *Dinamic Config.->State-driven Method…*será possível definir diversos parâmetros de configuração para a criação do ficheiro. Na Figura 67 encontra-se ilustrado um exemplo de página de configuração que será apresentado quando acedido ao menu anteriormente indicado.

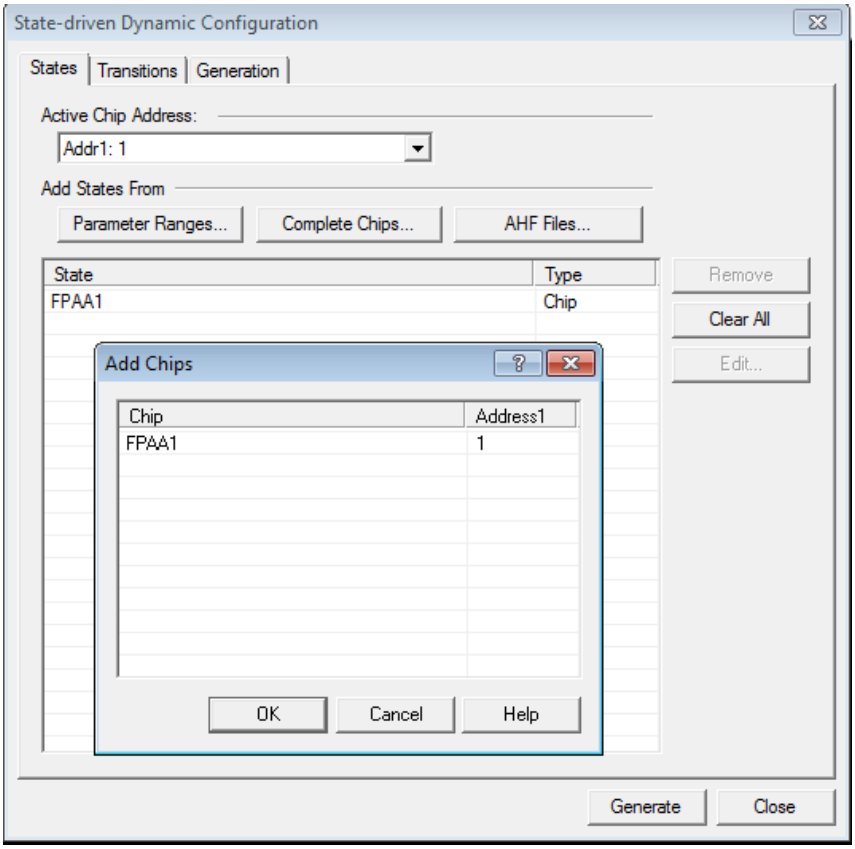


Figura 67 Definição de parâmetros para criação do ficheiro hexadecimal

Na opção *Active Chip Address:* deverá estar definido o endereço do dispositivo criado no ambiente gráfico e na tabela apresentada abaixo, é possível definir parâmetros configuráveis tais como frequências de corte, níveis de tensão, entre outros parâmetros variáveis de determinada CAM. Para a definição desses parâmetros deverá ser utilizada a opção *Parameter Ranges…* onde será listados os parâmetros configuráveis das CAM's criadas. Para gerar o código hexadecimal da configuração da FPAA, desenvolvida em ambiente gráfico, deverá ser selecionado através da opção *Complete Chips…* onde é apresentada uma lista de FPAA's utilizadas no ambiente gráfico. Na Figura 67, é possível verificar que não foram adicionados os parâmetros configuráveis, visto que a configuração da FPAA definida não utiliza CAM's, sendo apenas definida a FPAA1 como parâmetro de geração de código. Na tab *Transitions*, será apresentada uma lista idêntica à lista na tab *States*, onde se deverá selecionar o código principal. Visto que o único código a ser criado será da FPAA, fica automaticamente selecionado a FPAA1. Na tab *Generation*, será possível definir o tipo de ficheiro que será criado. Após algumas análises, verificou-se que o tipo de ficheiro ideal para o processamento, seria utilizar um ficheiro C com a configuração formatada em texto. Após a definição do tipo de ficheiro a ser gerado, inicia-se a geração do mesmo, obtendo um ficheiro idêntico ao da figura seguinte.

StateDrivenOutput_Address1_1.txt

```
/*****************************************************************************\
*                                                                             *
*                            Primary Configurations                           *
*                                                                             *
\*****************************************************************************/

/*-----------------------------------------------------------------------------*\
|                                FPAA1 (53 bytes)                             |
\*-----------------------------------------------------------------------------*/

0x00, 0x00, 0x00, 0x00, 0x00, 0xD5, 0xB7, 0x20,
0x01, 0x00, 0x01, 0xC1, 0xC4, 0x00, 0x0E, 0x20,
0x04, 0x00, 0x02, 0x00, 0x00, 0x00, 0x40, 0x00,
0x00, 0x51, 0xFF, 0x0F, 0xF1, 0x2A, 0xC2, 0x01,
0x01, 0x40, 0x2A, 0xDE, 0x01, 0x08, 0x02, 0xFF,
0x00, 0x50, 0x00, 0x00, 0x00, 0x50, 0x2A, 0x8D,
0x02, 0x07, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00,
0x40, 0x2A, 0x00


/*****************************************************************************\
*                                                                             *
*                          Transition Configurations                          *
*                                                                             *
\*****************************************************************************/
```

Figura 68 Ficheiro de configuração gerado através do *software* Anadigm

Analisando a Figura 68 verifica-se que o documento encontra-se estruturado em duas partes, a configuração primária e a configuração das transições. A configuração primária, tal como o nome indica será a configuração que deverá ser carregada quando a FPAA entra em funcionamento inicialmente. A configuração das transições são todos os parâmetros configurados anteriormente como sendo variáveis, podendo envia-los a qualquer momento para alterar em tempo real o funcionamento da FPAA de forma se ajustar ao ambiente em questão.

De forma a poder transmitir a informação exportada do *software* Anadigm para a FPAA torna-se importante enviar todos o bytes do ficheiro gerado para o Arduino, para que este tenha a capacidade de realizar a configuração através de SPI. Visto que a comunicação terá de ser feita entre o computador e o Arduino, foi necessário criar um protocolo de comunicação para que a informação seja enviada via UART. A tabela seguinte identifica o protocolo definido para implementação da comunicação entre o computador e o Arduino.

Tabela 14 Protocolo de comunicação entre o computador e o Arduino

Início	Tamanho1	Tamanho2	Tamanho…	Fim Tamanho	Dado1	Dado2	Dado3	DadoX
0x70	0xXX	0xXX	0xXX	0x95	0xXX	0xXX	0xXX	0xXX

Tal como se pode analisar pela Tabela 14, o protocolo define um identificador de início de trama, diversos *bytes* de dados que identificam o tamanho da mensagem, o identificador de fim do tamanho da mensagem, seguido dos dados que se pretende enviar através de SPI para a FPAA, de modo a que esta seja configurada. O tamanho dos dados, tal como se observa no protocolo, pode ser considerado por diversos números de *bytes*. Cada *byte*, corresponde a um número na ordem escrita, ou seja, imaginando que se pretende enviar 157 *bytes* de dados, a trama de dados a ser enviada deverá ser a seguinte.

Tabela 15 Exemplo de trama a enviar para 157 bytes de dados

Início	Tamanho 1	Tamanho 2	Tamanho 3	Fim Tamanho	Dado1	Dado2	Dado…	Dado157
0x70	0x01	0x05	0x07	0x95	0xXX	0xXX	0xXX	0xXX

Observando a Tabela 15, pode-se concluir que o tamanho de dados é definido pela seguinte equação:

$$Tamanho = Tamanho_1 * 100 + Tamanho_2 * 10 + Tamanho_3 * 1$$
$$Tamanho = 1 * 100 + 5 * 10 + 7 * 1 = 100 + 50 + 7 = 157$$

Na figura seguinte encontra-se ilustrado o fluxograma da aplicação que irá correr no Arduino de forma a implementar o protocolo definido anteriormente, de forma a configurar a FPAA através de SPI.

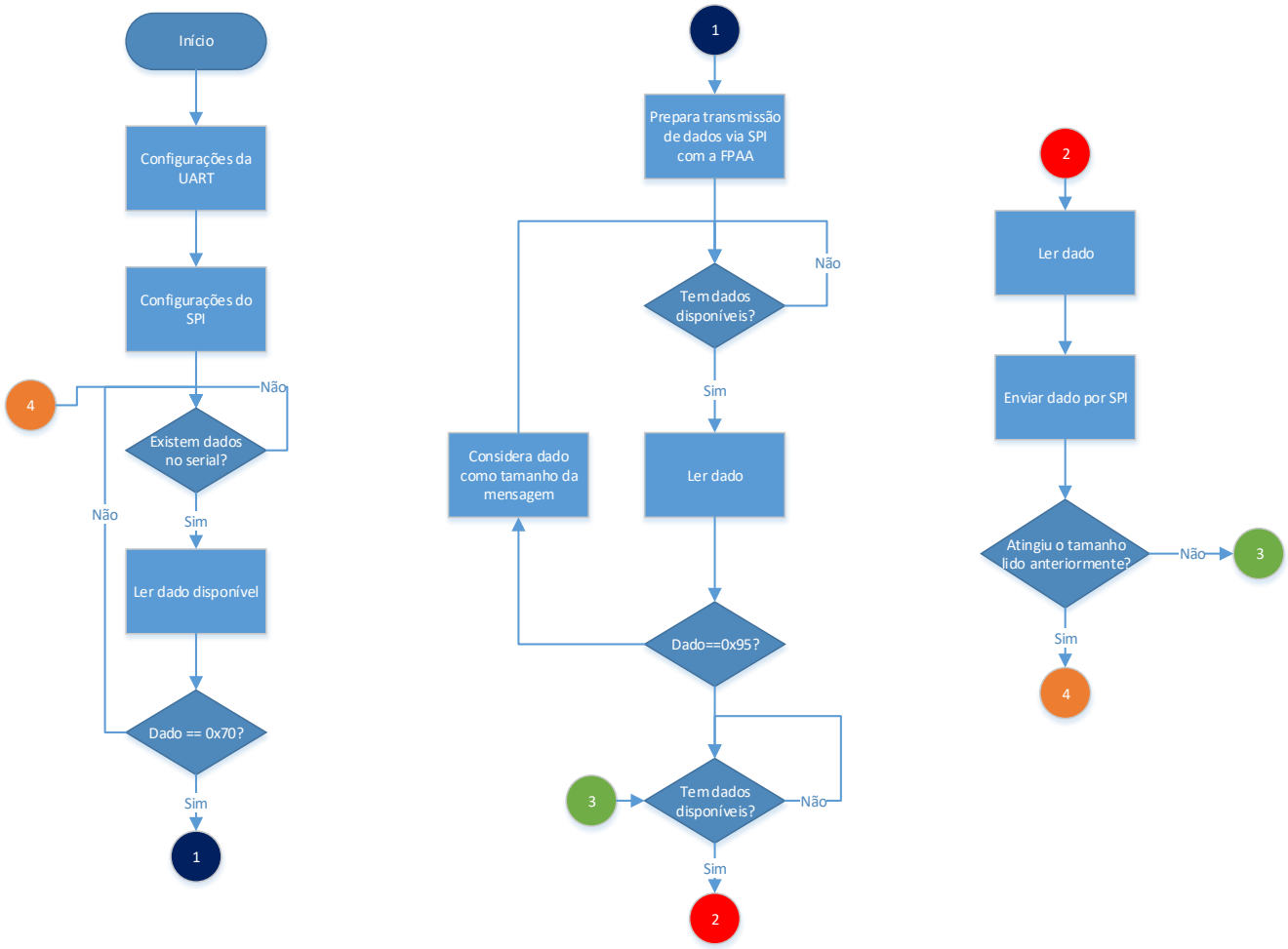


Figura 69 Fluxograma da aplicação no Arduino

A aplicação que irá ser executada no Arduino servirá apenas para realizar a interface entre o computador e a fase de programação da FPAA.

O Arduino irá iniciar a configuração da UART e do SPI, ficando à espera que o computador envie o dado 0x70 de início da trama. De seguida, irá ficar a espera de dados para cálculo do tamanho de dados, em que cada *byte* corresponde à ordem numérica do tamanho de *bytes*, sendo o terminador de tamanho definido pelo dado 0x95. Os dados seguintes, correspondem aos dados a serem enviados para a FPAA, sendo cada *byte* transmitido diretamente para a FPAA por comunicação SPI. Após o número de *bytes* atingir o tamanho de *bytes* definido anteriormente na trama, todo o fluxo será reiniciado, ficando a aguardar por um novo 0x70.

O *firmware* que deverá ser executado no Arduino irá implementar do protocolo definido anteriormente (ver Tabela 14). O código fonte será fornecido no DVD entregue juntamente com este documento, sendo este baseado no fluxograma da Figura 69. Após o desenvolvimento do código fonte, torna-se necessário programar o Arduino. Para tal, será necessário configurar a placa em utilização para a programação do dispositivo. Na figura Figura 70 encontra-se a opção para selecionar o dispositivo.

