Faculdade de Engenharia da Universidade do Porto


# Rede Acústica Reconfigurável

Ivo Alcântara Tavares Dias

Dissertacão realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major em Automação

Orientador: Prof. Dr. Aníbal Castilho Coimbra de Matos

Junho de 2008

# Resumo

O aumento das aplicações que necessitam da realização de trabalhos submersos, desde recolha de dados para investigação científica até à fixação de plataformas flutuantes para os mais diversos fins, e o avanço da tecnologia criaram condições para o uso de veículos autónomos subaquáticos para a realização deste tipo de trabalhos. Uma necessidade resultante do uso deste tipo de veículos é a determinação da sua localização.

Habitualmente, esta é feita aproveitando as boas propriedades de propagação das ondas acústicas em meios subaquáticos utilizando transdutores, que funcionam como emissor e receptor. Um deles é colocado no veículo subaquático autónomo (VSA) e os outros utilizados como faróis acústicos de modo a formarem as *baselines* necessárias à navegação do VSA. Basicamente o ciclo de detecção consiste na emissão de um sinal acústico com uma frequência distinta para cada farol, enviado pelo VSA, e após a detecção desse sinal pelo respectivo farol este responde através da emissão de um sinal acústico com uma frequência pré-definida. O transdutor colocado no VSA detecta a resposta do farol e o posicionamento do VSA relativamente ao farol é estimado dependo do tempo de resposta do farol. Após o VSA estimar a distância a todos os faróis o VSA calcula a sua localização dentro do referencial fixo definido pelo conjunto dos faróis.

Esta dissertação tem como objectivo desenvolver algoritmos para a medição do instante de detecção de uma onda acústica e implementá-los recorrendo à tecnologia de *hardware* reconfigurável baseada em *field programmable gate array* (FPGA). Estas medições são utilizadas para estimar a direcção de uma onda acústica recorrendo à técnica de posicionamento acústico *ultra short baseline* (USBL). Esta técnica permite obter o ângulo estimado da direcção de uma onda acústica a partir da diferença de tempo entre os instantes de detecção de uma onda acústica. Esta técnica utiliza os transdutores muito próximos (algumas dezenas de centímetros) e a sua precisão é proporcional à frequência de amostragem utilizada.

# Abstract

The incrising number of applications that need the realization of underwater works, such as the gathering of data to scientific research and the location fixing of floating platforms for many different purposes and also the advance of technology, made conditions possible for the use of autonomous underwater vehicles (AUVs) for the realization of this kind of works. A need resulting from the use of this kind of vehicles is the determination of it's location. Usually, the location is estimated taking advantage of the good propagation properties of acoustic waves in water, through the use of acoustic transducers, which work as emitters and receptors. One of these transducers is applied in the AUV and the others form the baselines needed to the navigation of the AUV.

Basically the detection cycle consists in emitting an acoustic signal at a given frequency for each acoustic beacon, sent by the AUV, and after the detection of the acoustic signal by the respective beacon this responds emitting an acoustic signal with a predefined frequency. The AUV transducer detects the response of the beacon and estimates it's position relatively to this beacon by measuring the time of arrival of the acoustic signal sent by the beacon. After estimating the distances to all the acoustic beacons, calculates it's own location within the fixed frame defined by the whole of the acoustic beacons.

The prime objective of this dissertation it's to develop algorithms for measuring the instant of detection of an acoustic signal and implement them through the use of reconfigurable hardware technology based in field programmable gate array (FPGA). This measures will be used for estimating the direction of an acustic signal using the acoustic positioning technique ultra short baseline (USBL). This technique allows obtaining the estimated direction angle of an acoustic signal from the time difference between detection instants of an acoustic signal. This technique has it's transducers very close (few tens of centimeters) and it's precision is proporcional to the used sampling frequency.

# Agradecimentos

Queria aproveitar este espaço para deixar uma palavra de apreço ao Prof. Aníbal Castilho Coimbra de Matos pela excelente orientação fornecida e pelas valiosas observações e comentários feitos.

Gostaria de agradecer ao Prof. José Carlos Alves por me ter disponibilizado o acesso aos recursos computacionais do laboratório I223 e a algum material necessário para o desenvolvimento deste trabalho.

Agradeço também ao colega Sérgio Rui Silva do laboratório I222 pelo auxílio prestado em momentos de maior dificuldade e a todos os que contribuíram para que de alguma forma este trabalho se realizasse.

# Índice

# Lista de figuras

# Lista de tabelas

# Abreviaturas e símbolos

Lista de abreviaturas (ordenadas por ordem alfabetica)

| | |
|---|---|
| AUV | *Autonomous Underwater Vehicle* |
| A/D | Conversor Analógico/Digital |
| ASIC | *Application-Specific Integrated Circuit* |
| CAD | Conversor Analógico/Digital |
| CDA | Conversor Digital/Analógico |
| CPU | *Central Processing Unit* |
| D/A | Conversor Digital/Analógico |
| DD | *Device Driver* |
| DSP | *Digital Signal Processing* |
| E/S | Entradas/Saídas |
| EHF | *Extra High Frequency* |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| FET | F*ield-effect Transistor* |
| FFT | *Fast Fourier Transform* |
| FIFO | *First In First Out* |
| FIR | *Finite Impulse Response* |
| FPGA | *Field Programmable Gate Array* |
| GPS | *Global Positioning System* |
| GNU | *GNU's Not Unix* |
| HF | *High Frequency* |
| IAPSO | *International Association for the Physical Sciences of the Ocean* |
| IP | *Intelectual Property* |
| IIR | *Infinite Impulse Response* |
| ISA | *Industry Standard Architecture* |
| LBL | *Long Baseline* |
| LF | *Low Frequency* |
| MF | *Medium Frequency* |
| PC | *Personnel Computer* |
| PCI | *Peripheral Component Interconnect* |
| PCB | *Printed Circuit Board* |
| PDS | Processamento Digital de Sinal |
| PSS | *Practical Salinity Scale* |
| ROV | *Remotely Operated Vehicle* |
| SBS | *Short Baseline* |
| UDP | *User Datagram Protocol* |
| USBL | *Ultra Short Baseline* |
| VHF | *Very High Frequency* |

| | |
|---|---|
| VLSI | *Very Large Scale Integrated* |
| VSA | Veículo Subaquático Autónomo |
| µC | Micro controlador |

Lista de simbolos

| | |
|---|---|
| ºC | Graus Célcius |
| dB | Decibel |
| km | Kilometro |
| m | Metro |
| m/s | Metro por segundo |
| µs | Microsegundo |
| ms | Milissegundo |

# Capítulo 1

# Introdução

Esta dissertação foi realizada no laboratório de sistemas oceanográficos *(Oceansys)*, onde decorre o desenvolvimento de um VSA (Veículo Subaquático Autónomo) cujo objectivo final é a recolha de dados respeitantes à qualidade da água, e no laboratório I222 ambos da Faculadade de Engenharia da Universidade do Porto (FEUP). O trabalho realizado enquadra-se nas actividades de investigação e desenvolvimento de sistemas acústicos para posicionamento subaquático e consistiu no estudo e implementação de um sistema de detecção de sinais acústicos baseado em hardware reconfigurável.

## 1.1 - Motivação

O aumento das aplicações que necessitam da realização de trabalhos submersos, desde recolha de dados para investigação científica até fixação de plataformas flutuantes para os mais diversos fins, e o avanço da tecnologia criou condições para o uso de veículos subaquáticos autónomos para a realização deste tipo de trabalhos. Uma necessidade resultante do uso deste tipo de veículos é a determinação da sua localização. Habitualmente, esta é feita aproveitando as boas propriedades de propagação das ondas acústicas em meios subaquáticos.

## 1.2 - Objectivos

Esta dissertação tem como objectivo desenvolver algoritmos para a medição do instante de detecção de uma onda acústica e implementá-los recorrendo à tecnologia de *hardware* reconfigurável baseada em *field programmable gate array* (FPGAs). Neste trabalho optou-se pela utilização de tecnologia de *hardware* reconfigurável baseada em

FPGA para a implementação de algoritmos de processamento digital de sinal, como por exemplo *matched filters,* pois esta tecnologia é altamente paralelizável e reconfigurável. Estas características das FPGAs permitem a execução de vários algoritmos complexos de processamento digital de sinal simultaneamente e modificar facilmente as suas configurações tais como frequência de amostragem, coeficientes dos *matched filters,* resolução utilizada pelos filtros, etc.

Tipicamente as técnicas de posicionamento acústico são baseadas nas diferenças de tempo entre detecções, através dos seus transdutores ou hidrofones, de sinais acústicos para estimar o posicionamento de um VSA, por exemplo. Várias técnicas existem sendo as mais comuns a *long base line* (LBL), *short baseline* (SBL) e *ultrashort baseline* (USBL). Estas técnicas devem o seu nome devido ao tamanho das *baselines*, isto é, a distância entre transdutores.

Os instantes medidos são utilizadas para estimar a direcção de uma onda acústica recorrendo à técnica de posicionamento acústico *ultra short baseline* (USBL). Esta técnica permite obter o ângulo estimado da direcção de uma onda acústica a partir da diferença de tempo entre os instantes de detecção de uma onda acústica. Esta técnica utiliza os transdutores muito próximos (algumas dezenas de centímetros) e a sua precisão é proporcional à frequência de amostragem utilizada.

## 1.3 – Estrutura

Este relatório está dividido em oito capítulos. No primeiro capítulo faz-se uma introdução genérica do problema e descreve-se a motivação e os objectivos para esta dissertação.

No segundo capítulo são abordados aspectos gerais sobre acústica subaquática, tais como domínios de aplicação, propagação do som na água, etc. São também abordadas algumas das técnicas de posicionamento acústico existentes e as suas principais características.

No terceiro capítulo são abordados alguns dos aspectos que envolvem o processamento digital de sinal, de modo a ilustrar de um modo simples uma panorâmica geral acerca do processamento digital de sinal. Também será descrita alguma teoria acerca de algumas operações de processamento digital de sinal que serão utilizadas como base nas questões relacionadas com o processamento digtal de sinal no desenvolvimento da dissertação.

No quarto capítulo são abordadas algumas das características mais importantes das FPGAs tais como, a sua estrutura, aplicacações, *tradeoffs* que se têm de fazer no desenvolvimento de um sistema baseado em FPGAs, etc. Também serão abordadas algumas das vantagens e desvantagens das FPGAs.

No quinto capítulo são abordados alguns aspectos relacionados com linguagens de descrição de *hardware* bem como algumas das ferramentas de projecto existentes que permitem a utilização dessas linguagens de descrição de *hardware* no desenvolvimento de projectos em FPGAs.

No sexto capítulo são abordados todos os passos executados no desenvolvimento da dissertação desde a abordagem ao problema, escolha de *hardware* e de ferramentas de projecto, opções tomadas ao longo da dissertação bem como uma descrição do sistema a desenvolver.

No sétimo capítulo são apresentadas algumas das simulações executadas no decorrer da implementação do sistema. Estas simulações foram feitas recorrendo à ferramenta de projecto MATLAB numa primeira fase e depois recorrendo à ferramenta de projecto *Simulink* já utilizando blocos da ferramenta System Generator.

No oitavo e último capítulo são apresentadas as conclusões e o trabalho futuro que poderá ser feito para melhorar a globalidade do sistema.

# *Capítulo 2*

# Acústica Subaquática

Neste capítulo serão abordados aspectos gerais sobre acústica subaquática, tais como domínios de aplicação, propagação do som na água, etc. Serão também abordadas as técnicas de posicionamento acústico existentes e as suas principais características.