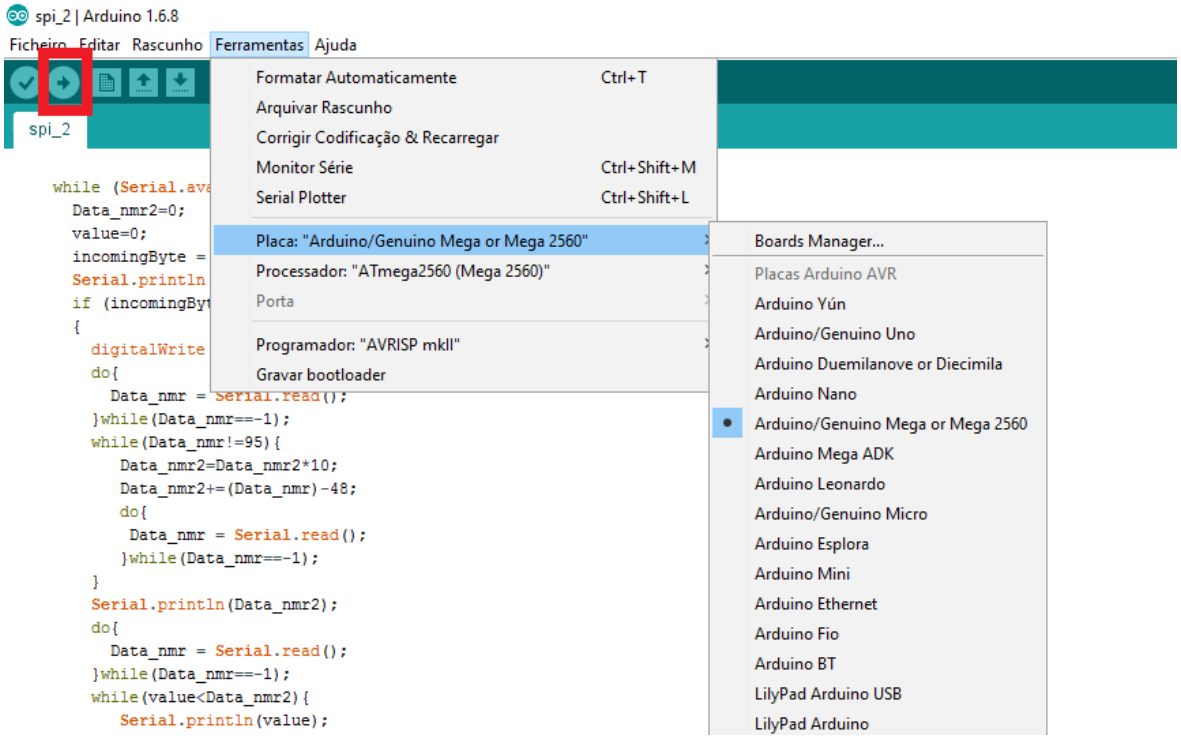


Figura 70 Selecionar a placa do Arduino

Após a seleção do dispositivo, o utilizador poderá clicar no botão de *Envio* (quadrado vermelho da Figura 70), para que o Arduino seja programado com o código desenvolvido. Note que a programação do Arduino é realizada apenas uma vez, sendo o *firmware* guardado

na memória do microcontrolador. Quando o Arduino é alimentado, a execução do código iniciará automaticamente.

Sendo este trabalho desenvolvido com base na aplicação Labview, foi desenvolvida uma função que tem como parâmetro de entrada o ficheiro gerado anteriormente, transformando-o num *array* de *bytes* descritos no ficheiro, de forma a realizar a comunicação direta com o Arduino (implementando o protocolo definido anteriormente) para que este transmita via SPI a configuração para a FPAA. Na Figura 71 encontra-se ilustrado o fluxograma da função que analisa o ficheiro gerado transformando-o num *array* de *bytes*.

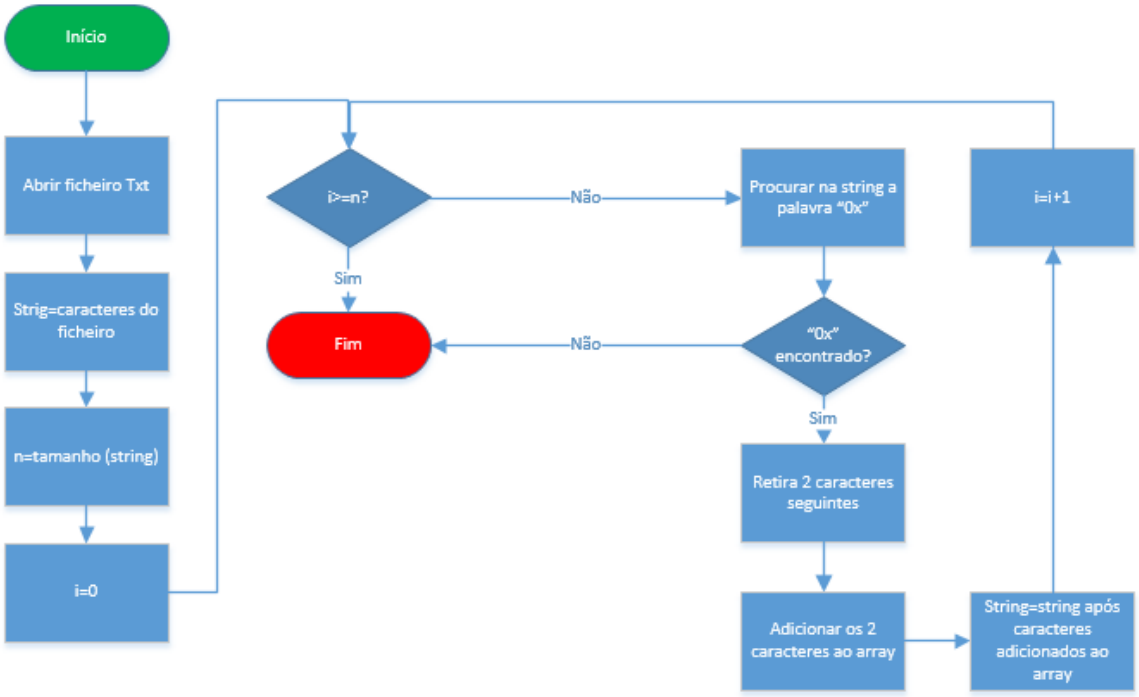


Figura 71 Fluxograma de transformação de ficheiro em array de dados

O objetivo da função anterior é percorrer o ficheiro *TXT* analisando todos os caracteres que se encontram a seguir aos caracteres "*0x*" colocando-os num *array* por ordem de escrita no ficheiro. Estando a configuração da FPAA descrita no *array* de dados, torna-se mais simples transmitir essa informação para a FPAA através do *Arduino*.

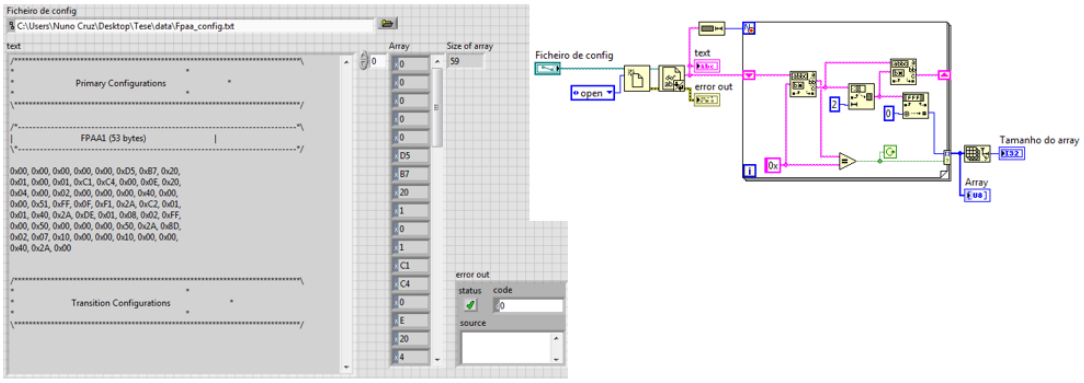


Figura 72 Função de conversão de ficheiro em *Array* de dados

Obtendo a informação da configuração a enviar para a FPAA em formato de *array*, e utilizando o protocolo de comunicação entre o Arduino e o computador (ver Tabela 14), o desenvolvimento da função de LabView torna-se de simples execução. Na Figura 73 encontra-se ilustrada a função de teste desenvolvida em Labview para enviar a configuração para a FPAA através do Arduino.

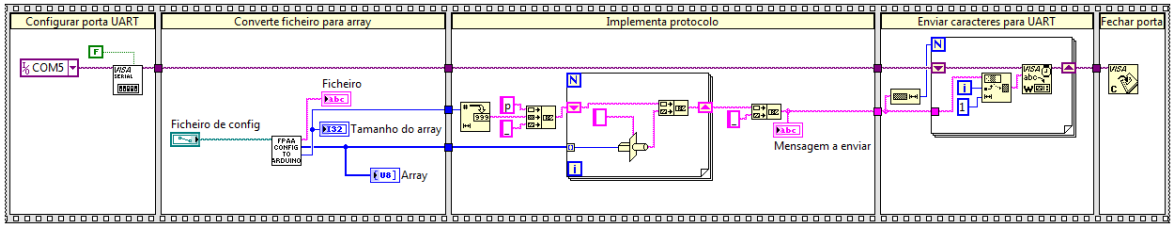


Figura 73 Envio da configuração através da comunicação com o Arduino

Analisando a figura anterior, verifica-se se são executados cinco passos. O passo inicial serve para configurar a porta UART em utilização para comunicar com o Arduino. O passo seguinte utiliza a função anteriormente explicada (Função de conversão de ficheiro em *Array* de dados), dando como *input* da função a localização do ficheiro de configuração, obtendo como output o *array* de dados, o tamanho do *array* e o conteúdo do ficheiro. Tendo o tamanho e o *array* de dados, é implementado o protocolo definido anteriormente, criando uma mensagem da seguinte forma "*p(tamanho do array)_(array de dados em hexadecimal)_*". Estando a mensagem a enviar criada, é percorrido cada caracter da mensagem, enviando um a um através da UART para o Arduino. Terminado o envio é fechada a ligação da UART.

Correndo a aplicação, o Arduino irá receber a configuração proveniente do PC, enviando-a através de SPI para a FPAA. Quando a FPAA recebe corretamente a configuração primária, coloca o pino *Activate* no estado lógico *True* [42]. Assim sendo, após realizar a configuração através de SPI, deverá ser analisado o estado do pino *Activate* de forma a validar que esta ficou corretamente configurada. Estando o sinal no estado lógico *True*, poderá ser ligado o gerador de sinais e o osciloscópio à placa de desenvolvimento, de forma a validar o funcionamento da FPAA de acordo com a configuração realizada. Após a introdução do osciloscópio e o gerador de sinais nos pinos respetivos de teste, verificou-se que o resultado obtido à saída da FPAA é igual ao injetado na entrada (estando configurada para fazer *bypass* interno).

5.4. IMPLEMENTAÇÃO DO SISTEMA FINAL

Após os vários testes desenvolvidos individualmente aos módulos, obteve-se a informação necessária para implementação do sistema final. O sistema final resulta na interligação dos sistemas acima referenciados. A interligação dos módulos teve por base a arquitetura do *hardware* apresentada na Figura 43.

Analisando os dois sistemas é possível verificar que ambos trabalham com níveis de tensão diferentes. Desta forma tornou-se relevante desenvolver um sistema de atenuação do sinal, diminuindo a amplitude do sinal de entrada para a FPAA proveniente do STA400. Inicialmente, o condicionamento de sinal foi idêntico ao já implementado anteriormente para interligar o sistema do Arduino à FPAA, ou seja um divisor de tensão, no entanto após a execução de alguns testes, verificou-se que o sinal era influenciado pela FPAA devido ao efeito de carga. Desta forma o condicionamento de sinal executado teve como base um divisor de tensão com a interligação a um seguidor de tensão (buffer). Na Figura 74 encontra-se ilustrado o condicionamento de sinal implementado.

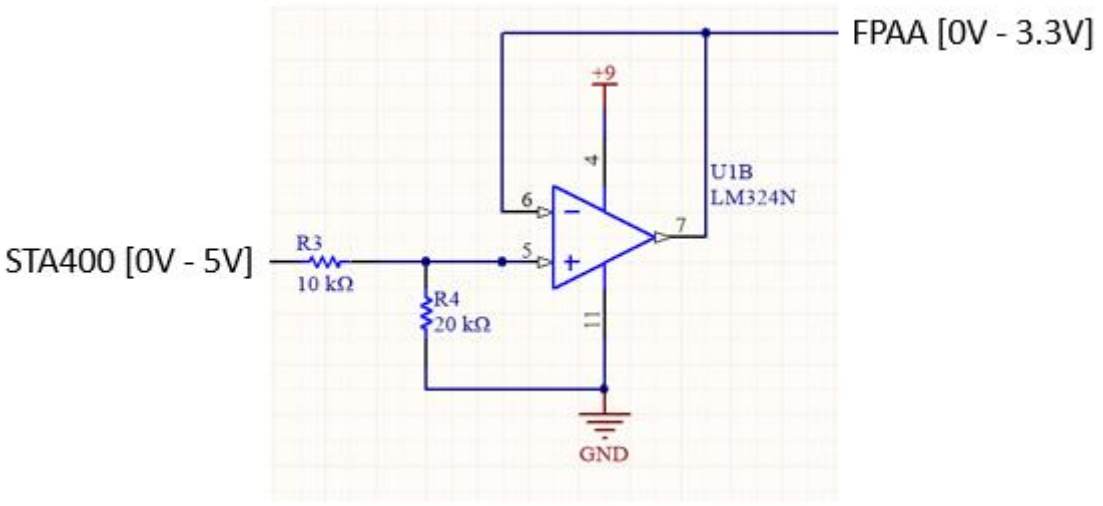


Figura 74 Condicionamento de sinal entre STA400 e FPAA

Tal como pode ser verificado no sistema anterior, o cálculo do divisor resistivo é igual ao já apresentado na expressão (2). Convertendo o sinal de 5V para 3.3V, no entanto, devido ao sistema interno da FPAA, o valor das resistências são influenciadas, tornando o valor à saída menor do que era esperado, dessa forma foi utilizado um amplificador operacional de ganho unitário. A escolha do *ampop* foi baseada nas alimentações existentes no sistema, visto que não existem tensões negativas, tornava-se necessário que o *ampop* funciona-se corretamente com valores de referência no GND. Outro aspeto relevante para a escolha deste *ampop* é que possibilita ter uma gama de tensões alargada tendo uma tensão de modo comum entre 0V e *Vsupply*-2V [43]. Visto que os valores de tensão disponíveis até ao momento para o sistema são de 5V e 9V, utilizando 5V para alimentar o *ampop*, a gama de tensão iria varia entre 0 e 3V, não atingindo os valores pretendidos. Visto que existe a possibilidade de utilizar a fonte de alimentação da FPAA de 9V, a gama de tensão do *ampop* será de 0V a 7V, obtendo uma gama de operação dentro das necessidades.

Sendo a saída da FPAA regida a valores de tensão nunca superiores s 3.3V, o STA400 funciona corretamente com estes níveis de tensão tendo em consideração que a função principal do módulo é ser *multiplexer*. Desta forma o sinal não necessita de ter amplificação, sendo a resposta o sinal directo proveniente da FPAA.

Tendo a interligação entre os dois sistemas bem definida, passou-se à implementação do esquema da placa de desenvolvimento, na figura seguinte encontra-se ilustrada a interligação entre todos os sistemas.