## 2.1 - Introdução

A acústica subaquática é utilizada em diferentes domínios, tais como "cartografar" as características do fundo do mar, comunicar informação através do meio aquático ou medição de propriedades da água.

Na acústica subaquática relacionam-se as propriedades da água com o comportamento da propagação, ruído e reflexões das ondas acústicas. As propriedades físicas e químicas da água e as características geológicas e biológicas da zona onde vão ser usados os métodos acústicos têm muita importância pois influenciam a propagação das ondas acústicas.

A variável acústica mais importante é a velocidade do som na água. A distribuição da velocidade do som debaixo de água influencia todos os outros fenómenos acústicos. A velocidade do som depende da salinidade, temperatura da água e pressão. O fluxo horizontal da velocidade do som é também afectado pelas correntes de água. A refracção do som nas várias características dinâmicas existentes debaixo de água tais como a composição e topografia do fundo que pode bloquear a propagação do som e organismos biológicos contribuem para o aumento do ruído e também espalham os sinais acústicos em diferentes direcções [1].

O som é uma onda cujo comportamento é comparável ao comportamento da luz. A mesma coisa que acontece na água com as ondas sonoras acontece com um feixe de luz de uma lanterna, por exemplo. A maior parte das vezes a luz de uma lanterna cria um círculo nos objectos aos quais está apontada sem se ver o feixe de luz.

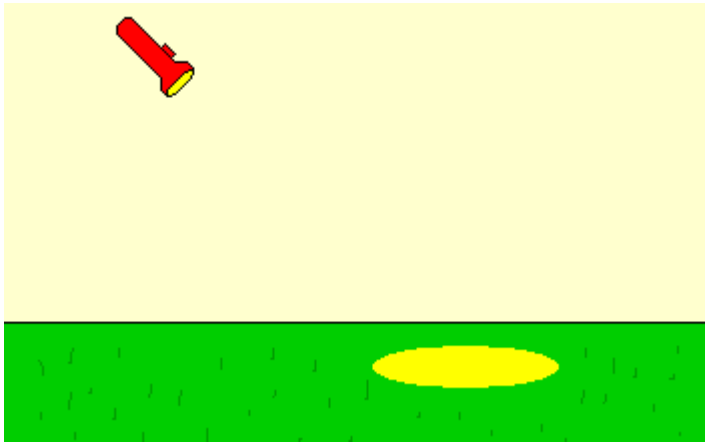


**Figura 2.2.1 - Exemplo propagação da luz 1**

Mas se existir nevoeiro apenas se vê o feixe de luz da lanterna e não o objecto ao qual está apontada a lanterna. Isto acontece porque o ar contem muitas moléculas de água que quando expostas à luz a espalham em todas as direcções, tornando o feixe de luz visível.

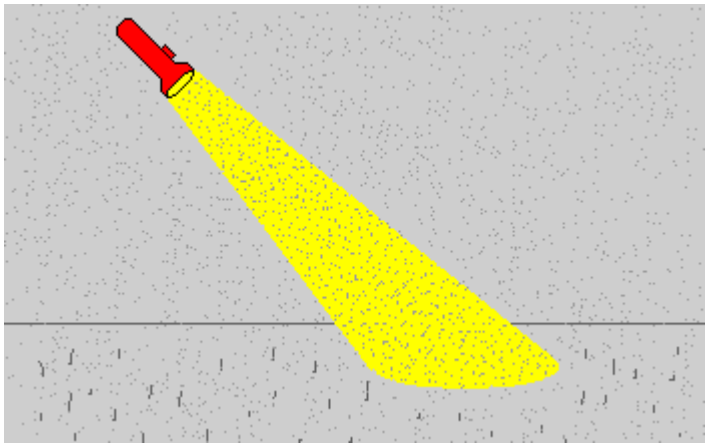


**Figura 2.2.2 - Exemplo propagação da luz 2**

A mesma coisa acontece na água com as ondas sonoras. A quantidade de espalhamento é afectada pelo tamanho dos objectos espalhados pelo meio aquático, que podem ser bolhas de ar, partículas suspensas, organismos, etc, e o comprimento de onda da onda sonora transmitida. Um objecto irá espalhar consideravelmente o som se o seu tamanho for comparável com o comprimento de onda do som. Pode-se calcular o comprimento de onda do som na água dividindo a velocidade do som na água, aproximadamente 1500 m/s, pela frequência do som. Por exemplo, uma onda sonora de 200 Hz tem um comprimento de onda de 7.5 metros. Se um objecto for menor do que o

comprimento de onda do som, este tende a deslocar-se à volta do objecto que se encontre no seu caminho e não é significativamente afectado pelo objecto. Daí o uso de ondas sonoras de frequência elevada na procura de objectos pequenos no oceano, tal como peixes [2].

## 2.2 - Propriedades físicas e químicas

A temperatura é a base para qualquer descrição física do meio aquático. É a propriedade mais fácil de medir, logo a mais comum de ser registada através medidas do meio aquático. A velocidade do som nas camadas superiores das grandes massas de água é bastante dependente da temperatura. A temperatura influência o tipo e a velocidade das reacções químicas que ocorrem na água, afectando assim a velocidade de propagação.

A água é um fluído constituído por um número variado de sais minerais. A presença destes sais afecta um número variado de propriedades da água incluindo compressibilidade, velocidade do som, expansão térmica, temperatura de congelamento, etc. [1]. Salinidade é o termo usado para medir a quantidade de sais dissolvidos na água e é expressa em partes por mil (‰ ou ppm). A definição precisa de salinidade é bastante complexa. Nicholas Paul Fofonoff reviu o desenvolvimento da escala prática de salinidade (PSS - practical salinity scale) e a equação de estado da água do mar [3]. A escala prática da salinidade redefinida pelos oceanógrafos em 1978 definiu a salinidade como uma relação salinidade-condutividade. Esta nova escala foi introduzida para rectificar as deficiências associadas à relação cloro-salinidade usada anteriormente para definir salinidade.

O standard prático actual é "IAPSO Standard Seawater", define a salinidade como uma grandeza sem dimensões pois os algoritmos nesta nova escala foram ajustados para eliminar a ‰ (ppm - partes por mil) usada em escalas anteriores.

A densidade da água relaciona-se com a temperatura, salinidade e pressão (que é quase proporcional à profundidade) através de equações de estado [3]. A densidade providencia uma medida da estabilidade hidrostática no oceano. Especificamente, uma coluna de água estável é uma coluna cuja densidade aumenta monotonamente com o aumento da profundidade.

A água do mar é compressível, embora menos do que a água doce. A compressibilidade da água do mar pode ser pode ser expressa por um coeficiente de compressibilidade, que relaciona mudanças num dado volume de água com a correspondente mudança de pressão [4].

A compressibilidade da água é um factor importante em várias aplicações: na determinação precisa da densidade da água, particularmente a grandes profundidades; na computação de variações na temperatura adiabática (num processo adiabático uma compressão resulta em aquecimento enquanto que uma expansão resulta em arrefecimento); e, o mais importante, na computação da velocidade do som na água numa dada profundidade [1].

## 2.3 - Velocidade do som

As variações da velocidade do som na água são relativamente pequenas. Tipicamente a velocidade do som varia entre 1450 e 1550 m/s [5]. Mesmo assim, pequenas variações da velocidade do som afectam significativamente a propagação do som na água. A velocidade da água pode ser medida directamente usando instrumentos especiais, como por exemplo velocímetros, ou calculada empiricamente se temperatura (T), salinidade (S) e pressão (P) ou profundidade (z) forem conhecidas. O erro de medida típico dos velocímetros actuais é de 0.1 m/s. A precisão dos cálculos pelas fórmulas empíricas mais completas é basicamente a mesma. No entanto as fórmulas com que se pode obter esse tipo de precisão são muito difíceis de calcular devido ao seu tamanho. Uma equação menos precisa mas mais simples é:

$$c = 1449.2 + 4.6T + 0.055T^2 + 0.00029T^3 + (1.34 - 0.1T)(S - 35) + 0.016z \qquad (2.3)$$

onde a temperatura é expressa em graus célsius (ºC), a salinidade em partes por mil (ppm ou ‰) e a profundidade em metros (m) e a velocidade do som (c) em metros por segundo (m/s). A equação 2.3.1 é apenas válida se:

- 0 ºC ≤ T ≤35 ºC;
- 0 ‰ ≤ S ≤ 45 ‰;
- 0 m ≤ z ≤ 1000 m.

Pode verificar-se que a velocidade aumenta com a temperatura, salinidade e pressão. De acordo com a equação 2.3.1 se considerarmos a temperatura e salinidade constantes a velocidade do som aumenta com o aumento da pressão causada pelo aumento da profundidade. Na maior parte do oceano o gradiente vertical da velocidade do som é 1000 vezes maior do que o horizontal excepto em áreas do oceano onde exista convergência de correntes, tanto frias como quentes, onde estes gradientes podem ser comparados. Daí poder-se considerar numa primeira aproximação que o oceano é um meio estratificado horizontalmente, isto é, as propriedades da água apenas variam com a profundidade mantendo-se constantes no plano horizontal [5].

O perfil da curva *c(z)* e a distribuição do gradiente da velocidade do som com a profundidade são bastante mais importantes para compreendermos a propagação do som na água do que o valor absoluto da velocidade. Para um tipo de perfil *c(z)*, o som pode-se propagar por centenas ou milhares de kilometros, e para outro tipo de perfil, para a mesma frequência de som, este apenas se propaga algumas dezenas de metros ou até menos. A figura seguinte é um exemplo de um perfil típico da velocidade do som debaixo de água para uma zona de oceano “aberto” em latitudes médias. Latitudes estas, que ocorrem entre os paralelos 30 e 50º N e também na mesma faixa no hemisfério sul.

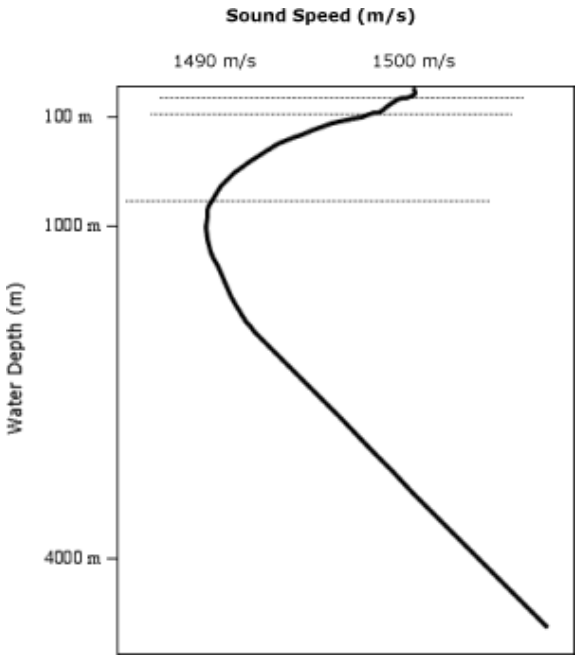



**Figura 2.3.1 - Perfil típico da velocidade do som no oceano "aberto" para latitudes médias**

Estes perfis variam conforme as regiões do oceano e variam também no tempo. As maiores flutuações destes perfis ocorrem nas camadas superiores do oceano devido às variações tanto sazonais e diárias da temperatura e salinidade. A profundidades superiores a 1 km, variações verticais da temperatura e da salinidade são tipicamente pequenas e o aumento da velocidade do som com a profundidade depende quase exclusivamente do aumento da profundidade. Como consequência das grandes profundidades a velocidade do som aumenta linearmente com o aumento de profundidade.