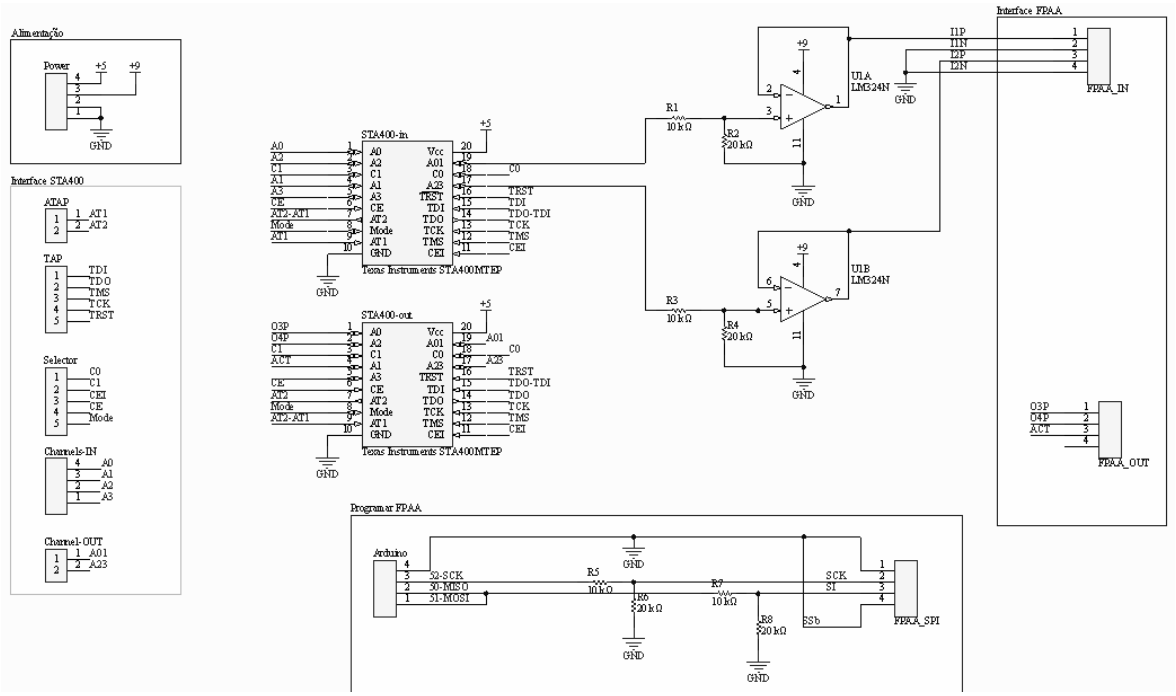


Figura 75 Esquema do sistema final

O esquema do sistema final, foi desenhado utilizando a divisão dos sinais através de conectores de forma a ser mais fácil interpretar os diferentes módulos. Analisando a Figura 75 é possível verificar que existem diferentes grupos de conectores. Um dos grupos passa pelo conector de alimentações, tendo a capacidade de ter 5V e 9V para ser distribuído na placa. Um outro grupo de interface com os módulos STA400, grupo este que irá ser interagido através do módulo USB da NI. Um grupo que realiza a interface com a FPAA, ligando diretamente os sinais aos pinos da placa de desenvolvimento da FPAA. O grupo de configuração da FPAA não é simplesmente abrangido por conectores, possuindo também o condicionamento de sinal, sendo este o responsável por realiza a interface de programação da FPAA através do módulo Arduino. E para finalizar, foi desenhado o esquema de ligações dos módulos STA400 (entrada e de saída) de forma a interligar todos os conectores dos grupos referidos implementando o sistema final pretendido.

Avaliando o grupo de interface com os STA400s, percebe-se que as ligações foram divididas por conectores diferentes de forma a tornar mais fácil a perceção do esquema. Desse modo, a divisão foi feita através de 5 conectores, o conector do ATAP, o conector do controlador TAP, o seletor do *mux*, os canais de entrada do ponto de vista da FPAA e os canais de saída do ponto de vista da FPAA. Sendo o *hardware* de aquisição e controlo uma pequena limitação devido ao número de canais digitais disponíveis, foi decidido que os sinais seletor

de *mux* iriam ser comuns para os dois STA400, desta forma, C0, C1, CE, CEI e MODE irão ficar interligados entre o módulo da NI e os dois módulos STA400. O sinal de AT1 irá ser ligado apenas ao STA400 in e o AT2 será ligado apenas ao STA400 out, pelo que o controlo será realizado através de AT1 e a observação pelo AT2,ligando estes dois sinais ao conector ATAP, que por sua vez irá ligar ao módulo da NI. Os sinais TDI e TDO estão ligados da mesma forma que os sinais AT1 e AT2, sendo TDI ligado ao STA400 de entrada e o TDO ao de saída, sendo o controlo realizado através do sinal TDI e a observação através do sinal TDO. Entre os dois módulos são interligadas as vias de observação/controlo de TDO e AT2 (do STA400-in) a TDI e AT1 (do STA400-out) respetivamente. Sendo esta uma ligação série, tal como ilustrado na Figura 16, TMS, TCK e TRST são comuns para todos os módulos. Para que seja possível controlar através do módulo da NI os sinais de entrada do *multiplexer* STA400-in (A0, A1, A2 e A3) são interligados aos canais de saída analógicos do modulo da NI, e os sinais de saída do *multiplexer* STA400-out (A01 e A23) são interligados aos canais de entrada analógicos do módulo da NI.

Sendo a ideia do sistema, ser capaz de se interligar ao módulo da FPAA, e ao sistema de configuração da FPAA num único dispositivo, foi desenvolvido protótipo de uma placa de circuito impresso (PCB) através do *software* CircuitMaker. O *software* CircuitMaker foi desenvolvido pela Altium, permitindo desenvolver placas de forma mais rápida e facilitada, visto que é uma ferramenta *Open Source*, permitindo que a comunidade que utiliza este *software* partilhe não só as ideias, como todos os componentes que são criados para cada PCB. Sendo o dispositivo STA400 um módulo muito específico, foi necessário criar o respetivo *footprint*, desenvolvendo também o módulo 3D do mesmo de forma a poder apresentar no final o 3D da PCB com todos os componentes para uma melhor perspetiva final da mesma [44].

Após o desenho do esquema do sistema final tal como ilustrado na Figura 75, passou-se à edição da placa, deixando todos os conectores na zona exterior para ser de fácil acesso e os restantes componentes na restante placa, na figura seguinte encontra-se ilustrado o protótipo da placa de circuito impresso desenvolvido em 3D.

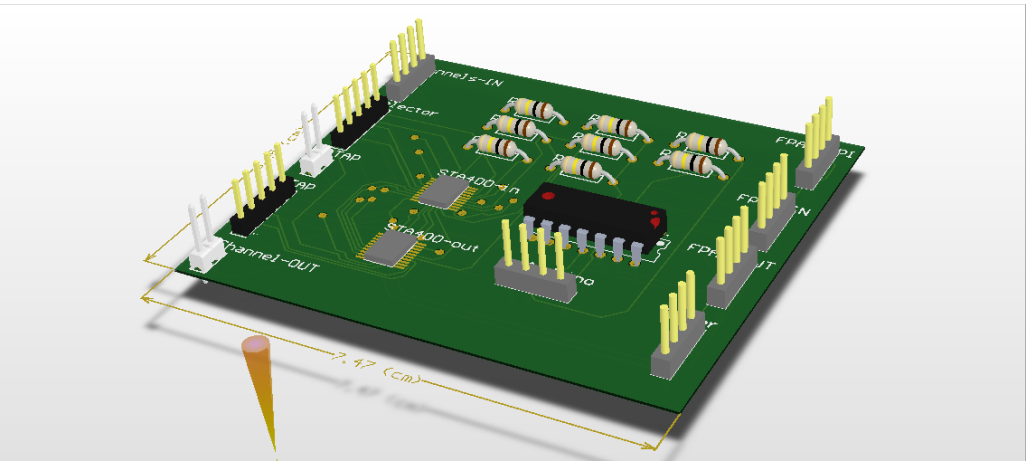


Figura 76 Placa protótipo em 3D do sistema final

De notar que a placa final prevista, teria uma dimensão de 7.47cm por 6.45cm. Sendo este um projeto protótipo de validação do sistema, a placa não foi desenvolvida fisicamente, sendo todo o sistema desenvolvido numa *breadboard* com interligação aos sistemas tal como foi apresentado até ao momento.

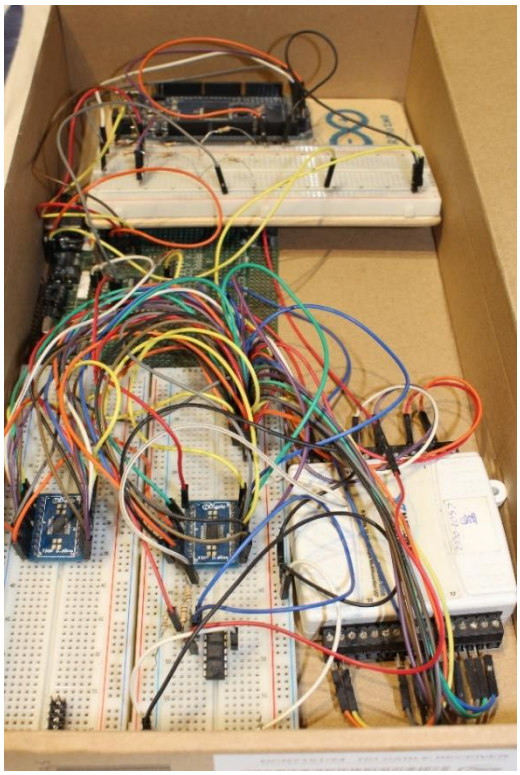


Figura 77 *Breadboard* com o sistema final

Para ser mais fácil a perceção das interligações entre todos os módulos do sistema, na tabela seguinte encontra-se a descrição de cada interligação dos diferentes módulos. Nem todos os

pinos para o sistema final se encontram a ser utilizados para validação do conceito, no entanto, estes poderão ser utilizados, sendo totalmente compatíveis com o *software*. Os pinos A1, A2 e A3 da placa de circuito impresso não se encontram ligados visto que o módulo da NI só possui 2 sinais de output analógicos, estando estes a ser utilizados para A0 e AT1. Tal como indicado anteriormente, estes pinos poderão ser ligados a um gerador de sinais por exemplo para realização de algum teste em específico.

Tabela 16 Interligação entre os diferentes módulos

Placa de circuito impresso	NI	FPAA	Arduíno
C0	P0.1	-	-
C1	P0.2	-	-
CEI	P0.5	-	-
CE	P0.4	-	-
MODE	P0.3	-	-
A0	AO1	-	-
A1	-	-	-
A2	-	-	-
A3	-	-	-
AT1	AO0	-	-
AT2	AI2	-	-
TDI	P1.1	-	-
_TDO	AI3	-	-
TMS	P0.7	-	-
TCK	P1.0	-	-
TRST	P0.6	-	-
A01	AI1	-	-
A23	AI0	-	-
ARDUINO-4	-	-	GND
ARDUINO-3	-	-	52
ARDUINO-2	-	-	50
ARDUINO-1	-	-	51
I1P	-	I1P	-
I1N	-	I1N	-
I2P	-	I2P	-
I2N	-	I2N	-
O3P	-	O3P	-
O4P	-	O4P	-
ACT	-	ACT	-
GND	GND	GND	GND
5V	5V	-	-

Estando a nível de *hardware* todo o sistema bem definido, foi iniciado o desenvolvimento a nível de *software*.

Tal como indicado anteriormente, o *software* foi desenvolvido através do ambiente gráfico *Labview*, sendo para tal necessário realizar o controlo de todos os módulos apresentados a nível de *hardware*. De forma a responder a todas as funcionalidades previstas para o *software*, serão desenvolvidos diversos processos que irão ser executados em paralelo, interligando-se para controlo global do *software*. Na Figura 78 encontram-se ilustrados os diversos processos a desenvolver.

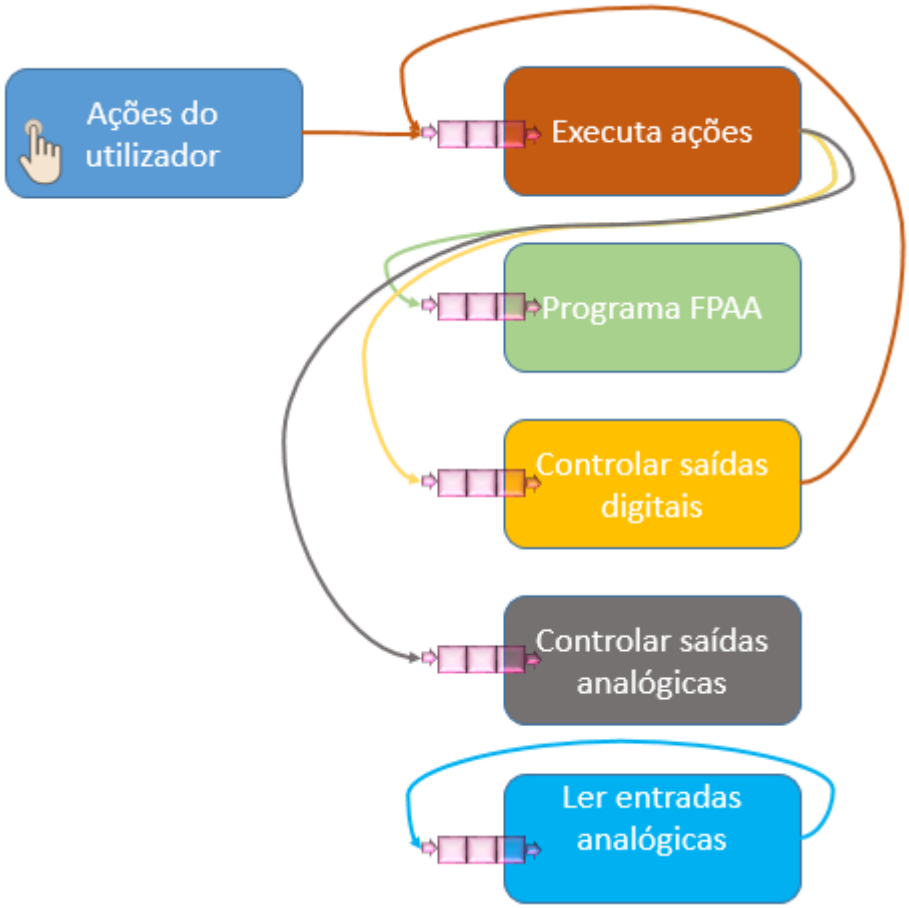


Figura 78 Processos a desenvolver para o *software*

Tal como ilustrado na figura anterior, serão desenvolvidos 6 processos que irão executar em paralelo. Alguns dos processos irão comunicar entre si, através de filas de mensagens criadas para cada um dos processos.