## 2.4 – Posicionamento acústico

Os sistemas acústicos são usados principalmente para detecção precisa de posicionamento e navegação de embarcações de superfície e para instrumentos e veículos subaquáticos. São usados em: estudos oceanográficos; prospecção geofísica; exploração subaquática de gás e óleo; posicionamento dinâmico de embarcações; vigilância subaquática; teste e avaliação da performance de sistemas militares, tais como sonares, torpedos, precisão e alcance de mísseis [6].

O princípio de funcionamento destes sistemas consiste na medição do tempo de chegada, da fase ou do desfasamento de Doppler dos sinais acústicos transmitidos entre pontos de referência. Algumas alternativas têm sido desenvolvidas. Uma delas usa a geometria de uma pirâmide, cuja base se encontra no leito do oceano ou rio (Long Baseline - LBL) onde pelo menos três dos quatro vértices da base têm um farol acústico. O vértice mais alto encontra-se acoplado ao casco de uma embarcação à superfície. Outra alternativa usa uma pirâmide invertida cujo vértice está no leito do oceano ou rio e a base está acoplada ao casco de uma embarcação à superfície (Short and Ultra Short Baseline - SBL e USBL respectivamente). Estas diferentes técnicas de posicionamento acústico, podem ser utilizadas simultaneamente (Long Ultra Short Baseline, por exemplo).

## 2.5 - Long Baseline

As técnicas de posicionamento acústico do tipo LBL podem-se dividir em dois segmentos. O primeiro inclui um certo número de faróis acústicos ancorados no leito. A posição destes faróis é descrita num sistema de coordenadas fixo no leito. A distância entre os faróis acústicos forma as *baselines* utilizadas pelo sistema.

O segundo segmento inclui um transdutor[1] acoplado a uma embarcação, ou a outro objecto que se queira conhecer a posição, como no exemplo da figura 2.5.1. A distância do transdutor ao farol pode ser estimado fazendo com que o transdutor emita um sinal acústico de curta duração que o farol detecta e responde com outro sinal acústico. O tempo que decorre entre a emissão do primeiro sinal e a detecção do segundo é medido. Como a velocidade do som na água é conhecido, a distância entre o transdutor e o farol pode ser estimada. O procedimento é repetido para os restantes faróis e a posição da embarcação relativamente aos faróis pode ser então calculada ou estimada.

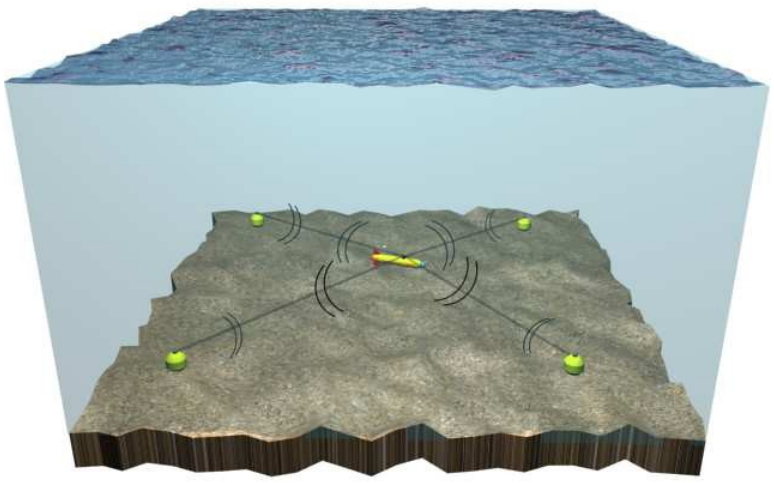


**Figura 2.5.1 - *Long Baseline* (LBL)**

A navegação pode ser conseguida à custa de apenas dois faróis ancorados no leito mas, neste caso, existe a possibilidade de ambiguidade em relação a de que lado da *baseline* a embarcação está. Três bóias acústicas são o mínimo necessário para uma navegação sem ambiguidades em três dimensões sendo que quatro são o mínimo necessário para existir alguma redundância, o que tem utilidade alguma informação acerca da qualidade da navegação.

Esta técnica implica o uso de um sistema de calibração ou auto-calibração dos faróis acústicos usados no leito do oceano e para tal existem várias maneiras de a fazer. A mais apropriada depende da tarefa ou aplicação em causa e do *hardware* disponível. A

1 - Parte de um sistema de posicionamento acústico que funciona como uma antena, enviando e recebendo sinais acústicos. O transdutor converte sinais eléctricos em sinais acústicos e vice-versa.

utilização de *transponders*[1] nos faróis acústicos permite medir directamente as *baselines*, distância entre faróis, e em sistemas de navegação por satélite, tal como GPS, a calibração dos faróis acústicos torna-se bastante rápida e simples [7].

Em operação normal, os faróis são interrogados com sinais de diferentes frequências, dependendo da profundidade da água (geralmente cerca de 10 kHz e entre 50-100 kHz para aplicações de alta frequência). O tempo útil de um farol acústico varia entre um mês e um ano, dependendo do modo de operação (cadência de transmissão) e da capacidade das baterias acopladas ao farol.

A flutuabilidade de um farol colocado no leito é ligeiramente positiva para que, quando acoplado a uma âncora permaneça alguns metros acima do leito, evitando assim a existência de múltiplas reflexões acústicas.

O termo *Long Baseline* é usado porque geralmente o comprimento da *baseline* é muito maior para a técnica LBL do que para a técnica SBL e certamente maior do que USBL! A técnica de posicionamento LBL é muito precisa comparada com a SBL ou USBL e tem a vantagem de posicionar a embarcação (ou outro qualquer objecto) directamente num sistema de coordenadas fixo ou inercial. Esta característica resolve a maior parte dos problemas associados com o movimento da embarcação, que existe nas outras técnicas.

As técnicas de posicionamento acústico LBL são usadas extensivamente em aplicações, tais como operações de perfuração, em zonas de grande profundidade, tipicamente em profundidades superiores a 1000 m e utilizando frequências entre 8-15 kHz.

## 2.6 - Short Baseline

A técnica de posicionamento acústico SBL é normalmente aplicada a embarcações de tamanho médio/grande, semi-submersíveis ou grandes embarcações de perfuração. Tipicamente são usados pelo menos três, mas normalmente quatro, transdutores acústicos dispostos de forma triangular, ou quadrada, acoplados ao casco da embarcação. As distâncias entre transdutores, denominadas por *baselines*, tendem a ser o maior possível, tipicamente no mínimo 10 m de comprimento.

1 - Dispositivo que automaticamente emite um sinal acústico após a recepção de um sinal acústico com frequência pré-definida. *Transponders* são usados para marcar ou seguir objectos debaixo de água.

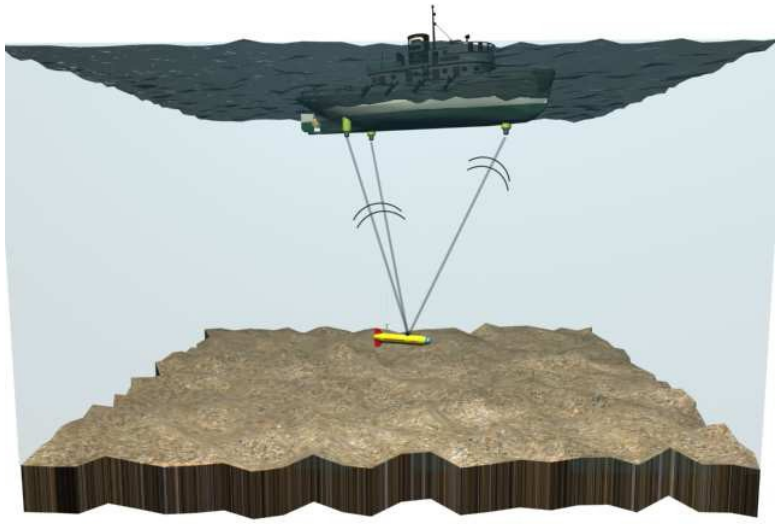


**Figura 2.6.1 - *Short Baseline* (SBL)**

Esta técnica pode também ser aplicada usando faróis, contendo os transdutores submersos, distribuídas em triângulo ou quadrado presas ao leito do rio ou oceano através de âncoras, caso a embarcação existente não ofereça as condições mínimas para obter medições de distância coerentes usando a técnica SBL.

A posição de cada transdutor num sistema de coordenadas fixo à embarcação é determinada através de técnicas de determinação de posicionamento convencionais ou pelo estudo da localização dos transdutores na embarcação.

Na técnica de posicionamento SBL, apenas um transdutor transmite mas todos os transdutores recebem o sinal. O resultado é uma distância medida e um número de tempos de recepção igual ao número de transdutores colocados na embarcação. Se as distâncias dos transdutores ao farol acústico forem medidas como o descrito para a técnica de posicionamento LBL, então a posição do farol acústico no sistema de coordenadas da embarcação, pode ser computada. Adicionalmente, se forem feitas medições redundantes, uma melhor estimativa pode ser determinada, sendo esta, estatisticamente, mais precisa do que a posição calculada pela forma mais básica.

Esta técnica de posicionamento acústico tem a desvantagem de necessitar de equipamento extra de sensorização de modo a compensar os movimentos a que a embarcação está sujeita, tais como inclinómetros para compensar o *roll* e *pitch e* uma bússola ou giroscópio para compensar a direcção (*yaw*). As coordenadas da embarcação podem então ser transformadas matematicamente de modo a remover os efeitos desses movimentos de rotação.

Se o objectivo for estimar a posição da embarcação num referencial fixo ou inercial, tal como o referencial fixo ao leito do oceano, então pelo menos um farol acústico deverá ser colocado numa posição fixa no leito do oceano e usado como ponto de referência.

## 2.7 - Ultra Short Baseline

A necessidade de instalar quatro hidrofones no casco de uma embarcação é uma desvantagem considerável. Daí a necessidade de se desenvolver uma técnica de posicionamento acústico mais simples do que a SBL, surgindo assim a USBL. Nesta técnica de posicionamento o transdutor e o hidrofone estão colocados no mesmo suporte cilíndrico vertical. Um *transponder* colocado no leito do oceano ou rio continua a ser necessário.

O princípio de medição difere do usado nas técnicas LBL e SBL (medições feitas através de impulsos). Na técnica de USBL as recepções de tempo são determinadas a partir das variações de fase do sinal recebido (cerca de 30 kHz) [7]. Estes factores implicam uma menor precisão na determinação do posicionamento do que a técnica SBL. No entanto, a precisão da técnica USBL continua, em situações favoráveis, em cerca de 1%. Em situações onde existam muitas reflexões dos sinais acústicos a precisão degrada-se bastante.

As técnicas de posicionamento acústico USBL são usadas em aplicações de navegação de pouco alcance. Tipicamente operam com frequências de 100 kHz e tem um alcance de 100-500 m [11].