O processo *Ações do utilizador* irá responder através de eventos, ou seja, quando o utilizador clicar num botão, irá despertar um evento, evento esse que irá enviar uma mensagem para o processo *Executa ações.*

O Processo *Executa ações* é o responsável por processar todas as ações pretendidas pelo utilizador, assim como executar diversos procedimentos internos ou descritos pelo processo *Controlar saídas digitais*.

O processo *Controlar saídas digitais,* responderá apenas a pedidos enviados pelo processo *Executa ações*, tendo a capacidade de enviar ordens para executar algumas ações, tais como validar se os níveis de tensão a analisar se encontram dentro dos valores esperados, após executar as configurações dos módulos STA400. Este processo terá a capacidade de controlar o estado dos *multiplexers* e de controlar os pinos TAP.

O processo *Programa FPAA* terá a tarefa de receber a informação a enviar através de RS232 para o Arduino, para que este processe essa informação e configurar a FPAA.

O processo *Controlar saídas analógicas* terá a capacidade de controlar o sinal aplicado na porta A0 do STA400 de entrada e poderá controlar o sinal aplicado no pino AT1 do ATAP, o controlo será feito apenas quando é dada a ordem para alteração de estado através do processo *Executa ações*.

O único processo que não tem interação perante os restantes processos é o *Ler entradas analógicas,* este processo encontra-se em constante atualização e terá a responsabilidade de adquirir os dados analógicos e apresentar essa informação através de um gráfico.

O *software* desenvolvido contem algumas funções já apresentadas nos subcapítulos de controlo dos STA400 (subcapítulo 5.1) e de programação da FPAA (subcapítulo5.3), encontrando-se de acordo com o que foi descrito na arquitetura de software (subcapítulo 4.1.2).

Estando o Arduino totalmente programado e capaz de programar a FPAA através da interface desenvolvida em Labview tal como apresentado no subcapítulo 5.3 e as funções de realização de interface com os módulos STA400, tornou-se relevante interligar todas as funções de modo a interagir diretamente com as necessidades do utilizador.

A interface do utilizador, tal como descrito no subcapítulo 4.1.2, permite que o mesmo possa realizar diversas ações tais como, configurar a FPAA carregando um ficheiro hexadecimal da configuração desenvolvida na interface Anadigm, depurar que o sistema se encontra a realizar as ações para o qual foi programado, controlar o sistema a partir de um controlo analógico específico, monitorizar o estado do sistema (pinos abrangentes dos módulos STA400), assim como permitir a implementação livre de configurações para realização de testes através do STA400. Desta forma a aplicação terá um aspeto tal como ilustrado na figura seguinte.

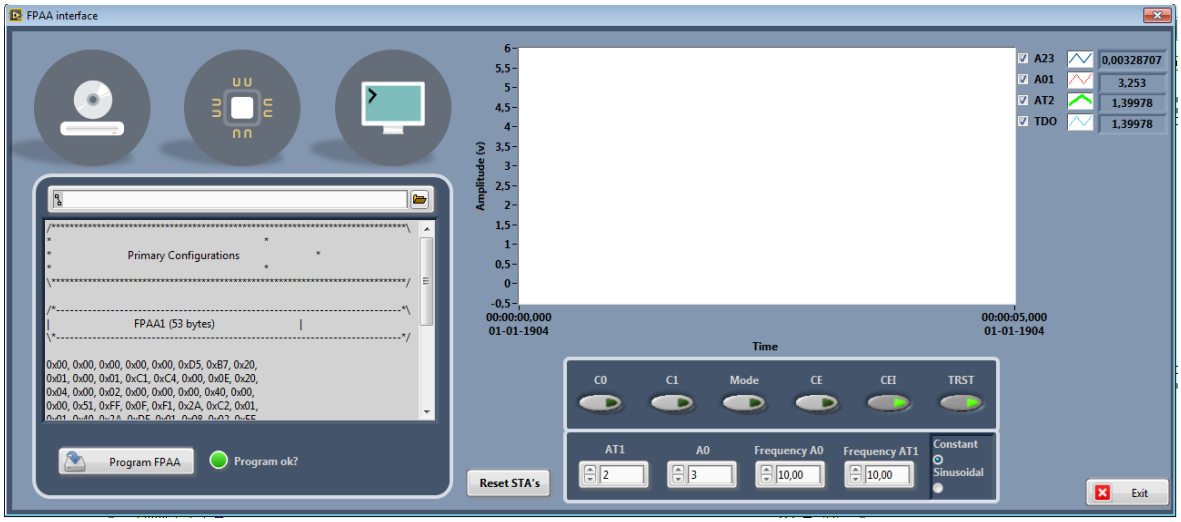


Figura 79 Aspeto do *software* final-Programar FPAA

Observando a Figura 79 é possível verificar que existem 3 menus na zona superior, um menu de configuração da FPAA, um menu de observação e controlo da FPAA e um menu de programação/configuração dos módulos STA400. Na imagem apresentada, encontra-se ativo o menu de configuração da FPAA, pelo que permite escolher um ficheiro hexadecimal através do explorador do Windows carregando-o para o indicador textual abaixo. Após o utilizador carregar o ficheiro com a configuração da FPAA, estará apto para iniciar a configuração através do botão *Program FPAA*. Terminada a configuração da FPAA o indicador *Program ok?* irá indicar se o dispositivo foi corretamente programado (o Led ficará com a cor verde) ou se não se ocorreu um erro (o Led ficará com a cor vermelha). Sendo os restantes indicadores comuns para todos os menus, a explicação será dada após a explicação de todas as opções previstas nos menus. Na figura seguinte encontra-se ilustrado o menu de observação e controlo da FPAA.

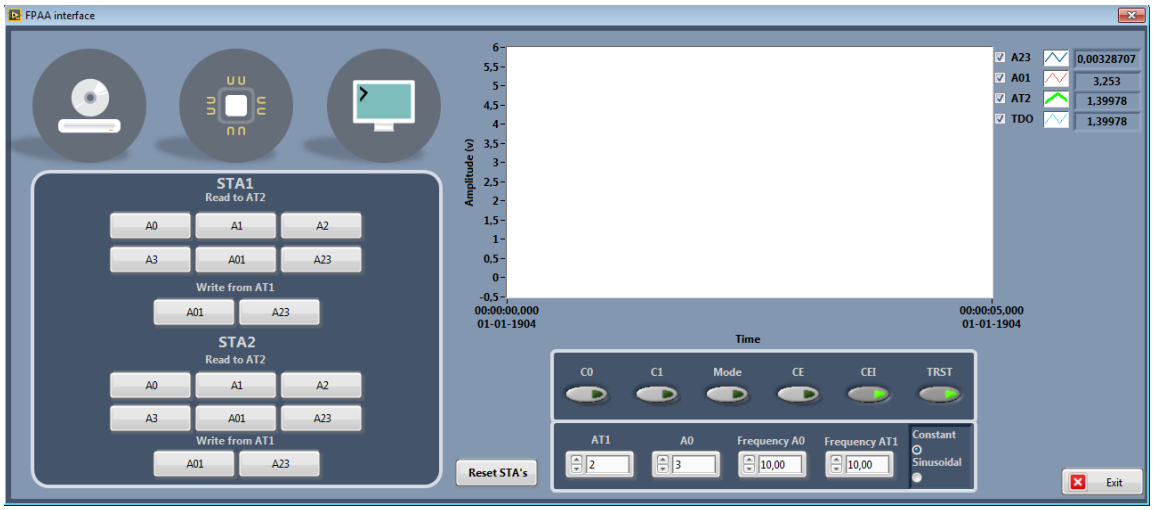


Figura 80 Aspeto do software final-Observação e controlo da FPAA

No menu apresentado na Figura 80 é possível observar que estão disponíveis 4 opções distintas, uma de leitura de sinais provenientes do *STA1-Input* (A0, A1, A2, A3, A01, A23) através do sinal AT2, uma de escrita para os pinos do STA1 através do sinal AT1, e as mesmas opções no entanto direcionado para o módulo *STA2-Output*. Desta forma, escolhendo uma das opções clicando no sinal que se pretende controlar ou observar, será enviado automaticamente a configuração para os controladores TAP dos STA's de forma a realizar a ação pretendida. Com estas opções, será possível realizar a observação e o controlo dos pinos da FPAA que se encontram interligados com os módulos STA400. Na figura seguinte encontra-se ilustrado o menu de implementação livre de envio de comandos para controlar o estado dos módulos STA400.

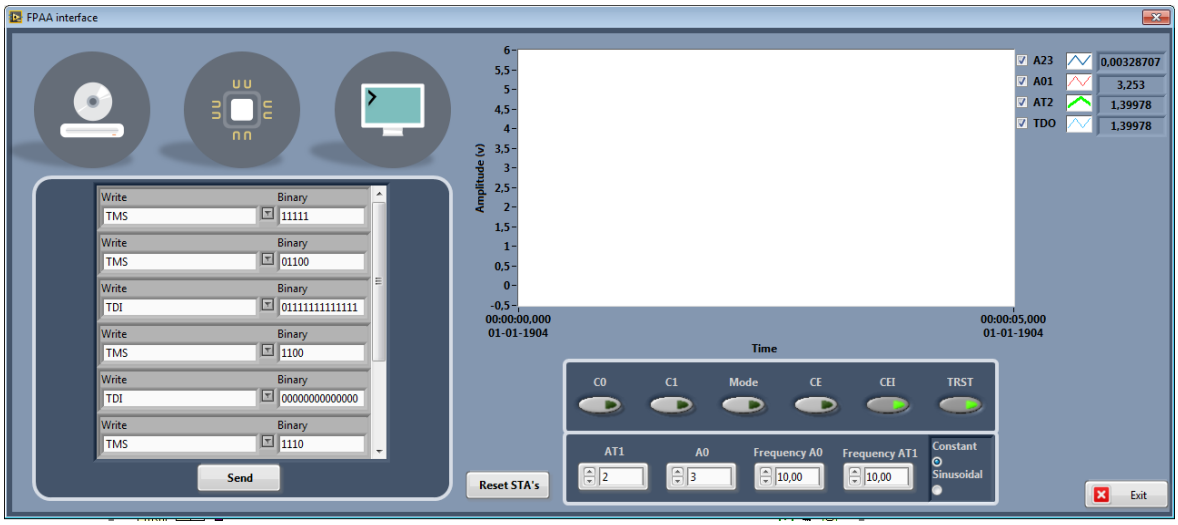


Figura 81 Aspeto do *software* final-Implementação livre STA400's

Analisando a figura anterior, é possível verificar que as opções aqui descritas são idênticas às opções apresentadas no menu de configurações utilizado para realização de testes individuais aos módulos STA400, dando a possibilidade ao utilizador de definir que mensagens pretende enviar para o TMS e TDI, estas ações necessitam de conhecimento prévio do tipo de mensagens a enviar para controlar os módulos.

Tal como foi indicado anteriormente, todos os indicadores que se apresentam do lado direito do *software*, é comum para todos os menus. Na zona superior, encontra-se um gráfico que terá acesso em tempo real do estado dos sinais A23, A01, AT2 e TDO do ponto de vista da placa de circuito impresso apresentado anteriormente no nível de *hardware*, tendo a possibilidade de obter os valores atuais ao lado da legenda de cada sinal. O botão *Reset STA's* envia uma mensagem para o TMS de cinco 1's consecutivos fazendo com que o estado do controlador TAP reinicie, tendo a capacidade de perceber em que estado se encontram na máquina de estados. Os botões C0, C1, Mode, CI , CEI e TRST, tal como indicado anteriormente, são comuns aos dois módulos STA400, sendo as funcionalidades de cada um para controlo do estado dos *multiplexres* (já descritos na Tabela 7) com a exceção do sinal TRST que é utilizado como *Reset* dos controladores TAP dos STA400 (este sinal deverá manter-se no estado lógico *true*, caso contrário estará sempre em *reset*). Os restantes controlos permitem a geração de sinais analógicos para os pinos AT1 e A0, com a capacidade de gerar dois tipos de sinais distintos (constante ou sinusoidal, sendo necessário definir a frequência de cada sinal quando está ativa a geração de ondas sinusoidais).

Uma vez construído o ambiente gráfico que que o utilizador interaja com os diferentes módulos, torna-se relevante realizar todo o código que irá gerir todas essas informações. De seguida será dada uma explicação do código desenvolvido para realizar as ações pretendidas. O código desenvolvido foi baseado no diagrama de blocos apresentados na Figura 78, desta forma todas as ações do utilizador no ambiente gráfico irá despoletar um evento nos processos de forma a realizar a ação pretendida.

O código encontra-se dividido em 6 processos, sendo necessário a troca de informação entre os mesmos, foram criadas diversas filas de mensagens para tratar cada um dos processos poder comunicar com outro, ou consigo mesmo para executar uma ação específica.

O processo *ações do utilizador* irá gerar eventos sempre que o utilizador execute uma ação no ambiente gráfico, enviando para o processo *Executa ações* o que deverá efetuar. De seguida serão apresentados os processos realizados para as ações mais relevantes.

Uma das ações importantes para este trabalho passa pela realização da programação da FPAA, sendo também necessário obter o *feeback* do sistema de forma a perceber se a programação da FPAA foi realizada corretamente ou incorretamente. Na figura seguinte encontra-se lustrado o processo de execução para realizar esta ação.