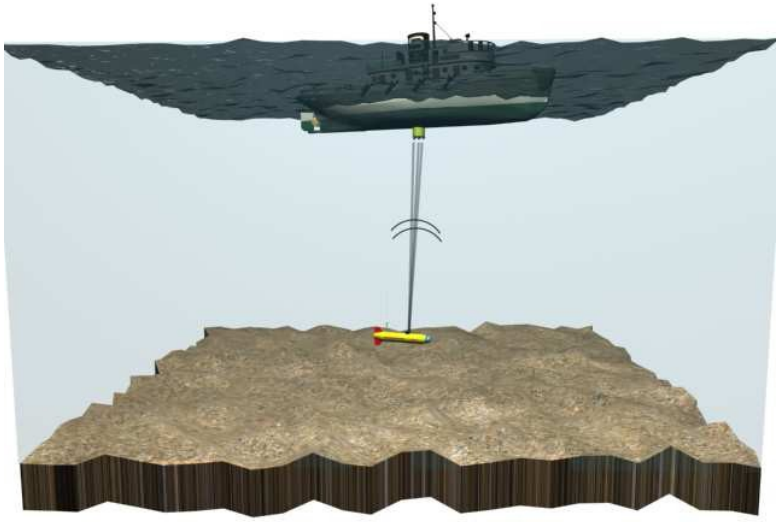


**Figura 2.7.1 – *Ultra Short Baseline* (USBL)**

O princípio de funcionamento do USBL é muito similar ao do SBL, onde um conjunto de transdutores acústicos é colocado na estrutura de uma embarcação, estando neste caso particular os transdutores construídos num único dispositivo acústico, isto é, o conjunto de transdutores separados fisicamente é substituído por um conjunto de transdutores embutidos num único dispositivo acústico.

As distâncias são medidas como na técnica de posicionamento SBL mas as diferenças de tempo usadas para calcular distâncias são muito menores. Se estiverem a

ser usados sinais sinusoidais mede-se o atraso de fase do sinal em cada transdutor em relação a uma referência no receptor. As diferenças de fase entre os transdutores são calculadas por subtracção, obtendo-se assim a direcção do sinal acústico [7]. Alternativamente, a correlação entre os sinais recebidos pelos diferentes transdutores pode também ser usada para determinar as diferenças entre os tempos de detecção.

Os erros de posicionamento são, em teoria, maiores que para os outros métodos devido ao reduzido comprimento das linhas de *baseline* e à influência dos movimentos de rotação *roll* e *pitch* do veículo. Por outro lado, dado que a posição do veículo é calculada com base na orientação da onda acústica recebida, este sistema fica sujeito a forte degradação em ambientes onde exista muita reflexão dos sinais acústicos. A interpretação de sinais reflectidos dá origem a erros graves, já que a orientação do sinal processado, pode ser bastante diferente daquela que une o veículo ao farol acústico. Em aplicações que envolvam grandes distâncias este problema penaliza muito mais a medida de direcção do que a medida de distância, embora esta seja também afectada. O problema das multi-reflexões é particularmente notório em ambientes de baixa profundidade [8].

## 2.8 - Panorâmica Geral

A distância entre faróis acústicos (*baseline*) é geralmente usada para definir o tipo de técnica de posicionamento acústico utilizada. A tabela seguinte mostra as distâncias típicas das *baselines* para algumas das técnicas de posicionamento acústico existentes.

| **Técnica de posicionamento** | **Comprimento da *Baseline*** |
|---|---|
| Ultra Short Baseline | < 10 m |
| Short Baseline | 20 a 50 m |
| Long Baseline | 100 a 6000 m |

**Tabela 2.8.1 – Comprimentos típicos das *baselines***

As técnicas de posicionamento acústico podem ser usadas com bandas de frequências distintas dependendo da finalidade da aplicação para a qual estão a ser utilizadas. A tabela seguinte mostra as bandas de frequência e alcances máximos respectivos para cada gama de frequências utilizadas. As gamas de frequência foram definidas como exemplificado nas tabelas mesmo existindo sobreposição entre gamas de frequências consecutivas [9].

| | **Frequência** | **Alcance Máximo** |
|---|---|---|
| Frequências Baixas (LF) | 8 a 16 kHz | > 10 km |
| Frequências Médias (MF) | 18 a 36 kHz | 2 a 3.5 km |
| Frequências Altas (HF) | 30 a 64 kHz | 1.5 km |
| Frequências Extra Altas (EHF) | 50 a 110 kHz | < 1000 m |
| Frequências Muito Altas (VHF) | 200 a 300 kHz | < 100 m |

**Tabela 2.8.2 - Alcances máximos das gamas de frequências**

Para cada gama de frequências utilizadas obtêm-se diferentes gamas de precisão nas medidas obtidas. A tabela seguinte mostra as gamas de precisão típicas que se podem obter para diferentes gamas de frequências utilizadas.

| | **Frequência** | **Precisão típica** |
|---|---|---|
| Frequências Baixas (LF) | 8 a 16 kHz | 2 a 5 m |
| Frequências Médias (MF) | 18 a 36 kHz | 0.25 a 1 m |
| Frequências Altas (HF) | 30 a 64 kHz | 0.15 a 0.25 cm |
| Frequências Extra Altas (EHF) | 50 a 110 kHz | < 0.05 m |
| Frequências Muito Altas (VHF) | 200 a 300 kHz | < 0.01 m |

**Tabela 2.8.3 - Precisões possíveis para cada gama de frequências**

Para cada técnica de posicionamento acústico existem vantagens e desvantagens. Algumas das mais importantes vantagens e desvantagens são enumeradas na próxima tabela [9].

| | **Vantagens** | **Desvantagens** |
|---|---|---|
| **LBL** | Muito boa precisão independentemente da profundidade. | Sistema complexo que necessita de operadores experientes. |
| | Redundância na observação. | Necessita de maior número de faróis acústicos. |
| | Pode providenciar boa precisão em áreas de operação grandes. | Necessita de maior quantidade de tempo para implementação e recuperação do equipamento. |
| | Simples e de dimensões reduzidas, utilizando apenas um transdutor acoplado à embarcação. | As técnicas necessitam de uma calibração rigorosa em cada implementação do equipamento |
| | | |
| **SBL** | A simplicidade da técnica faz com que o SBL seja uma ferramenta fácil de usar. | Necessita de grandes *baselines* para se obter uma boa precisão. |
| | Boa precisão obtida através do tempo de resposta. | Necessita de uma muito boa calibração inicial, dos transdutores aplicados na embarcação. |
| | Apenas é necessário colocar um farol acústico no fundo do oceano. | Necessita de pelo menos 3 *transceivers* implementados na embarcação. |
| | Redundância na observação. | Precisão de posicionamento absoluta, necessita de sensorização extra (bússolas e inclinómetros). |
| | | |
| **USBL** | A simplicidade da técnica faz com que o USBL seja uma ferramenta fácil de usar. | Precisão de posicionamento absoluta, necessita de sensorização extra (bússolas e inclinómetros). |
| | Boa precisão obtida através do tempo de resposta. | Necessita de uma muito boa calibração inicial, dos transdutores aplicados na embarcação. |
| | Apenas é necessário colocar um farol acústico no fundo do oceano. | Redundância na observação não garantida. |
| | Necessita apenas de um transdutor à superfície. | |

**Tabela 2.8.4 - Vantagens e desvantagens das diferentes técnicas de posicionamento**

Se combinarmos algumas destas técnicas poderemos usar os benefícios das técnicas acima descritos e obter medidas de posicionamento bastante fiáveis e redundantes, advindo daí técnicas de posicionamento acústico bastante mais complexas. Estas técnicas combinadas existem em grande variedade:

- *Long and Ultra Short Baseline (*L/USBL*);*
- *Long and Short Baseline (*L/SBL*);*
- *Short and Ultra Short Baseline (*S/USBL*);*
- *Long, Short, Ultrashort Baseline (*L/S/USBL*).*

No entanto, estas técnicas exigem equipamentos muito mais complexos e apenas se justificam em algumas aplicações. Um exemplo de uma aplicação da técnica L/USBL é em ambientes onde existem várias embarcações posicionadas dinamicamente (*DP vessels*), que necessitam de operar dentro dos limites de interferência acústica uns dos outros [10].

# Capítulo 3

# Processamento digital de sinal

Neste capítulo serão abordados alguns aspectos que envolvem o processamento digital de sinal, de modo a ilustrar de um modo simples uma panorâmica geral acerca do processamento digital de sinal. Também será descrita alguma teoria acerca de algumas operações de processamento digital de sinal que serão utilizadas como base nas questões relacionadas com o processamento digtal de sinal no desenvolvimento da dissertação.

## 3.1 - Sinais e processamento de sinal

Os sinais desempenham um papel importante no nosso dia-a-dia. Entre exemplos de sinais que encontramos frequentemente destacam-se a fala, música, imagem e sinais de vídeo. Um sinal é uma função com variáveis independentes como tempo, distancia, posição, temperatura pressão, etc. Por exemplo, a fala e a música representam a pressão do ar como função do tempo num dado ponto no espaço. Uma fotografia a preto e branco é uma representação da intensidade de luz como função de duas coordenadas espaciais.

A maioria dos sinais que encontramos é gerada de modo natural. No entanto, um sinal pode ser gerado sinteticamente ou através de simulação computorizada. Um sinal contém informação e o objectivo do processamento digital de sinal é extrair a informação contida no sinal. O método de extracção da informação depende do tipo de sinal e da natureza da informação contida nesse sinal. Daí que, de modo geral, o processamento de sinal está directamente ligado à representação matemática do sinal e às aplicações algorítmicas efectuadas neste para extracção da informação presente [14].

Para poderem ser usadas as propriedades físicas das ondas acústicas na estimação do posicionamento de um VSA, é necessário a existência de um mecanismo que nos permita converter sinais acústicos em sinais analógicos e sinais analógicos em sinais digitais para posterior processamento, pois o processamento digital de sinal pressupõe a existência de um sinal discreto.

O processamento digital de um sinal analógico consiste basicamente em quatro passos: acondicionamento do sinal, conversão do sinal analógico para um formato digital, processamento da versão digital do sinal analógico e, dependendo da aplicação, a conversão do sinal processado de volta a uma forma analógica. A figura 2.1.1 mostra um sistema de PDS na forma de um diagrama de blocos.

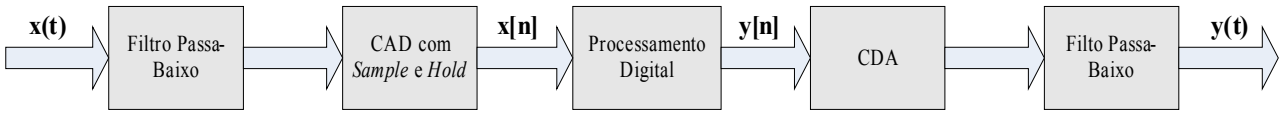



**Figura 3.1.1 - Diagrama de blocos simplificado de um sistema com PDS**

A razão de se chamar processamento digital de sinal decorre do facto de a realização de um sistema discreto implicar normalmente a digitalização das amostras dos seus sinais de entrada e saída analógicos.

## 3.2 - Breve história do processamento digital de sinal

Grande parte da base teórica do PDS está assente em modelos matemáticos básicos dos sinais e sistemas contínuos que tiveram origem no século XIX, com as transformadas de Fourier, Laplace e Z.

A transformada de Fourier foi baptizada com este nome em homenagem a Jean Baptiste Joseph Fourier que no seu trabalho "*Théorie Analytique de la Chaleur*" publicado em 1822 afirma que qualquer função de uma variável contínua ou descontínua pode ser expandida numa série de senos de frequência múltipla. Apesar deste resultado não ser correcto, a observação de Fourier de que algumas funções contínuas são a soma de séries infinitas foi um grande avanço e está na base do trabalho de Johann Dirichlet que foi o primeiro a demonstrá-lo satisfatoriamente com algumas condições restritivas denominadas as condições de Dirichlet [11] [12].