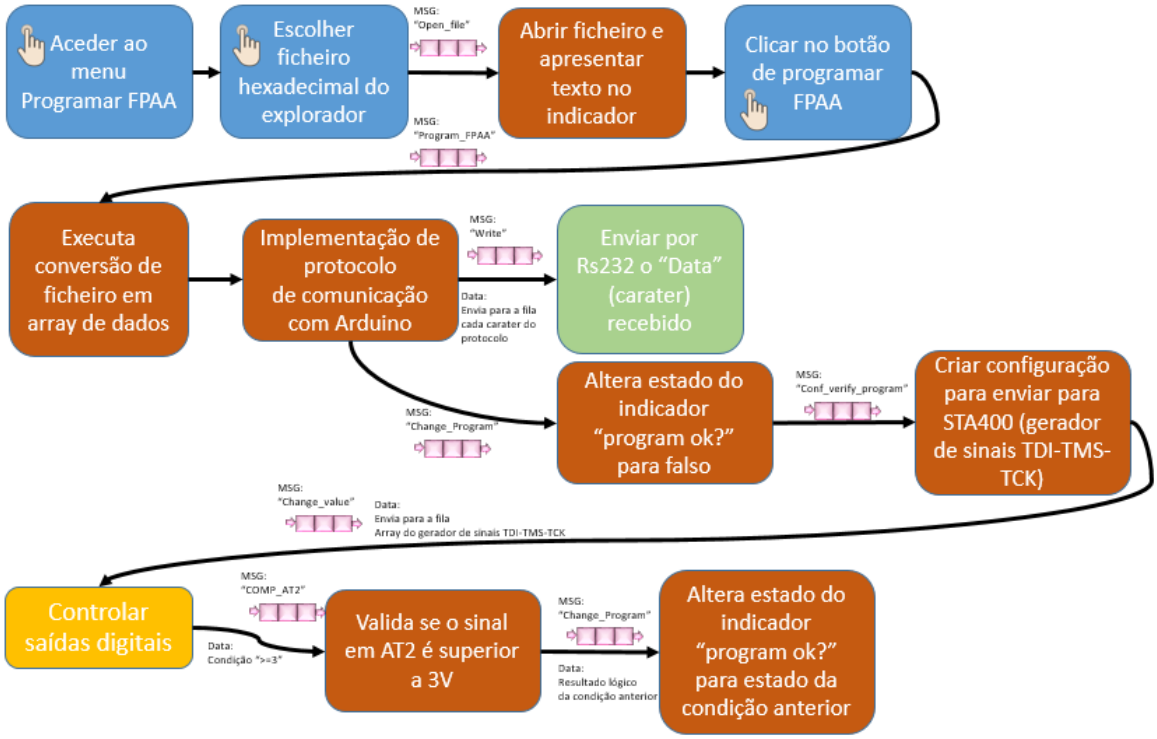


Figura 82 Processos executados para configuração da FPAA

Analisando a Figura 82 é possível verificar que existem 4 processos que irão entrar em funcionamento de forma a realizar a programação e a validação da configuração da FPAA. Os processos que irão iniciar o funcionamento são os seguintes: *Ações do utilizador* (com a cor azul), *Executa ações* (com a cor laranja), *Programa FPAA* (com a cor verde) e *Controlar saídas digitais* (com a cor amarela).

O utilizador, não estando no menu apropriado para a realização desta ação, terá de aceder ao menu para programar a FPAA clicando no botão mais a esquerda (ilustrado com um CD) no ambiente gráfico do *software*.

O utilizador ao clicar nesta opção será apresentado o respetivo menu de programação, onde o mesmo poderá carregar o ficheiro hexadecimal adquirido como output da configuração no *software* Anadigm, através do botão explorador de ficheiros.

Após o carregamento do ficheiro será enviada a mensagem *Open_file* através da fila de mensagens para o processo *Executa ações*, para que este execute a abertura do ficheiro carregado pelo utilizador apresentado a informação contida no ficheiro no indicador criado para o efeito.

De seguida o utilizador estará apto para iniciar a programação, sendo apenas necessário clicar no botão *Program FPAA*. Esta ação do utilizador irá enviar a mensagem *Program_FPAA* para o processo *Executa ações*, sendo este o responsável por converter os dados contidos no ficheiro carregado para um *array* de dados tal como explicado anteriormente no subcapítulo 5.3 (ver função da Figura 72).

Obtendo o *array* de dados do ficheiro, será iniciado o tratamento da informação de forma a gerar a mensagem protocolar definida para comunicação com o Arduino (ver função apresentada na Figura 73).

Terminado o processo de geração da mensagem protocolar, será enviado carater a carater através do processo *Programa FPAA* a mensagem global, sendo este envio realizado através de RS232 para o Arduino.

Durante o tempo em que se encontra a realizar a programação da FPAA, será dada a ordem para alteração do estado do led indicador de programação, ficando este no estado lógico desligado.

De seguida será necessário realizar a leitura do pino ACT da FPAA de forma a validar que a configuração da FPAA foi corretamente entendida pela mesma. Desta forma seria necessário aceder diretamente ao pino da FPAA através de um sinal ligado ao módulo da NI, no entanto este não se encontra diretamente ligado ao módulo. Visto que o sinal ACT foi ligado ao pino A1 do módulo STA400-Out, torna-se possível configurar os módulos de forma a ligar o pino A1 a AT2 (estando este diretamente ligado ao módulo da NI). Para ligar o pino A1 a AT2 do módulo STA400-out será necessário criar a seguinte sequência de tramas (Tabela 17) a enviar para o controlador TAP.

Tabela 17 Configurar pino ACT a ligar a AT2

Sinal	Dado
TMS	11111
TMS	01100
TDI	11111111111111111110111111111111111000
TMS	1100
TDI	00000000000000010000000010000000000000000000000
TMS	11100
TDI	11111111111111111111111111111111111000
TMS	100

A configuração criada reflete as oito ações seguintes: (1) reiniciar o controlador TAP para que seja possível saber o estado da máquina de estado do controlador; (2) Configurar a máquina de estados para o estado *Shift-IR*; (3) Configurar o módulo STA400-in para trabalhar como *bypass* e o módulo STA400-out para executar a instrução de amostragem (*Sample*); (4) Configurar a máquina de estados para o estado *Shift-DR*; (5) Configurar os interruptores do ABM de forma a ligar o pino A1 ao sistema *core* (mantendo a funcionalidade normal do circuito) e ligado a AB2, esta configuração também configura o TBIC de forma a ligar AT2 a AB2 e AB1 a VClamp. Desta forma o pino A1 irá ser ligado a AT2 tal como pretendido; (6) Configurar a máquina de estados para o estado *Shift-IR*; (7) Configurar o módulo STA-400-in para *bypass* e o módulo STA-400-out para *INTESTD* que irá executar as ações carregadas anteriormente no registo de dados, ligando os pinos ao sistema core de forma a executar a ação para o qual se destina (*multiplexer*) normalmente; (8) Configura a máquina de estados para o estado *Run-Test-Idle* ficando o sistema a funcionar com a configuração atual.

Estando a configuração de interligação dos pinos ACT a AT2 do módulo STA400-out totalmente validada, será necessário gerar os sinais de TMS, TCK e TDI, utilizando a função já apresentada anteriormente (ver Figura 47, Figura 48 e Figura 49). Após a geração dos

sinais de TDI, TMS e TCK será enviada essa informação para o processo responsável de gerar os sinais digitais para o módulo da NI, processo esse denominado de *Controlar saídas digitais*.

Após a finalização da configuração dos módulos STA400, será enviado para o processo *Executa ações* a mensagem *COMP_AT2* com o dado *>=3*, de forma a validar se o sinal AT2 lido pelo módulo da NI está com um valor superior a 3V.

Dependendo do resultado da operação anterior, será controlado o led indicativo *Program ok?*, ficando ligado no caso da programação da FPAA ser bem-sucedida, caso contrário o led manter-se-á apagado.

Com a apresentação desta ação, já foi possível perceber a importância de utilização da infraestrutura IEEE1149.4 para este trabalho. Para o sistema atual, torna-se possível aceder aos pinos diretamente na FPAA, podendo validar através de um multímetro o valor de tensão no pino ACT. No entanto num sistema mais complexo, ou com um *package* diferente, torna-se impossível aceder aos pinos, sendo esta funcionalidade muito importante para o teste e depuração de sistemas. Estando a configuração da FPAA totalmente finalizada, o utilizador poderá ter necessidade de saber o estado dos pinos da FPAA, ou controlar o estado de cada pino individualmente, utilizando a infraestrutura IEEE1149.4 o utilizador terá a capacidade total para testar e depurar sem ter necessidade de se deligar do sistema, o que nem sempre é possível, principalmente numa placa já desenvolvida.

Desta forma, o utilizador poderá controlar o estado de *STA400-in* e *STA400-out* para que possa controlar e observar a configuração realizada na FPAA. Para tal, o utilizador deverá aceder ao menu de observação e controlo da FPAA escolhendo o tipo de ação que pretende realizar e qual o dispositivo correspondente. Uma vez que o número de opções disponíveis é elevado, irá ser explicado apenas uma das opções que contempla opções ainda não explicadas em passos anterior. Visto que anteriormente foi explicado como realizar a configuração dos módulos para observar o estado de ACT que está ligado ao *STA400-out* através do AT2, irá ser dada a explicação de controlo de um pino da FPAA através de AT1 do módulo *STA400-in* (ver Figura 83).

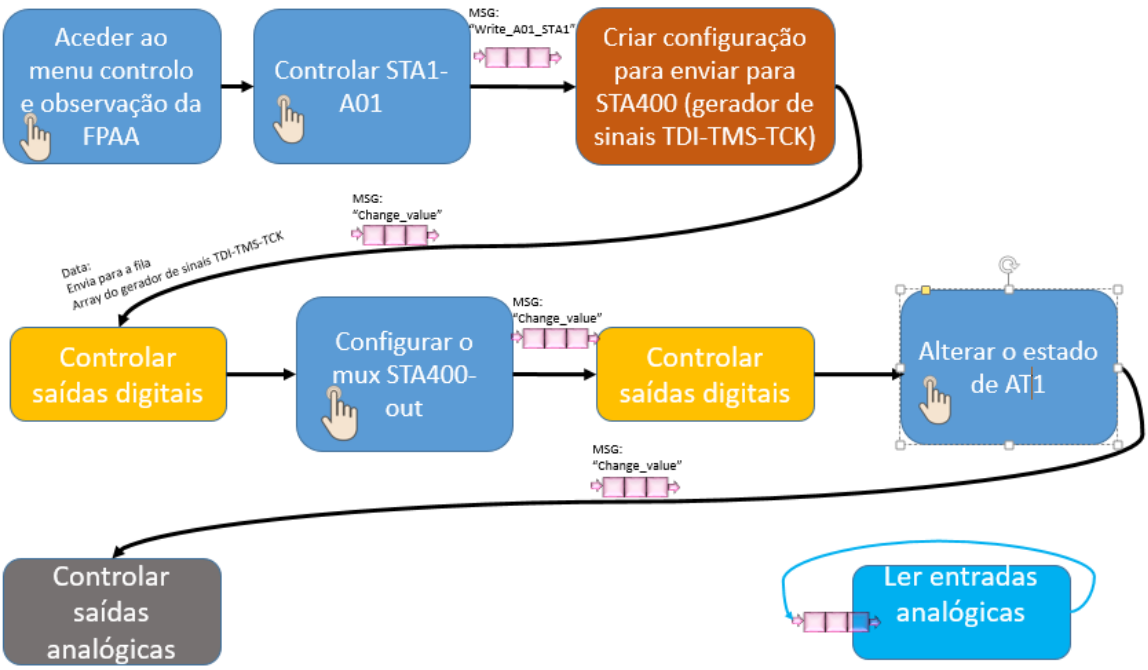


Figura 83 Processos executados para controlo de I1P da FPAA através de AT1

Para controlar o sinal de I1P será necessário validar a que canal do *STA400-in* se encontra ligado, analisando o esquema elétrico (ver Figura 75) verifica-se que I1P está ligado ao pino A01 do STA400, desta forma o utilizado deverá aceder ao menu de controlo e observação da FPAA através do *software* (Menu ilustrado com um microcontrolador) onde deverá escolher a opção de controlo de STA1 de forma a escrever para o pino A01 através de AT1. Essa ação irá despoletar uma ação no processo *Executa ações* através do envio da mensagem *Write_A01_STA1* pela fila de mensagens.

O processo *Executa ações* irá dar ordem para configurar os módulos STA400 de forma a ligar AT1 a A01 para controlar o sistema. Na Figura 12 encontra-se definida a sequência de tramas que são configuradas par enviar para o controlador TAP.

Tabela 18 Configurar pino AT1 a ligar a A01 de STA400-in

Sinal	Dado
TMS	11111
TMS	01100
TDI	01111111111111111000111111111111111111
TMS	1100
TDI	00000000000000000000000010000000000000000000001
TMS	11100
TDI	00000000000000000001111111111111111111
TMS	100

A configuração criada reflete as oito ações seguintes: (1) reiniciar o controlador TAP para que seja possível saber o estado da máquina de estado do controlador; (2) Configurar a máquina de estados para o estado *Shift-IR*; (3) Configurar o módulo STA400-in para executar a instrução de amostragem (*Sample*) e o módulo STA400-out para funcionar como *bypass* de forma a não interferir no sistema; (4) Configurar a máquina de estados para o estado *Shift-DR*; (5) Configurar os interruptores do ABM de forma a ligar o pino A01 ao sistema *core* (mantendo a funcionalidade normal do circuito) e ligado a AB1, esta configuração também configura o TBIC de forma a ligar AT1 a AB1 e AB2 a VClamp. Desta forma o pino A01 irá ser ligado a AT1 tal como pretendido; (6) Configurar a máquina de estados para o estado *Shift-IR*; (7) Configurar o módulo *STA400-in* para *EXTEST* que irá executar as ações carregadas anteriormente no registo de dados, estando desligado do sistema core para que não haja dois sinais analógicos distintos no mesmo pino (causando problemas ao módulo), neste ponto o módulo *STA400-out* ficará configurado para funcionar como *BYPASS* não influenciando o sistema, apenas mantendo o funcionamento do sistema core; (8) Configura a máquina de estados para o estado *Run-Test-Idle* ficando o sistema a funcionar com a configuração atual. Estando a configuração de interligação dos pinos AT1 a A01 do módulo *STA400-in* totalmente validada, será necessário gerar os sinais de TMS,

TCK e TDI, utilizando a função já apresentada anteriormente (ver Figura 47, Figura 48 e Figura 49). Após a geração dos sinais de TDI, TMS e TCK será enviada essa informação para o processo responsável de gerar os sinais digitais para o módulo da NI, processo esse denominado de *Controlar saídas digitais*.

Terminado o envio da configuração dos módulos STA400, o sistema está apto para receber o sinal analógico proveniente em AT1 de forma a injetar esse mesmo sinal em I1P. No entanto para o utilizador ter noção do estado da saída de FPAA, torna-se necessário configurar o *MUX* de AT1, para tal o utilizador deverá configurar os sinais digitais C0, C1, Mode, CE e CEI de forma a ligar a saída da FPAA correspondente ao sistema configurado. No exemplo apresentado, seria selecionado o canal AI0 ligado a A01 do STA400-out de forma a ligar o pino da FPAA O3P ao módulo da NI.

Tabela 19 Configurar MUX para ligar A01 de STA400-out a O3P da FPAA

CE	Mode	C1	C0	A01	A23
	0	0	0	A0	A2
Not equal to CEI	0	0	1	A1	A2
	0	1	0	A0	A3
	0	1	1	A1	A3
	1	0	0	A0	Not connected
	1	0	1	A1	Not connected
	1	1	0	Not connected	A2
	1	1	1	Not connected	A3
Equal to CEI	x	x	x	Not connected	Not connected

Nota:
CE=0 e CEI=1

A configuração dos sinais digitais anteriomente indicados, irá despoletar ações no processo *Controlar Saídas digitais*. Após a realização de todas as configurações, o utilizador poderá facilmente controlar o sinal analógico de AT1 através do controlo para o efeito no *software*, gerando acções no processo *Controlar Saídas analógicas* de forma a aplicar o sinal indicado na saída do módulo da NI. Note que o sinal antes de chegar à FPAA será converido através da função indicada em (2), pelo que 5V na saída do módulo da NI irá corresponder a um sinal de 3.3V na entrada da FPAA. Para terminar, o utilziador poderá observar o estado do sinal A01 do *STA400-out* através do gráfico. Os valores do gráfico encontram-se sempre em atualização, sendo o processo *Ler entradas analógicas* o responsável por adquirir os sinais e apresentalos no gráfico.

5.5. IMPLEMENTAÇÃO DE EXPERIMENTAÇÃO REMOTA

De forma a responder ao objetivo definido inicialmente, para desenvolvimento de uma infraestrutura que permita integrar todos os elementos desenvolvidos com vista à experimentação remota, foi ativada a funcionalidade de publicação *Web* do *Labview* (*Web publisher tool*). Esta ferramenta permite criar um serviço *web*, onde será gerada uma página em *html* com a aplicação desenvolvida. Desta forma, a aplicação irá correr num servidor onde será ligado o *hardware* desenvolvido e cada cliente poderá aceder ao hardware a partir de qualquer ponto com ligação à internet.

O utilizador deverá desenvolver a configuração da FPAA no *software* da Anadigm, gerando o correspondente ficheiro hexadecimal tal como explicado no subcapítulo5.3. Após o ficheiro hexadecimal estar criado, o utilizador deverá envia-lo para o servidor através de uma *Cloud* partilhada entre ambos, por exemplo. Estando o ficheiro hexadecimal no servidor, o utilizador poderá aceder à página *web* tal como ilustrado na Figura 84, onde poderá configurar a FPAA e validar o seu funcionamento *online*.

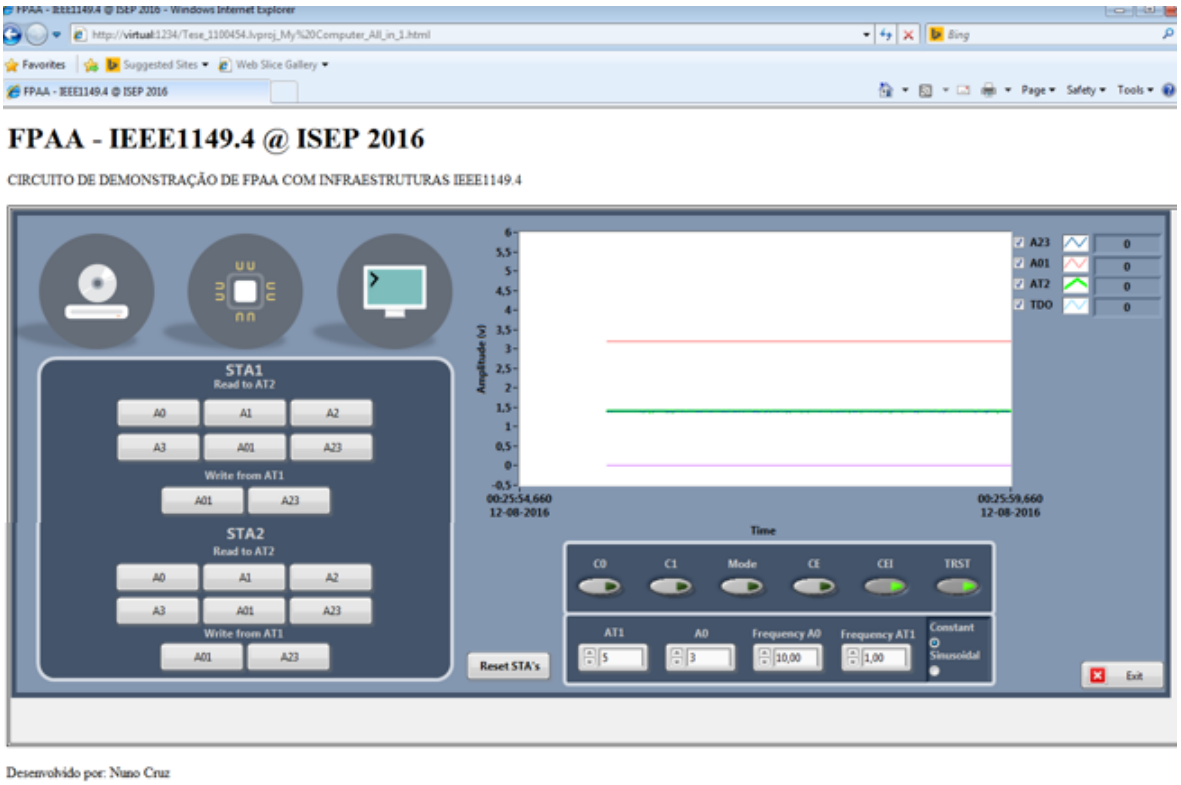


Figura 84 Página web de acesso ao *software*

Sendo este um sistema que não permite o acesso a mais do que um utilizador, o *website* irá proteger-se de múltiplos acessos. Quando um utilizador pretende aceder ao equipamento, quando este já se encontra acedido por um outro utilizador, irá ser apresentada a mensagem *Waiting for control: Either the server is locked or another client has control*. Desta forma os utilizadores ficaram com acesso restrito através de uma fila.

6. VALIDAÇÃO DA SOLUÇÃO

Neste capítulo será apresentado os diversos testes que serão realizados para validação da solução. Para tal será necessário injetar sinais analógicos na FPAA (indiretamente, ou seja passando pelo módulo STA400) e analisar a sua saída através do ATAP, validando que o que foi configurado na FPAA está correto. Será também necessário validar que é possível injetar sinais através do ATAP, tendo acesso a qualquer pino da FPAA. Idealmente, será utilizado a validação com comparação entre o sinal simulado e o sinal de resposta ao sinal injetado. Tanto o sinal injetado como o sinal de resposta deverão ser validados através do ATAP.

6.1. VALIDAÇÃO DO SISTEMA FINAL

Para validação do sistema final, foram executados dois testes. O primeiro teste implica a utilização do sistema total com utilização do módulo da NI para injetar e observar sinais, aplicando uma configuração de *bypass* na FPAA. Visto que o módulo da NI não permite a geração de sinais de frequências muito elevadas, assim como a sua aquisição, será utilizado um osciloscópio na saída de AT2 do módulo *STA400-out* e um gerador de sinais na entrada de AT1 do módulo *STA400-in*, de forma a controlar a injeção e a possível observação de sinais de alta frequência. Desta forma, o segundo teste irá conter uma configuração na FPAA, que permita trabalhar e validar o sistema com um filtro passa baixo.

Para o teste inicial, foi desenvolvida a configuração do sistema da FPAA de forma a colocar o sinal de I1P na saída de O3P. Considerando esta a ação pretendida, acedeu-se ao *software* da Anadigm, e realizou-se a configuração tal como ilustrado na figura seguinte.

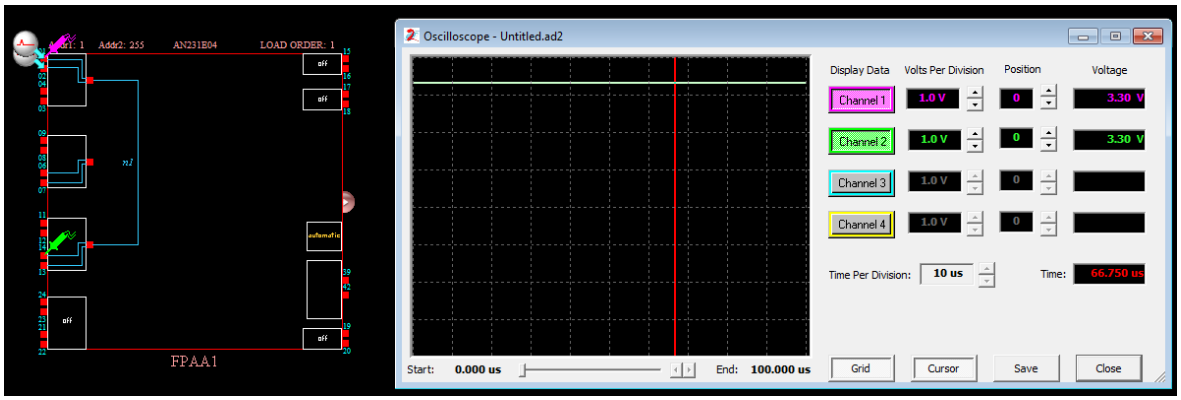


Figura 85 Configuração de FPAA – Primeiro teste do sistema final

Terminada a configuração, passou-se para a validação da configuração, desta forma foi iniciada uma simulação onde é possível validar que o sinal injectado em I1P (3.3V) é igual ao sinal de saída O3P. Estando a validação da configuração da FPAA bem executada, passa-se à criação do ficheiro hexadecimal, para que possa ser possível o envio para a FPAA através do *softwre* de teste.

No *software* de teste o utilizador deverá escolher o ficheiro hexaecimal criado e clicar em *Program FPAA*, iniciando o modo de programação. Assim que o dispositivo esteja programado, o utilizador receberá a informação através do Led indicativo *Program OK?*.

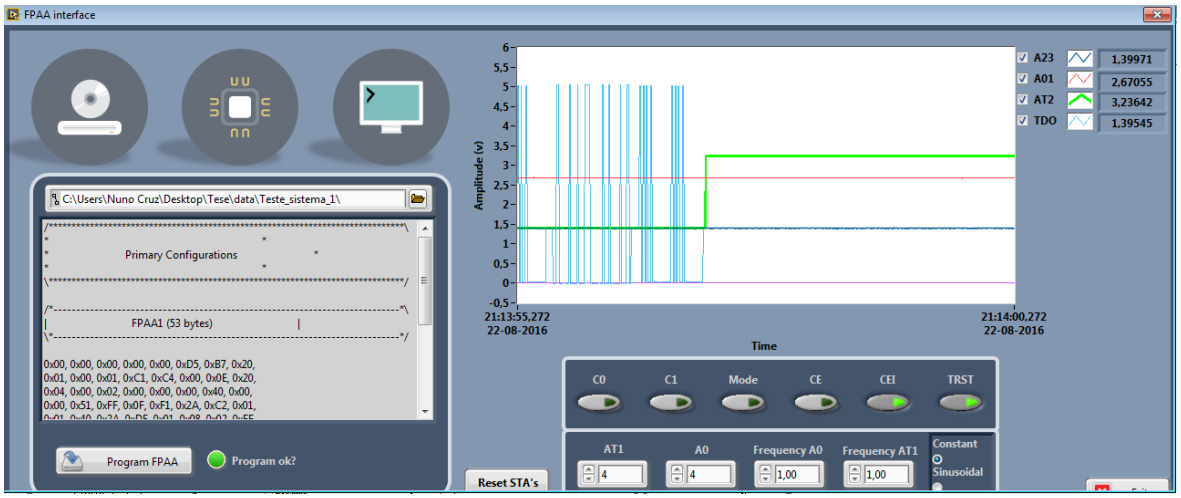


Figura 86 Configuração da FPAA – Primeiro teste do sistema final

Tal como se descreveu anteriormente, o dispositivo irá configurar o STA de saída de forma a conseguir ler em AT2 o sinal de A1 no STA de entrada. No gráfico anterior é possível verificar que quando o dispositivo está configurado, o sinal em AT2 ficará a 3.3V.

Nesse momento o utilizador poderá controlar os sinais de entrada e de saída da FPAA. Sendo a configuração deste dispositivo programado para o sinal de saída ser igual ao sinal de entrada, aplicando um sinal de 5V em A0, o sinal será convertido para 3.3V, podendo ser observado esse valor em A01 (considerando que a configuração do *multiplexer* está definido de forma a ligar A0 a A01 para os dois módulos). Na Figura anterior, é possível observar que o sinal de A0 injetado para o dispositivo *STA400-in* é de 4V, sendo este sinal convertido para 2.64V através do divisor resistivo, sendo este o valor apresentado em A01 do módulo *STA400-out*. Para o sistema aqui presente, encontra-se a analisar os sinais de A01, no entanto num sistema mais complexo, não seria viável observar todos os sinais dos dispositivos. Então este sistema permite analisar o sinal de A01 em AT2, sendo este o pino de acesso obrigatório para analisar qualquer um dos pinos do módulo STA400. No teste seguinte, configurou-se o dispositivo para analisar o sinal presente no sinal de A01 através de AT2.

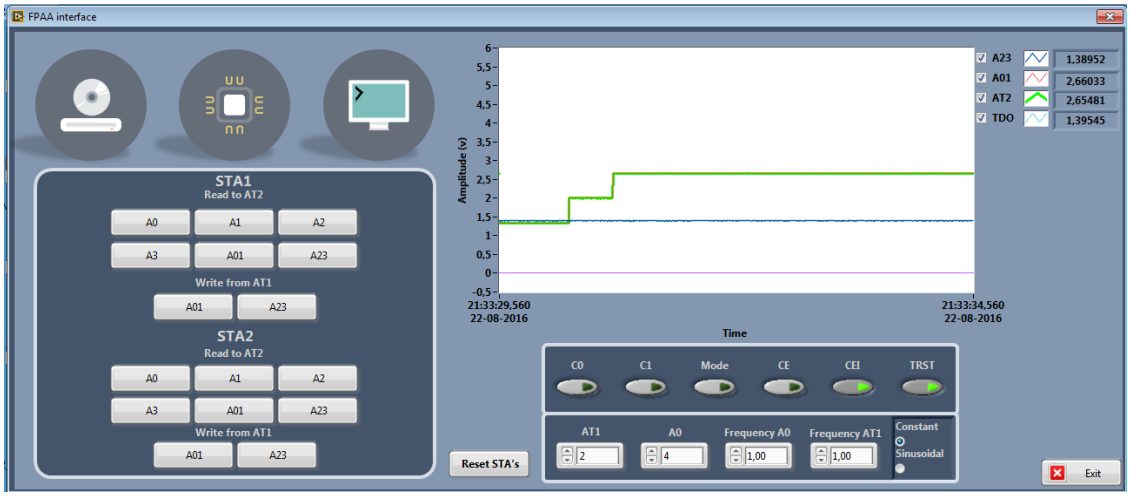


Figura 87 Observação de A01 (STA-out) em AT2 (STA-out)

Na figura anterior é possível observar que o sinal de AT2 segue o sinal de A01, sendo o sistema completamente observável através de AT2. Para além de observável, o sistema também é controlável, para tal o sinal injetado em AT1 poderá ser ligado a A01 ou a A23 de ambos os módulos. No teste apresentado de seguia, ilustra a capacidade do sistema controlar através de AT1 o sinal de A01 do módulo *STA400-in*. Estando o *multiplexer* configurado para ligar A0 a A01, o *STA400-in* em funcionamento normal estaria a colocar o nível de tensão 0V no sinal de A01, sendo este sinal transmitido para a FPAA. Como a FPAA está configurada para fazer *bypass* ao sinal, o sinal à entrada será colocado na saída do dispositivo, sendo este sinal transmitido para o A0 do *STA400-out* de forma a coloca-lo em A01.