A transformada de Laplace recebeu o seu nome em homenagem ao matemático e astrónomo teórico Pierre-Simon, Marquês de Laplace, que a usou no seu trabalho *"Probability Theory"* e rapidamente foi aplicada em muitas outras áreas científicas. Também De Moivre, que em 1730 introduziu a hoje chamada transformada em z, deve ser creditado como um dos precursores do PDS.

Os maiores avanços na área do PDS começaram a surgir apenas a partir da década de 50 com a utilização dos computadores digitais no PDS. Engenheiros e cientistas como Shannon e Bode nos Bell Telephone Laboratories e Linville no MIT foram os primeiros a equacionar a utilização de computadores de sinal em PDS [13].

No início dos anos 60, Kaiser, nos laboratórios Bell, apresentou importantes contribuições para a análise e a síntese de filtros digitais, e a transformada rápida de Fourier (FFT) foi 'descoberta' em 1965 por Cooley e Tukey, apesar de a sua origem poder ser atribuída aos matemáticos alemães Runge e mesmo Gauss.

Actualmente, o PDS emergiu das aplicações militares e desempenha um papel chave em produtos de consumo, industriais e de telecomunicações. Microprocessadores DSP de baixo custo são componentes essenciais de jogos electrónicos, telefones celulares, brinquedos, leitores de CDs, discos de computadores, modems, impressoras, sistemas de reconhecimento de voz e de conferência vídeo, e muitos outros produtos familiares. Cada vez mais aplicações tradicionalmente do domínio dos sistemas analógicos estão a encontrar soluções digitais mais baratas e mais fiáveis [13].

## 3.3 - Porquê processamento digital de sinal

Ao contrário dos circuitos analógicos, o funcionamento dos circuitos digitais não depende de valores precisos dos sinais digitais. Em resultado disso, um circuito digital é menos sensível às tolerâncias dos valores dos componentes e é razoavelmente independente da temperatura, envelhecimento e da maioria dos parâmetros externos. Um circuito digital pode ser facilmente reproduzido em grandes quantidades e não necessita de ajustes durante a construção ou posteriormente durante o seu uso. Tendo ainda a vantagem de ser totalmente integrado e com os recentes avanços em "*very large scale integrated (VSLI) circuits*" tem sido possível a integração de sistemas de PDS muito sofisticados e complexos num único chip [14].

O processamento digital permite a partilha de um dado processador para processar vários sinais dividindo o tempo de processamento por cada sinal, reduzindo assim o custo de processamento por sinal.

A implementação digital permite o fácil ajuste das características do processador durante o processamento, tal como é necessário na implementação de filtros adaptativos. Tais ajustes podem ser executados simplesmente alterando periodicamente os coeficientes de um algoritmo que represente as características do processador. Outra aplicação de mudança de coeficientes na realização de um sistema com características programáveis é um sistema que necessite de filtros selectivos com frequências de corte ajustáveis.

Sinais digitais podem ser armazenados quase indefinidamente sem qualquer perda de informação na maioria dos equipamentos de armazenamento enquanto que sinais analógicos armazenados deterioram-se rapidamente ao longo do tempo, e não podem ser recuperados na sua forma original.

Outra vantagem é a aplicabilidade do processamento digital a sinais de muito baixa frequência, tais como os que ocorrem em aplicações que monitorizam actividades sísmicas, onde as bobinas e condensadores necessários para o processamento analógico teriam tamanhos físicos muito grandes.

Por outro lado, o processamento digital requer a utilização de circuitos de conversão, tais como conversores A/D e D/A. Uma desvantagem associada ao processamento digital de sinal é a gama limitada de frequências disponíveis para

processamento. Esta propriedade limita a aplicação do processamento digital em sinais analógicos. De um modo geral, um sinal analógico contínuo no tempo tem que ser amostrado a uma frequência pelo menos o dobro da componente com maior frequência presente no sinal. Se esta condição não for satisfeita, as componentes do sinal com frequências maiores que metade da frequência de amostragem aparecem como componentes do sinal com frequência menor do que esta frequência particular, distorcendo totalmente o sinal analógico inicial. Presentemente o limite máximo para a frequência de amostragem de um conversor A/D é tipicamente de 10MHz. Embora existam conversores com maior frequência de amostragem, até 1GHz, estas frequências de amostragem são tipicamente alcançadas às custas de uma menor resolução do conversor que neste caso é de 6 *bits*. Este tipo de conversores de elevada frequência de amostragem tem aplicações na área do vídeo ou instrumentação de medida, por exemplo, ou ainda outras aplicações onde é mais importante a taxa de refrescamento dos dados do que a resolução destes. As aplicações de processamento de sinal na linha das que são tratadas neste trabalho são necessários conversores A/D de maior resolução, tipicamente 12 a 16 *bits,* diminuindo a frequência de amostragem máxima dos conversores A/D disponíveis no mercado para esta gama de resolução.

Outra desvantagem advém do facto que os sistemas de processamento digital de sinal serem construídos usando dispositivos activos que consomem alguma potência eléctrica. Existe uma variedade de algoritmos de processamento que pode ser implementada usando circuitos passivos, tais como bobinas, condensadores e resistências que não precisam de alimentação [14].

No entanto, as vantagens superam as desvantagens em muitas aplicações e, com a diminuição crescente do custo do *hardware* para o processamento digital, as aplicações de processamento digital de sinal estão a aumentar rapidamente. Refira-se finalmente, que o processamento digital de sinal permite uma grande flexibilidade de reconfiguração das aplicações uma vez que a implementação destas baseia-se em larga medida no desenvolvimento de *software*.

## 3.4 - Correlação entre sinais

Uma das operações mais importantes no processamento digital de sinal é a correlação. Existem duas formas de correlação:

- Auto-correlação;
- Correlação cruzada.

A função de correlação cruzada é uma medida de similariedades ou propriedades partilhadas entre dois sinais. A aplicacação de função de correlação cruzada inclui detecção/recuperação de sinais com ruído como por exemplo detecção do eco de sinais dos radares, medidas de atrasos, etc. A função de correlação cruzada pode ser definida como:

$$\rho_{xy}(n) = \frac{r_{xy}(n)}{\left[r_{xx}(0)r_{yy}(0)\right]^{1/2}} \quad n = 0,\pm1,\pm2,... \tag{3.3.1}$$

onde $r_{xy}(n)$ é uma estimativa da covariância cruzada que é definida como:

$$r_{xy}(n) = \begin{cases} \frac{1}{N}\sum_{k=0}^{N-n-1} x(k)y(k+n) & n = 0,1,2... \\ \frac{1}{N}\sum_{k=0}^{N+n-1} x(k-n)y(k) & n = 0,-1,-2,... \end{cases} \tag{3.3.2}$$

$$r_{xx}(0) = \frac{1}{N}\sum_{k=0}^{N-1}\left[x(k)\right]^2, \quad r_{yy}(0) = \frac{1}{N}\sum_{k=0}^{N-1}\left[y(k)\right]^2 \tag{3.3.3}$$

onde $x(k)$ e $y(k)$ são vectores com tamanho *N* com média zero.

A função de auto-correlação envolve apenas um sinal e providencia informação acerca da estrutura do sinal ou do seu comportamento no domínio dos tempos. É uma forma especial da função de correlação cruzada, que se caracteriza pela correlação de dois sinais iguais, e é utilizada em aplicações similares à correlação cruzada. É particularmente útil na identificação de periodicidades escondidas em sinais. Uma estimativa da auto-correlação, $\rho_{xx}(n)$, de um vector de tamanho *N*, $x(k)$, com média zero é dada por:

$$\rho_{xx}(n) = \frac{r_{xx}(n)}{r_{xx}(0)} \quad n = 0,1,2,... \tag{3.3.4}$$

onde $r_{xx}(n)$ é uma estimativa da auto-covariância e é definida como:

$$r_{xx}(n) = \frac{1}{N}\sum_{k=0}^{N-n-1} x(k)x(k+n) \quad n = 0,1,2,... \tag{3.3.5}$$

A correlação é também uma parte integrante do processo de convolução. O processo de convolução é essencialmente a correlação de duas sequências de dados onde uma das sequências foi invertida no tempo. Isto significa que os mesmos algoritmos podem ser usados para computar a correlação e a convolução apenas invertendo uma das sequências. A implementação do processo de correlação pode ser conseguida utilizando, por exemplo, filtros FIR [15]. Neste trabalho será utilizada a correlacção entre o sinal recebido e uma cópia local do sinal transmitido, de forma a determinar o instante de recepção do sinal. Esta técnica é habitualmente reconhecida como *matched filtering*.

# Capítulo 4

# *Hardware* reconfigurável

Neste capítulo serão abordadas algumas das características mais importantes das FPGAs tais como, a sua estrutura, aplicacações, *tradeoffs* que se têm de fazer no desenvolvimento de um sistema baseado em FPGAs, etc. Também serão abordadas algumas das vantagens e desvantagens das FPGAs.

## 4.1 - Introdução

*Field Programmable Gate Arrays* (FPGAs) são uma classe de dispositivos lógicos programáveis baseados num *array* de células lógicas que podem ser interconectadas através de interruptores programáveis para formar circuitos complexos. É um dispositivo que pode ser programado para executar diferentes tipos de funções lógicas e a sua estrutura ou arquitectura é essencialmente um *array* de recursos que podem ser configurados e ligados *on-chip* para fazer um *chip* com uma finalidade específica para um sistema específico.

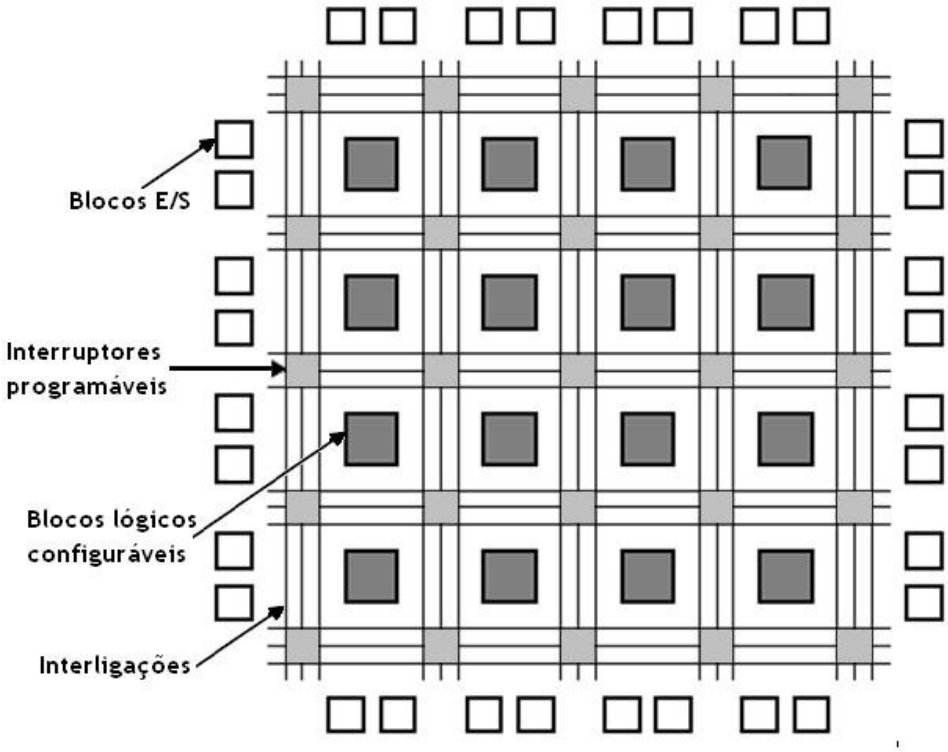



**Figura 4.1.1 – Diagrama conceptual da estrutura de uma FPGA**

Este diagrama básico é uma maneira útil de descrever a arquitectura básica de uma FPGA. Pode-se ver que no exterior da FPGA existem blocos de E/S (entradas/saídas) que providenciam uma interface entre os pinos da FPGA e a sua lógica interna, interruptores programáveis e as linhas de interconexão. No seu interior estão os *arrays* de blocos lógicos configuráveis, muitas vezes designados como CLBs *(configurable logic blocks)*, que providenciam os elementos funcionais para a construção do sistema lógico. Dependendo do tamanho da FPGA podem existir milhares destes CLBs. Entre estes CLBs estão os recursos de *routing* programáveis que conseguem interconectar os CLBs de modo a que os circuitos pretendidos sejam realizados.

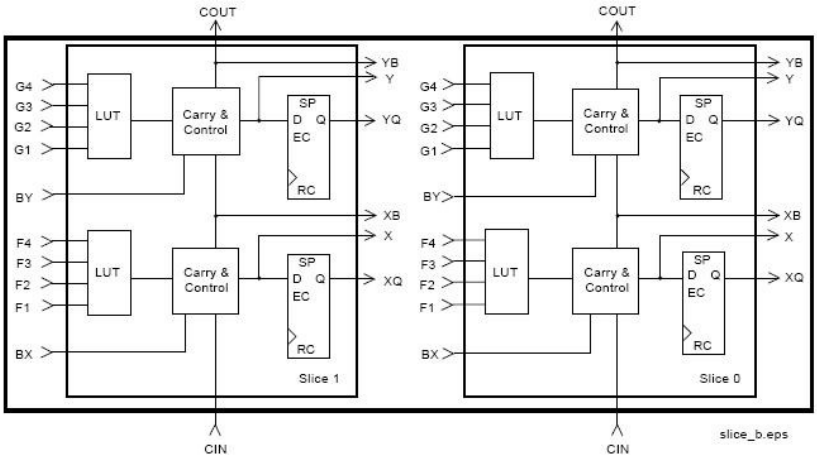



**Figura 4.1.2 – CLB com duas *slices* de uma FPGA da *Virtex***

Um CLB é constituído por duas *slices*. Uma *slice* é por sua vez constituída por duas células lógicas. A *Xilinx* (líder de mercado de FPGAs) define *slice* como sendo aproximadamente 2.25 células lógicas, isto porque as suas *slices* conseguem fazer mais por CLB do que em outras arquitecturas pois possuem dois f*lip-flops,* duas LTUs e *multiplexers, carrys* e controlo de lógica associados. Uma célula lógica é constituída por uma tabela de verdade de quatro entradas, também designada por LUT (*Look Up Table*), um *carry* lógico, um elemento de armazenagem e conexões com as células lógicas adjacentes.

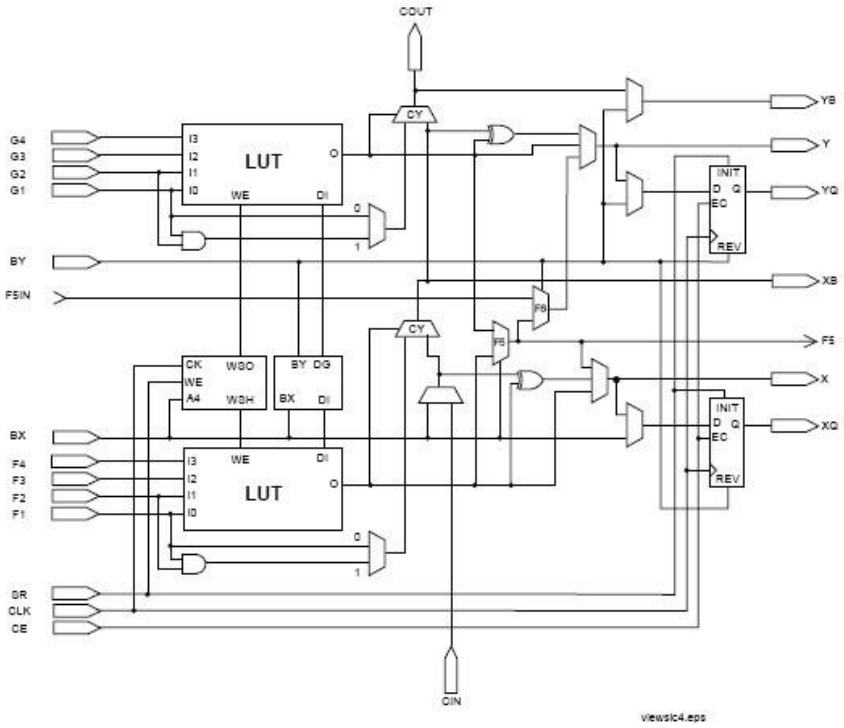



**Figura 4.1.3 – Vista detalhada de uma *slice* de uma FPGA da *Virtex***

Uma LUT utiliza lógica combinacional para implementar uma expressão de quatro entradas (ANDs, Ors, adições, etc). As LUTs são essencialmente pequenos blocos RAM que podem ser utilizadas apenas como tabelas de verdade ou interconectadas a outras LUTs para criar blocos RAM maiores. Esta arquitectura combinada beneficia o sistema final ao aumentar o desempenho na execução lógica pela FPGA. Na Virtex (FPGA de elevado desempenho da *Xilinx*), os geradores de funções são implementados através da interligação de várias LUTs. Para além de funcionarem como um gerador de funções, cada LUT consegue providenciar uma RAM síncrona de 16x1 *bit*. Adicionalmente, as duas LUTs de uma *slice* podem ser combinadas para criar uma RAM sincrona de 16x2 *bits* ou 32x1 *bit*, ou também uma RAM síncrona com dois portos de 16x1 *bits*. Uma LUT da *Virtex* pode também providenciar um *shift register* de 16 *bits* o que é ideal para capturar dados com grande cadência ou em *burst-mode*. Este modo pode também ser utilizado para guardar dados em aplicações de processamento digital de sinal [16].

O número de portas lógicas (*gates*) representa o número de combinações de blocos de memória, células lógicas, multiplicadores, etc. É importante ter em conta que a eficiência de uma porta lógica numa FPGA não é equivalente à eficiência de a uma porta lógica num ASIC (*Application-Specific Integrated Circuit*). Por exempo, um milhão de portas lógicas numa FPGA é aproximadamente equivalente a cem mil portas lógicas num ASIC.

Existem algumas características que diferenciam as FPGAs dos microcontroladores. Nos microcontroladores os circuitos lógicos básicos são predefinidos mas nas FPGAs os circuitos lógicos são criados a partir dos CLBs existentes das FPGAs. Outra diferença é que nos microcontroladores as suas funções lógicas são controladas através do desenvolvimento de *software* enquanto que nas FPGAs as suas funções lógicas são controladas através do desenvolvimento de circuitos lógicos que são criados a partir dos recursos existentes nas FPGAs.

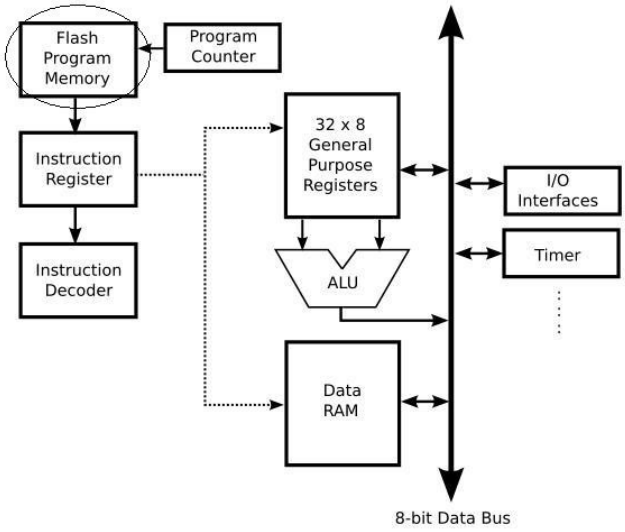



**Figura 4.1.4 – Arquitectura típica de um microcontrolador**

Pode-se reparar através figura 4.1.4 que as funções dos microcontroladores são predefinidas e todo o trabalho do projectista consiste basicamente em desenvolver o código do programa e guardá-lo numa memória *flash* designada por memória de programa. Numa FPGA todos os recursos são muito mais fléxiveis e estão prontos para serem modificados em novos circuitos na FPGA.

As FPGAs tornaram-se numa ferramenta a ter em conta no projecto de *hardware*. Nenhum outro tipo de dispositivo semiconductor *off-the-shelf* é capaz de ser utilizado em tantas aplicações. A versatilidade da arquitectura das FPGAs permite utilizar as FPGAs em muitas aplicações, desde num leitor MP3 até ao processamento de imagem. No entanto, à medida em que as arquitecturas das FPGAs evoluiram para serem utilizadas num maior número de aplicações, tornaram-se mais complicadas de utilizar. Hoje em dia é necessário *software* dedicado para sintetizar e implementar os projectos nas FPGAs.

Quando a tecnologia de FPGAs apareceu, o seu conceito era simples: criar um conjunto de células lógicas que possam ser programadas para produzir uma qualquer configuração lógica possível – basicamente, um conjunto de células lógicas que podesse ser projectada e programada a partir de um computador. Esta abordagem funciona bem para projectos simples mas é limitada para lidar com projectos mais complicados. Por exemplo, muitos projectos requerem grandes quantidades de memória e utilizar apenas células lógicas para criar memória é muito ineficiente. Os projectistas foram "forçados" a utilizar *chips* de memórias quando usavam FPGAs num projecto com grandes requisitos de memória, aumentando assim tanto o custo do sistema como também o tamanho da placa de circuito impresso. Os fabricantes de dispositivos lógicos programáveis responderam a esta mudança nos requisitos dos utilizadores introduzindo blocos especiais

que aparecem nas mais recentes arquitecturas de FPGAs. Colocando uma coluna com memórias RAM entre as colunas de células lógicas fez com que estes dispositivos programáveis fossem mais práticos para um conjunto muito maior de aplicações, permitindo que as necessidades de memória fossem satisfeitas. O mesmo aconteceu com a baixa performance das FPGAs quando se tentava sintetizar aplicações com um grande número de multiplicações relacionadas com algoritmos de processamento digital de sinal numa FPGA o que obrigou os fabricantes a incluir nas arquitecturas das suas FPGAs blocos com multiplicadores ou blocos DSP que, para além de serem eficientes em termos de área utilizada podem operar a uma frequência muito maior do que o circuito correspondente desenvolvido apenas através das células lógicas programáveis.

Cada vez mais tecnologia baseada em FPGAs está a ser utilizada para realizar tarefas de processamento digital de sinal, particularmente em áreas de aplicação que necessitam de grandes recursos computacionais tais como processamento de imagem e comunicações. O seu modo de funcionamento em paralelo permite, muitas vezes, que as FPGAs lidem com velocidades de processamento superiores às de circuitos integrados dedicados ao processamento digital de sinal e processadores de utilização geral. Recentemente, fabricantes de FPGAs começaram a incluir blocos com características orientadas ao processamento digital de sinal tal como multiplicadores nos seus *chips*, tornando as FPGAs numa solução mais interessante para muitas aplicações de processamento digital de sinal.