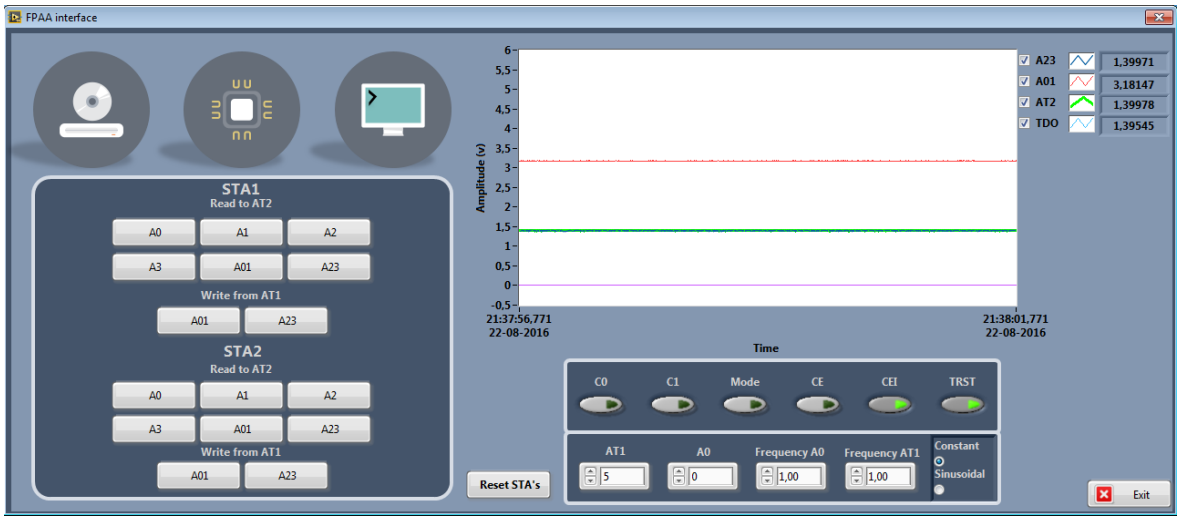


Figura 88 Controlo de A01 (STA-in) através de AT1 (STA-out)

Analisando a figura anterior, é possível verificar que A01 encontra-se com o nível de tensão de aproximadamente 3.3V em vez de 0V. Este sinal é proveniente do sinal AT1 que se encontra a 5V (e que de seguida é convertido para 3.3V), visto que o dispositivo de STA400-in está configurado para ligar AT1 a A01, transmitindo o sinal em AT1 para a FPAA.

Para além destes exemplos, é possível controlar e observar qualquer sinal que se encontra nos botões do *software*, tal como indicado anteriormente, sendo assim comprovado que o sistema permite observar e controlar uma FPAA, sem que seja obrigatório aceder aos pinos diretamente. Também é de notar que o exemplo de controlo aqui apresentado nunca seria possível num dispositivo sem interface IEEE1149.4, visto que o sinal de A01 estaria sempre ligado a A0, não sendo possível injetar qualquer sinal no dispositivo.

Visto que o sistema de aquisição e controlo não é o ideal para sinais com frequências muito elevadas, no teste seguinte encontra-se ilustrado uma configuração da FPAA para funcionar como comparador, sendo o sinal de AT1 injetado através de um gerador de sinais, e observado o resultado em AT2 através de um osciloscópio.

No presente teste, a FPAA foi configurada com um funcionamento de comparador. A ideia será gerar um sinal analógico de 0 a 1.5 Volts sinusoidal no pino de entrada, sendo este convertido para o sinal de 2V sempre que o sinal sinusoidal for inferior a 1 Volt, caso contrário ficará com 0V.

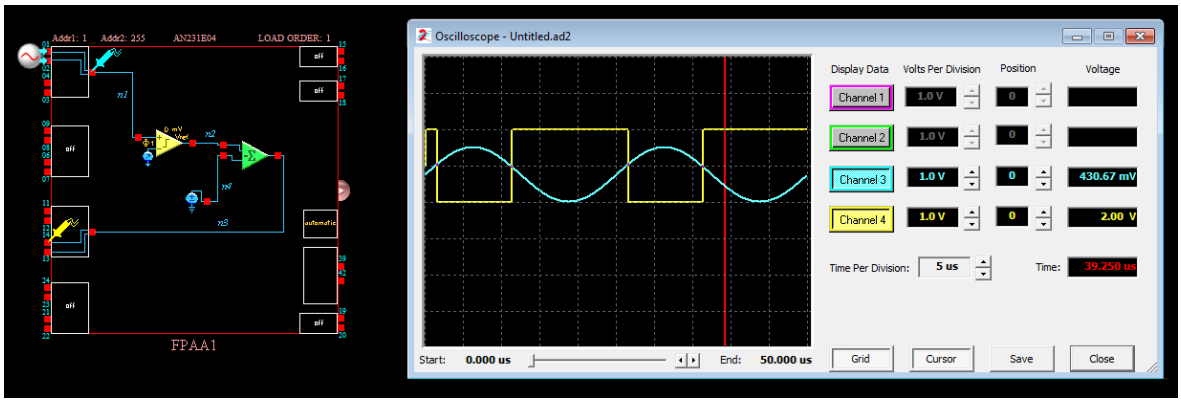


Figura 89 Configuração da FPAA– Segundo teste do sistema final

Na figura anterior é possível observar, através de simulação, que o sinal à entrada é uma onda sinusoidal de 0 a 1.5V, sendo o sinal à saída convertido para valores entre 0 e 2V dependendo do estado do sinal à entrada. Após a realização da simulação, criou-se o código hexadecimal de forma a programar a FPAA através do *software* em *Labview*.

Estando a configuração pronta a ser enviada para a FPAA, realizou-se algumas alterações a nível de *hardware*, de forma comprovar que o sistema está a funcionar tal como era esperado. Desligando o sinal de controlo da NI que liga a A0 do STA400-in e ligando um gerador de sinais ao pino A0 do STA400-in, gerou-se uma onda sinusoidal com as características representadas na figura abaixo.

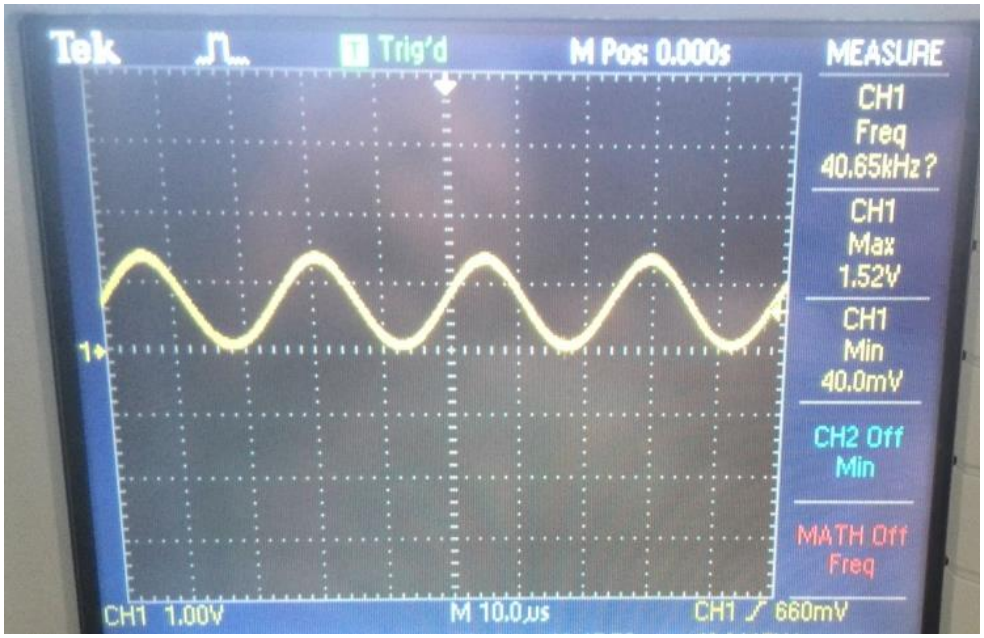


Figura 90 Onda gerada no pino A0 *STA400-in*

Note que o sinal é idêntico ao sinal aplicado na simulação, com frequência de aproximadamente 40kHz, com amplitude máxima de 1.52V e mínima de 40mV. Desta forma pretendeu-se validar através do AT2 do módulo *STA400-out*, o estado de A0, para tal foi ligada a ponta de prova do osciloscópio ao sinal AT2 de *STA400-out* e no *software* configurou-se o sistema para analisar A0 de *STA400-in* a partir de AT2. A onda gerada em AT2 pode ser analisada na figura abaixo, validando que o sinal é idêntico ao anterior.

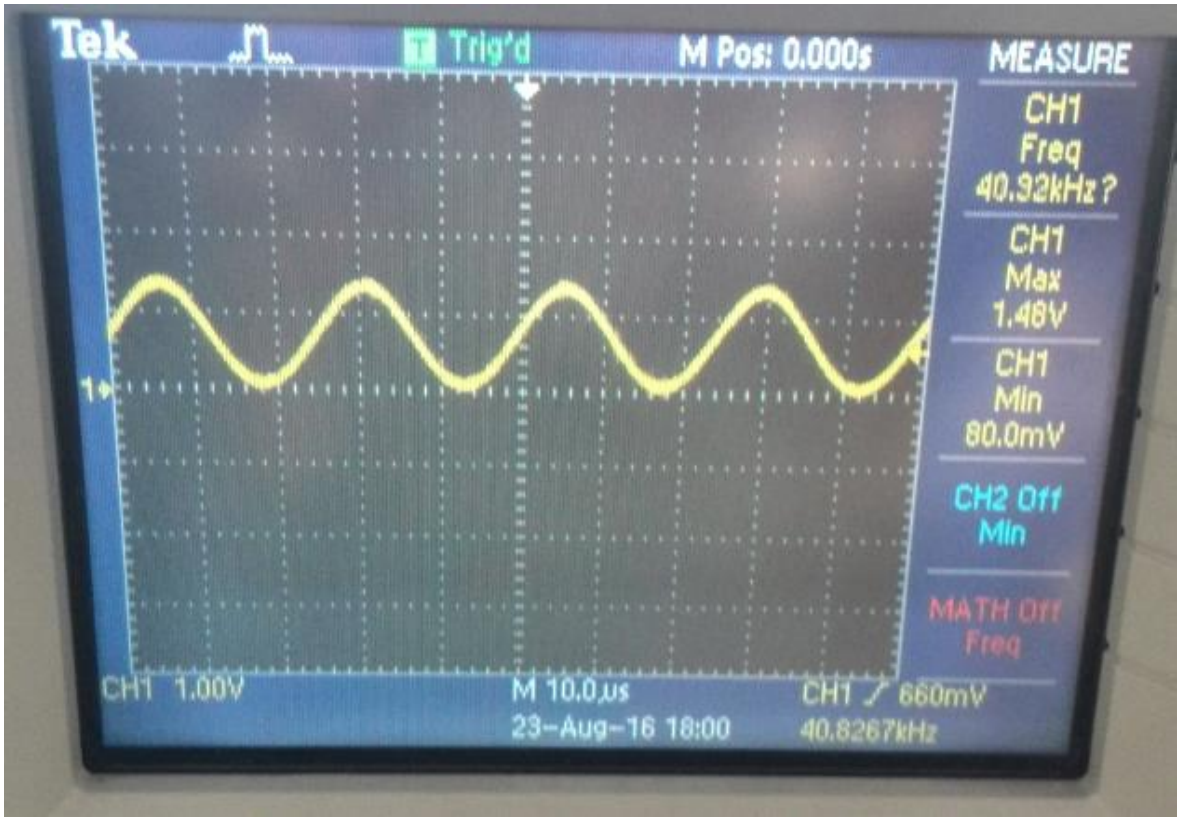
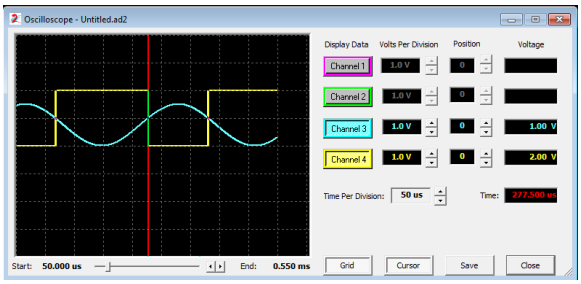


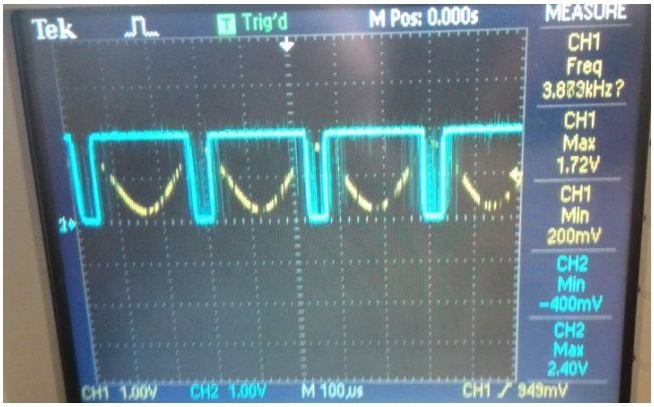
Figura 91 Onda analisada no pino AT2 *STA400-out*

Com esta análise comprova-se que o sistema permite observar os sinais injetados sem necessidade de acesso, ou alteração do circuito (note que o circuito foi alterado apenas para injetar um novo sinal, sem ser proveniente do módulo da NI).