Mas implementar um algoritmo numa FPGA requer muito mais esforço de desenvolvimento comparativamente a um circuito integrado dedicado ao processamento digital de sinal ou um processador de utilização geral. Uma implementação eficiente numa FPGA envolve muitas escolhas subtis e complexos *tradeoffs*. Adicionalmente as linguagens de descrição de *hardware* e as ferramentas de projecto utilizadas para o desenvolvimento de projectos em FPGAs são ainda pouco familiares para a maioria dos engenheiros especializados em aplicações de processamento digital de sinal. Felizmente os fabricantes de FPGAs já providenciam ferramentas de alto nível com a finalidade de implementar algoritmos de processamento digital de sinal em FPGAs.

Frequentemente algoritmos de processamento digital de sinal são projectados e desenvolvidos utilizando linguagem MATLAB da companhia *The MathWorks* ou através de linguagem gráfica de blocos de diagramas tal como o *Simulink* também da *The MathWorks*. Implementações de algoritmos de processamento digital de sinal cuidadosamente optimizadas, historicamente, tomaram a forma de código RTL (*register transfer level*). Este código tem muito poucas semelhanças relativamente aos abstratos e intuitivos blocos de diagramas e código MATLAB.

Fabricantes de FPGAs e empresas independentes de ferramentas de projecto disponibilizam actualmente ferramentas de projecto para FPGAs que ajudam a diminuir esta diferença entre a descrição intuitiva de um algortimo e a sua implementação optimizada. A maioria destas ferramentas apoia-se nas liguagens *Simulink* e MATLAB para representar os algoritmos de processamento digital de sinal e possibilitar a sua implementação em FPGAs a partir destas linguagens de alto nível.

Tal como a maioria das linguagens de blocos de diagramas, o *Simulink* permite aos fabricantes providenciar blocos "customizados" de modo a que os utilizadores os configurem e apliquem no desenvolvimento de algoritmos. Os fabricantes de FPGAs e empresas de ferramentas de projecto para FPGAs, tipicamente, disponibilizam livrarias de blocos *Simulink* representando funções comuns de processamento digital de sinal, tais como filtros digitais ou FFT's. Quando os utilizadores desenvolvem os seus algoritmos utilizando estes blocos especiais as ferramentas de projecto para FPGAs conseguem

converter o algoritmo em um ficheiro implementável numa FPGA. Os líderes de mercado de FPGAs, Xilinx e *Altera*, ambos disponibilizam tais livrarias *Simulink* e ferramentas associadas. A *Xilinx* disponibiliza o *System Generator* e a *Altera* o *DSP Builder*.

Outras ferramentas de projecto para FPGAs tal como a *Synplify DSP* da *Synplicity* segue o mesmo rumo mas disponibiliza uma tecnologia independente do fabricante de FPGAs. A ferramenta *Synplify DSP* pode programar qualquer FPGA ou ASIC a partir das suas livrarias de blocos *Simulink* enquanto que as ferramentas disponibilizadas pela *Xilinx* e pela *Altera* são específicas para as suas FPGAs. Em vez de disponibilizar as suas próprias livrarias de blocos a empresa *Lattice Semiconductor* confia na ferramenta *Synplify DSP* para permitir a implementação, nos seus dispositivos de FPGAs, de um projecto baseado em *Simulink*.

O desenvolvimento de sistemas complexos em FPGAs é pouco comum, pois a programação em HDL *(Hardware Description Language),* VHDL ou Verilog, é pouco conhecida para a maioria das pessoas que desenvolvem *software* que estão mais confortáveis com a programação sequencial, como por exemplo a linguagem de programação C.

## 4.2 - Desafios de *designs* baseados em FPGAs

A implementação de algoritmos de processamento digital de sinal em FPGAs de um modo eficiente requer muitas escolhas subtis e *tradeoffs* no seu desenvolvimento. As ferramentas de projecto em FPGAs têm por isso que permitir aos seus utilizadores explorar esses *tradeoffs*. Uma das características dessas ferramentas é que devem permitir aos utilizadores controlar uma grande variedade de detalhes de implementação mantendo, no entanto, uma implementação tão intuitiva e abstracta quanto possível.

Algumas das escolhas e *tradeoffs* mais importantes a ter em conta pelos projectistas ao implementarem algoritmos de processamento digital de sinal em FPGAs envolve o desenvolvimento de uma arquitectura de fluxo de dados apropriada para cada porção do algoritmo. As FPGAs possuem grande flexibilidade, permitindo ao utilizador implementar diferentes maneiras de transferir dados entre porções do algoritmo. Os utilizadores de FPGAs têm de escolher entre técnicas de aritmética convencional ou aritmética distribuída, determinar o nível de paralelismo apropriado e decidir como transferir dados entre cada porção do algoritmo. Este tipo de escolhas está sujeito a restrições temporais e de *routing* e influenciam o desempenho, custo e consumo energético do sistema. Como resultado, projectar um fluxo de dados óptimo para implementar um algoritmo de processamento digital de sinal numa FPGA pode ser extremamente complexo.

As ferramentas de projecto em FPGAs respondem a estes desafios de optimização providenciando blocos IP (*Intelectual Property*) para as funções mais comuns em processamento digital de sinal. Por exemplo, a *Xilinx*, através do *System Generator* providencia um bloco denominado *FIR compiler* que gera automaticamente implementações eficientes de filtros digitais do tipo FIR, permitindo ao utilizador especificar os parâmetros do filtro tais como o número de coeficientes do filtro, precisão desejada, atraso do filtro, etc. Esta ferramenta permite também extraír os parâmetros de um filtro digital projecto através do *Simulink* e aplicá-los no filtro do bloco IP.

## 4.3 - FPGAs Vs microcontroladores

Várias diferenças podem ser encontradas entre FPGAs e microcontroladores sendo a que mais salta à vista é a grande flexibilidade das FPGAs relativamente aos microcontroladores. Enquanto que os registos e contadores dos microcontroladores, para além de serem em número muito limitado, têm também uma resolução fixa de 8 ou 16 *bits* numa FPGA consegue-se facilmente criar um contador com 3 *bits* para ser usado como *select* de um multiplexador 3x8 ou criar um contador de 50 *bits* para contar dias o que permite ao utilizador melhor distribuir os recursos da FPGA. Esta característica dos microcontroladores torna difícil a sua utilização com *encoders* de alta resolução que criam milhares de incrementos por cada volta de 360º sendo a captura de todos estes impulsos bastante difícil, enquanto que numa FPGA uma simples máquina de estados concectada a um contador assegura que nenhum impulso é perdido.

As FPGAs permitem também a possibilidade de implementar algoritmos em *Hardware* sem perder a sua versatilidade podendo suportar um grande número de escolhas algoritmicas providenciando uma elevada performance.

Os microcontroladores têm, no entanto, menor consumo energético, menor tamanho, grande poder de integração e menor tempo de desenvolvimento. As FPGAs necessitam de engenheiros especializados o que torna o desenvolvimento de projectos em FPGAs mais demorados e caros.

As FPGAs possibilitam a criação de módulos únicos que não são encontrados em nenhum microcontrolador, mais sofisticado ou não.

Hoje em dia é comum implementar nas FPGAs *soft cores*, como por exemplo o *picoBlaze* ou *microBlaze* para FPGAs de baixo custo ou o *PowerPC* para FPGAs de maior performance, que emulam CPUs e que podem executar sistemas operativos baseados em plataformas *Linux,* como por exemplo o *µClinux,* podendo complementar as FPGAs com periféricos especificos, tudo no mesmo *chip*. Esta prática de integração de todos estes componentes num único *chip* é conhecida como *system on chip* (SoC). Com a crescente disponibilidade de *soft* e *IP cores* implementados nas FPGAs através das ferramentas de projecto em FPGAs, esta tecnologia está a tornar-se cada vez mais atractiva para a implementação dos mais variados tipos de sistemas.

# Capítulo 5

# Linguagens de descrição de *hardware*

Neste capítulo serão abordados alguns aspectos relacionados com linguagens de descrição de *hardware* bem como algumas das ferramentas de projecto existentes que permitem a utilização dessas linguagens de descrição de *hardware* no desenvolvimento de projectos em FPGAs.

## 5.1 - Introdução

Para além de linguagens mais antigas, tal como o ABEL, ou outras que ainda não permitem a síntese directa de circuitos lógicos, por exemplo *SystemC,* existem neste momento duas linguagens *standard* de descrição de *hardware,* o VHDL (*Very high speed integrated circuit Hardware Description Language*) e o *Verilog*. A complexidade dos projectos em FPGAs tem aumentado cada vez mais e com isso, tem aumentado também o número de engenheiros especializados nesta área que têm preferências diferentes, no que a linguagens de descrição de *hardware* diz respeito. Como resultado, é importante que as ferramentas de projecto em FPGAs providenciem um ambiente que permita o uso de ambas as linguagens simultaneamente. Por exemplo, utilizar um modelo de uma interface de um barramento PCI *(Peripheral Component Interconnect)* escrito em VHDL simultaneamente com uma *macro* escrita em *Verilog*.

Exitem dois aspectos para modelar *hardware* que qualquer linguagem de descrição de *hardware* permite:

- Abstração de comportamento;
- Estrutura do *hardware*.

Isto implica que o comportamento do *hardware* modelado não é prejudicado por aspectos estruturais, ou de *design,* do *hardware* e que a estrutura do *hardware* tem capacidade de ser modelada independentemente do comportamento do seu *design*. A estrutura do *hardware* pode ser modelada com igual eficiência tanto em VHDL como em *Verilog*. A escolha de qual usar não é, portanto, baseada somente nas suas capacidades técnicas mas na:

- Preferência pessoal;
- Ferramentas de projecto disponíveis;
- Questões comerciais e de *marketing*.

Um outro factor de escolha pode residir na facilidade de aprendizagem, caso não se esteja familiarizado com nenhuma da linguagens de descrição de *hardware*. O V*erilog* é, provavelmente, mais fácil de compreender e de interiorizar os conceitos de programação de *hardware*. A forma como se descreve o *hardware* e como se modeliza o seu comportamento é bastante intuitiva e facilmente se compreende o raciocínio lógico inerente à sua realização. A descrição e modelização do *hardware* é executada basicamente, seguindo os seguintes passos:

- Criar um módulo de *hardware,* ou componente*;*
- Descrever os seus pinos;
- Modelizar o seu comportamento.

Se o sistema a implemetar for constituído por vários módulos apenas é necessário seguir estes passos novamente sendo necessário ter algum cuidado na descrição dos módulos, pois diferentes módulos podem necessitar dos mesmos pinos para funcionarem correctamente como por exemplo, um *clock* ou um *enable*.

Na criação de um módulo, por exemplo um contador, descrevem-se todos os seus pinos como por exemplo, os portos de entrada *(clock, reset, enable, clear,* etc*)* e saída, bem como o tipo de dados que lhes vão estar associados (*wire,* registo, etc). Após a descrição dos pinos do componente criam-se os blocos, dentro do módulo de *hardware,* onde se modeliza o comportamento do componente. Estes blocos são constituídos por porções de código *Verilog,* que modelizam o comportamento do componente. A execução dessas porções de código são feitas em paralelo, isto é, todos as instruções dos blocos são executadas no mesmo instante de *clock*. É necessário, no entanto, ter em conta que existem alguns comandos que não são sintetizáveis. Embora possam ser compilados e simulados, se quisermos sintetizar o módulo desenvolvido para o implementar na FPGA ocorrerá um erro e a sua implementação na FPGA ficará impossibilitada. O processo de sintetização do projecto desenvolvido, é um procedimento executado para criar um ficheiro binário, que trasferido para a FPGA, permite a sua programação.