Para que possa ser validado que o sistema também permite realizar um controlo do sistema, foi injectado em AT1 do *STA400-in* o sinal com um novo valor de frequência (3.8KHz), mas com valor de amplitude igual e configurados os módulos para injectar em A01 do *STA400-in* o sinal de AT1, sendo assim este o valor injectado na FPAA, podendo ser observado o resultado em A01 do *STA400-out* visto que este se encontra a funcionar como *bypass*.

Em simulação, o sistema também estaria apto para realizar a acção para o qual foi configurado, tal como ilustrado na figura seguinte, no entanto em ambiente prático isto não se verifica. Na figura abaixo do lado esquerdo encontra-se a simulação realizada para a frequência de 3.8 KHz, onde é possível verificar que o dispositivo reage tal como foi idealizado, no entanto na figura do lado direito, onde se pode observar o sinal injectado com a cor amarela e o sinal observado com a cor azul, conclui-se que o resultado é diferente.





Assim sendo, através da análise anterior, é possível validar que o sistema é capaz de controlar o sistema através de AT1 sem necessidade de acesso direto aos pinos dos módulos.

Com este resultado, é possível tambem concluir que este sistema se torna bastante importante, para validar que o sistema está a agir para o qual foi programado. Desta forma a simulação irá ajudar a validar a ideia pretendida, sendo este sistema capaz de validar que em ambiente real o sitema realiza as acções para o qual foi desenhado.

7. CONCLUSÕES

Ao longo deste trabalho foram apresentadas várias conclusões que permitem validar a importância do desenvolvimento deste projeto.

Até ao momento, as FPAA's apenas permitiam realizar a configuração do dispositivo, não sendo possível realizar a validação funcional do sistema, sendo uma das grandes limitações deste tipo de dispositivos, algo que já não acontece com dispositivos configuráveis digitais tal como a FPGA. Face à grande vantagem de utilização de uma infraestrutura IEEE1149.1 para apoio a depuração dos circuitos digitais, foi estudada a capacidade de desenvolver uma forma capaz de introduzir uma infraestrutura IEEE1149.4 de forma a validar o funcionamento do sistema configurado. Desta forma o sistema desenvolvido passou por diversas fases.

O trabalho iniciou-se por um estudo aprofundado das normas IEEE1149.1 e 1149.4, para clarificar as capacidades das infraestruturas e de forma a perceber como é realizado o controlo dos dispositivos que implementem estas infraestruturas. Para além do estudo das infraestruturas tornou-se necessário realizar um estudo dos diversos meios de configuração dos dispositivos FPAA's de forma a tornar mais percetível a configuração do dispositivo para que seja possível ter o controlo total do mesmo.

Visto que a utilização das infraestruturas mencionadas anteriormente, são bastante uteis para apoio na depuração desenvolveu-se um sistema capaz de apoiar no domínio académico, de forma a tornar o sistema de controlo da infraestrutura mais próxima dos alunos. Com o sistema desenvolvido, é possível ligar um ou mais dispositivos STA400 ao módulo da NI para que seja possível através do *software* desenvolvido, controlar e observar o estado do controlador, controlar e observar os nós dos dispositivos, assim como definir os próprios procedimentos a desenvolver durante o decorrer do teste. Desta forma o sistema torna-se mais prático, não sendo toda a informação fornecida apenas a nível teórico.

Sendo as FPAA's muito direcionadas ao desenvolvimento de sistemas analógicos, por vezes, nota-se uma grande dificuldade no que toca à configuração dos dispositivos através dos

meios disponíveis apenas com utilização de uma FPAA, sem necessidade de utilização de utilização de um kit de desenvolvimento (onde já inclui um sistema capaz de realizar a programação diretamente através do computador). Atualmente, no que toca a configuração dos dispositivos, existem diversas formas, sendo elas através de utilização de um kit de desenvolvimento com comunicação RS232 (através de uma porta série ou através de um USB), através de uma memória, ou através de um dispositivo externo com comunicação SPI. Para tal, foi desenvolvido um capítulo específico para descrever sucintamente a forma de configuração de uma, ou várias FPAA's utilizando um dispositivo externo com comunicação SPI.

Para dar resposta ao problema inicialmente descrito, desenvolveu-se um circuito capaz de observar e controlar a entrada e saída de uma FPAA, sendo também introduzido um módulo no circuito para permitir a realização da configuração da FPAA através de comunicação SPI. Para além do desenvolvimento do circuito, tornou-se imprescindível ter forma de realizar o controlo de todo o sistema, para tal foi necessário desenvolver código em duas linguagens distintas. Sendo a configuração da FPAA realizada através do módulo com comunicação SPI (Arduino), foi necessário desenvolver código para ser processado no Arduino, com capacidade de implementação de comunicação SPI e com capacidade de comunicação com o *software* desenvolvido para o computador. O *software* de alto nível que é executado no computador foi desenvolvido em Labview, sendo esta a interface que o utilizador irá visualizar quando se encontrar a realizar o teste ao dispositivo. Nesta interface o utilizador terá a capacidade de realizar diversas ações tais como realizar a configuração da FPAA utilizando um ficheiro proveniente do *software* de configuração da Anadigm, controlar e observar os nós da FPAA, sendo estas configuração facilitadas para o utilizador, não sendo necessário realizar a configuração dos STAA400 diretamente, e para finalizar o software permite também que o utilizado possa enviar tramas de configuração para os módulos STA400 para que possa realizar os seus meios de teste.

Para além da capacidade de configuração, observação e controlo tudo a partir de um único *software*, o sistema também está capaz de ser acedido através de uma página *web*, apenas por um utilizador de cada vez, tornando assim o sistema mais abrangente para qualquer utilizador que não possua o *hardware* e necessita de validar o funcionamento de algum circuito a implementar na FPAA.

Em termos funcionais, o sistema desenvolvido está de acordo com os objetivos propostos inicialmente, sendo todos eles validados. De forma a garantir que o sistema fica completamente independente de outros sistemas externos, o ideal seria remover o sistema de programação diretamente por SPI, introduzindo no sistema módulos com interface IEEE1149.1 de forma a implementar a programação do mesmo apenas pelas linhas de TDI, tendo assim a capacidade de implementação de uma infraestrutura única de programação e teste, tal como as atuais FPGA.

Referências Documentais

[1] Transmissão de sinais. Disponível em: http://katzenn.webnode.pt/transmissão-de-sinais/. Acesso em: 28 Out . 2015.

[2] The LM324 Quad Op-Amp Line Follower Robot with Pulse width Modulation. Disponível em: http://www.ermicro.com/blog/?p=1908. Acesso em: 28 Out. 2015.

[3] Yurish, Sergey.
Smart Sensors Systems Design. Disponível em: http://www.ieee-sensors.org/files/2011/archive/Tutorials/Yurish/Yurish_course.pdf. Acesso em: 27 Out. 2015

[4] Stolfi, Guido.
Tópicos sobre estratégias de projecto. Disponível em: www.lcs.poli.usp.br/~gstolfi/PPT/TecProj.pps Acesso em: 02 Nov. 2015

[5] AOI test. Disponível em: http://www.european-business.com/matas_electronics_bv/portrait/ Acesso em: 02 Nov. 2015

[6] Anh-Vu H. Pham,Morgan J. Chen,Kunia Aihara.
LCP for Microwave Packages and Modules, Cambridge. 2012

[7] Poole, Ian.
Ball Grid Array, BGA. Disponível em: http://www.radio-electronics.com/info/data/smt/smd-bga-ball-grid-array-package.php Acesso em: 3 Nov. 2015

[8] X-ray Application: PCB (Printed Circuit Board). Disponível em: http://engsec.deeb.co.kr/eng/application/pcb.php Acesso em: 3 Nov. 2015

[9] Failures Analysis Service. Disponível em: http://www.fujitsu.com/jp/group/fql/en/services/product-quality/failure-analysis/ Acesso em: 3 Nov. 2015

[10] ICT, In Circuit Test. Disponível em: http://www.radio-electronics.com/info/t_and_m/ate/ict-in-circuit-test-tutorial.php Acesso em: 3 Nov. 2015

[11] SPEA responds to the new needs of electronics manufacturers with its Bed of Nails automated test cell. Disponível em: http://www.spea.com/Portals/0/SPEANewsTemplates/Generic.aspx?pItemId=268&pModuleId=845&pViewType=AllPress Acesso em: 3 Nov. 2015

[12] Rcd210 pcb bottom side, test points http://www.lg3gservice.com/pub/martech%20support%20archive/rcdpro/schematics/ Acesso em 18 Nov. 2015

[13] Sonde mobili e letto d'aghi: tecnologie a confronto. Disponível em: http://www.spea.com/Portals/0/SPEANewsTemplates/Generic.aspx?pItemId=89&pModuleId=844&pViewType=AllPress Acesso em: 3. Nov.2015

[14] LED ACDC Burnin Test System. Disponível em: http://www.chromaus.com/product_58266_LED_ACDC_Burnin_Test_System.php. Accesso em: 4. Nov. 2015

[15] Mobile Phone with function of creative poster design. Disponível em: http://www.free-psds.com/2014/10/mobile-phone-with-function-of-creative-poster-design/ Acesso em: 4. Nov. 2015

[16] Analog and Digital. Disponível em: https://learn.sparkfun.com/tutorials/analog-vs-digital Acesso em: 28. Out. 2015

[17] (TEDSE) Boundary Scan Tutorial - ASSET BS Company. 2000

[18] Why Boundary-Scan?. Disponível em: http://www.jtag.com/en/content/why-boundary-scan Acesso em: 9. Nov. 2015

[19] Inspection and Testing. Disponível em: http://www.sparqtron.com/core-services/inspection-and-testing Acesso em: 4. Nov. 2015

[20] Inspection & Test of electronic boards - concepts & strategies. Disponíivel em: http://see-industry.com/industrial-statiieng.aspx?br=21&rub=119&id=340 Acesso em: 3. Nov. 2015

[21] In- Circuit vs. Functional Test http://www.bloomy.com/support/blog/circuit-vs-functional-test Acesso em: 4. Nov. 2015

[22] Functional Testing. Disponível em: http://www.nexlogic.com/pcb-testing/functional-testing/ Acesso em: 4. Nov. 2015

[23] IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993), IEEE Standard Test Access Port and BoundaryScan Architecture.2

[24] (TEDSE) JTAG Tutorial - TI Test Symposium. 1997

[25] IEEE Std 1149.4-1999, IEEE Standard for a Mixed Signal Test Bus

[26] What is programmable logic? Disponível em: http://www.xilinx.com/company/about/programmable.html Acesso em: 4. Jan. 2016

[27] Plataformas FPGA da NI. Disponível em: http://www.ni.com/fpga/pt/ Acesso em: 4. Jan. 2016

[28] Field Programmable Gate Array (FPGA). Disponível em: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm Acesso em: 4. Jan. 2016

[29] Chris Evans-Pighe.
Programmable Analog: More Gain, Less Pain. 2001

[30] David Marsh.
Programmable analogue ICs challenge Spice-and-breadboard designs. 2001

[31] Gloria Huertas. José L. Huertas. Emilio Lora
Very-Large-Scale integration of electronic circuits

[32] Tyson S. Hall
Field-Programmable Analog Arrays: A Floating-Gate Approach. 2014

[33] Sachin Gupta. Using Switched capacitors to create programmavle analog logic blocks in mixed-signal designs. Disponível em: http://www.eetimes.com/document.asp?doc_id=1278236 Acesso em: 12. Jan. 2016

[34] Capacitors and Switches to Resistance. Disponível em: http://www.eeweb.com/electronics-quiz/capacitors-and-switches-to-resistance Acesso em: 12 Jan. 2016

[35] FPAA – Field Programmable Analog Array. Disponível em: http://www.fpgacentral.com/pld-types/fpaa-field-programmable-analog-array Acesso em: 13. Jan. 2016

[36] Christian Birk
Application and Evaluation of FPAA. Disponível em: http://www2.engr.arizona.edu/~cmsl/Publications/FPAA/ChrisTh.PDF Acesso em: 14. Jan. 2016

[37] Ambiente gráfico de desenvolvimento de sistemas LabVIEW. Disponível em: http://www.ni.com/labview/pt/ Acesso em: 27. Jun. 2016

[38] Se há algo que você pode ligar, dirigir ou pilotar, provavelmente a tecnologia da NI e o LabVIEW ajudou a torná-lo possível. Disponível em: http://www.ni.com/pdf/products/pt/22851_LV_Interactive.pdf Acesso em: 27. Jun. 2016

[39] TestStand. Disponível em: http://www.ni.com/teststand/pt/ Acesso em: 27. Jun. 2016

[40] Áreas de Aplicação: Onde o Teststand é usado? Disponível em: http://www.ni.com/teststand/applications/pt/ Acesso em: 27. Jun. 2016

[41] AN231K04-DVLP3 – AnadigmApex Development Board Disponível em: http://www.anadigm.com/_doc/UM231000-K001A.pdf Acesso em: 02. Jul. 2016

[42] AN231E04 dpASP, Primary Configuration timing Diagram. Disponível em: http://www.anadigm.com/_doc/DS231001-U002.pdf Acesso em: 03. Jul. 2016

[43] LMx24-N, LM2902-N Low-Power, Quad-Operation Amplifiers Disponível em: http://www.ti.com.cn/cn/lit/ds/symlink/lm2902-n.pdf Acesso em: 09. Jul. 2016

[44] CircuitMaker: Components - STA400 IEE 1149.4 Disponível em: http://circuitmaker.com/Components/Details/CMP-ba4c73181ec5658b-1 Acesso em: 07. Abr. 2016

[45] AnadigmApex dpASP Family User Manual. Disponível em: http://www.anadigm.com/_doc/UM000231-U001.pdf Acesso em: 02. Jul. 2016

[46] AN-1200 Mixed Signal Testing Using the IEEE 1149.4 STA400 Disponível em: http://www.ti.com/lit/an/snla052/snla052.pdf. Acesso em: 20. Mai. 2016

[47] STA400EP. Enhanced Plastic Dual 2:1 Analog Mux with IEEE 1149.4 Disponível em: http://www.farnell.com/datasheets/51708.pdf Acesso em: 21. Mai. 2016

[48] Datalogging with SD Card – Serial Peripheral Interface (SPI) Disponível em: http://akibarduino.blogspot.pt/2015/08/datalogging-with-sd-card-writing-data.html Acesso em: 17. Set. 2016