A linguagem de descrição de *hardware* VHDL tem conceitos muito similares aos de *Verilog,* embora o modo como se criam os módulos de *hardware*, como se descrevem os seus pinos e como se modeliza o seu comportamento difere bastante do *Verilog* e a própria sintaxe da linguagem tem bastantes diferenças. A descrição e modelização de *hardware* em VHDL é executada basicamente, seguindo os seguintes passos:

- Criar uma entidade de *hardware* ou componente;
- Criar uma arquitectura comportamental;
- Criar os processos que modelizem o *hardware*.

Estes três passos na linguagem de descrição de *hardware* VHDL podem ser equiparados com o mesmo procedimento em *Verilog* e correspondem basicamente, à criação de um módulo de *hardware,* descrever os seus componentes e modelizar o seu comportamento, respectivamente, embora contendo algumas *nuances* que os diferenciam.

Na entidade, recorrendo ao mesmo exemplo do contador utilizado para o *Verilog,* são descritos todos os pinos que fazem parte do contador tais como, portos de entradas *(clock, reset, clear, enable,* etc*)* e de saídas e também o tipo de dados associados a cada pino ou conjunto de pinos. Na arquitectura comportamental são descritos os sinais temporários que têm necessidade de serem utilizados, quer em termos do seu tamanho quer em termos do seu tipo (vector, *array, ect*) podendo-se também aqui inicializar as constantes "globais" desta arquitectura comportamental. Dentro da arquitectura comportamental são criados também os processos que modelizam o comportamento do componente. Estes processos são porções de código VHDL, cujas instruções são executadas paralelamente, isto é, em cada ciclo de *clock* é executada uma instrução de cada porção de código.

A linguagem de descrição de *hardware* VHDL pode parecer menos intuitiva do que o Verilog pois específica algumas restrições no modo como as operações que envolvem variáveis de diferentes tipos de dados, isto é, objectos de dados de diferentes tipos não podem ser atribuídos um ao outro, sem conversão especifica. No entanto, esta é uma característica que a torna robusta e eficaz para utilizadores experientes depois de uma longa fase de aprendizagem. O VHDL tem também a vantagem, relativamente ao *Verilog,* de permitir muitas maneiras diferentes de modelizar o mesmo circuito, especialmente aqueles com grandes estruturas hierárquicas.

## 5.2 - Ferramentas de projecto

Como os projectos em FPGAs podem processos complexos, os fabricantes de FPGAs disponibilizam ferramentas de desenvolvimento de *software* que permitem ao utilizador contornar alguma desta complexidade. No entanto estas ferramentas têm o inconveniente do facto de a compilação de um projecto ser sempre muito demorada, mesmo para projectos relativamente simples. À medida que a complexidade dos projectos aumenta é necessário ter algum cuidado na altura de se decidir compilar o projecto pois esta é uma operação que pode demorar mais de uma hora mesmo utilizando um computador recente.

Para o desenvolvimento de projectos em FPGAs cada fabricante de FPGAs fornece as suas ferramentas de projecto desenvolvidas especificamente para as suas FPGAs, o que faz com que ao escolhermos uma FPGA de um dado fabricante, estamos implicitamente obrigados a recorrer às suas ferramentas de projecto. No caso da *Xilinx,* esta tem disponíveis várias ferramentas de projecto em que cada uma delas tem a sua finalidade específica. Do leque de ferramentas de projecto disponibilizadas pela *Xilinx,* as mais importantes são:

- ISE *(Integrated Software Environment);*
- EDK *(Embedded Devolpment Kit);*
- *System Generator*.

O ISE é a ferramenta base para o desenvolvimento de projectos nas FPGAs da *Xilinx*, pois todas as suas outras ferramentas necessitam da instalação do ISE. Esta ferramenta tem um grande leque de utilidades integradas tais como análises temporais, *routing* e programação dos dispositivos. Esta ferramenta tem também *templates* de programação em *Verilog* e VHDL, que vão desde contadores ou máquinas de estados até operadores aritméticos. A *Xilinx* fornece uma versão desta ferramenta *online* para *download*, embora a verão completa desta ferramenta tenha um custo associado à sua aquisição.

O EDK é um conjunto de ferramentas e de blocos IP que permitem ao utilizador projectar um sistema completo com processamento embebido e implementá-lo num dispositivo FPGA da *Xilinx*. É necessário no entanto que o ISE esteja instalado para se poder utilizar esta ferramenta. Uma dessas ferramentas é o XPS *(Xilinx Platform Studio)*. O XPS é um ambiente de desenvolvimento ou GUI *(Graphical User Interface)* utilizado para projectar a parte de *hardware* do sistema com processamento embebido. A figura seguinte mostra um diagrama que descreve de modo simplificado o fluxo do processo de desenvolvimento de um sistema em H*ardware* reconfigurável com processamento embebido.

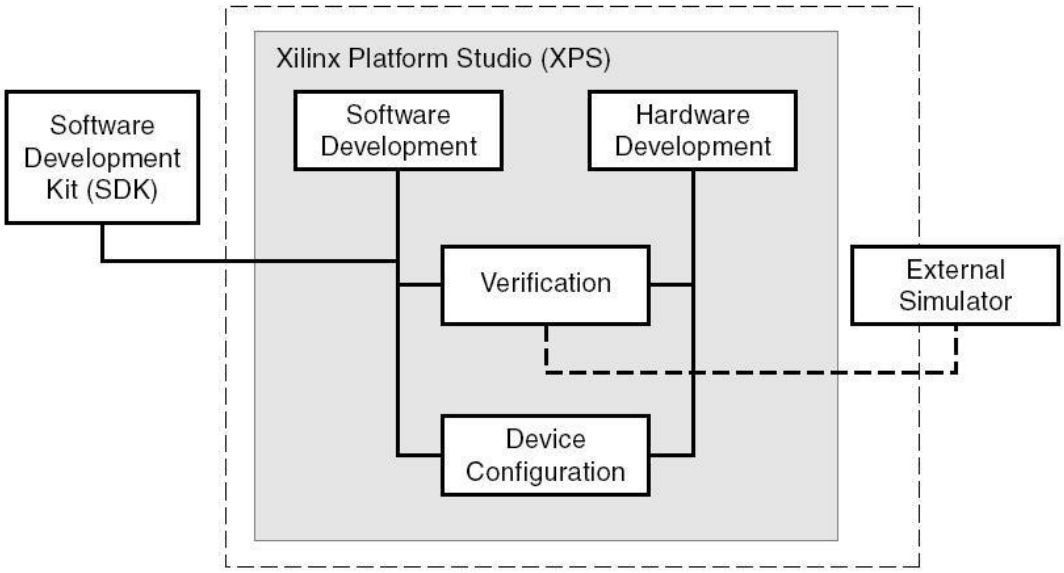



**Figura 5.2.1 – Fluxo do processo de desenvolvimento de um sistema embebido**

Outra ferramenta existente na ferramenta de projecto EDK é o SDK *(Software Development Kit)*. O SDK é um ambiente de desenvolvimento integrado, complementar ao XPS, que é utilizado para a criação e verificação de aplicações de *software* em linguagem de programação C/C++. O SDK foi desenvolvido numa estrutura de trabalho *opensource* da *Eclipse*™. O EDK inclui outros elementos tais como:

- *Hardware* IP para os processadores embutidos da *Xilinx;*
- *Drivers* e livrarias para o desenvolvimento de *software;*
- Compilador GNU para o desenvolvimento de *software* em linguagem de programação C/C++ para aplicação nos processadores *MicroBlaze™ e PowerPC™*.

Estes elementos têm como objectivo ajudar os utilizadores em todas as fases do desenvolvimento de sistema com processamento embebido. Não existe, no entanto, nenhuma versão do EDK grátis. Apenas uma versão para experimentar com 60 dias de validade [17].

O *System Generator* é uma ferramenta, que pode ser vista como uma extensão do *Simulink* para providenciar um ambiente de modelização que seja adequado ao desenvolvimento de *hardware,* especialmente para sistemas de processamento digital de sinal de alta performance. Esta, é uma ferramenta de alto nível que utiliza livrarias IP, quando apropriado, para providenciar implementações eficientes de funções existentes nesses blocos e permite sintetizá-los, de modo a poderem ser implementados numa FPGA, muito facilmente. Estas livrarias incluem funções de complexidade elevada tais como FFTs ou filtros digitais. A ferramenta *System Generator* não substitui os projectos baseados em linguagens de descrição de *hardware,* por exemplo não permite definir entradas e saídas bidirecionais obrigando ao desenvolvimento de componentes para serem integrados externamente através de *Verilog* ou VHDL, mas faz com que seja possível focar a atenção do utilizador apenas nas partes críticas do sistema. Uma questão crítica na instalação desta ferramenta é a sua compatibilidade com o MATLAB, isto é, cada versão do S*ystem Generator* apenas é compatível com uma versão específica do MATLAB, o que implica à aquisição de uma nova versão do MATLAB sempre que existe uma nova versão do *System Generator* e como esta é uma tecnologia bastante recente está sujeita a actualizações de *software*. Esta ferramenta reserva algumas surpresas interessantes para quem ainda está numa fase de aprendizagem do mundo da tecnologia de *hardware* reconfigurável. O facto de esta ferramenta estar muito mais evoluída tecnologicamente do que as FPGAs permite a idealização de algumas funções que após serem sintetizadas ocupam a maioria das *slices* disponíveis da FPGA, ou até mais do que as *slices* disponíveis para o caso de FPGAs de baixo custo, não possibilitando a implementação de todos os componentes que o *System Generator* possui.

# Capítulo 6

# Desenvolvimento da dissertação

Neste capítulo serão abordados todos os passos executados no desenvolvimento da dissertação desde a abordagem ao problema, escolha de *hardware* e de ferramentas de projecto, opções tomadas ao longo da dissertação bem como uma descrição do sistema a desenvolver.

## 6.1 - Introdução

A estimação da direcção do veículo subaquático autónomo é baseada nos tempos de atraso na detecção de uma onda acústica entre dois transdutores e recorrendo à técnica de posicionamento acústico USBL como ponto de partida para o desenvolvimento do sistema.

O processo de estimação da direcção do VSA é inspirado no que acontece no córtex auditivo do cérebro dos mamíferos, uma vez que estes conseguem bons resultados apenas com dois "sensores". No processo de detecção da direcção das ondas sonoras, pelos mamíferos, é estimada a diferença do tempo de chegada do som entre os dois ouvidos, criando a sensação da proveniência azimutal das ondas sonoras sendo também utilizada a diferença de intensidades entre ouvidos para sons de baixas frequências.

## 6.2 - Abordagem ao problema

Uma das primeiras decisões foi qual a linguagem de descrição de *hardware* que se deveria utilizar. Devido aos poucos conhecimentos acerca da tecnologia de *hardware* reconfigurável, optou-se pela linguagem de descrição de *hardware* que, à partida, seria de maior facilidade de aprendizagem, compreenção da sintaxe e do modo como esta descreveria *hardware*. O ferramenta de projecto utilizada para cimentar conhecimentos acerca do *Verilog* foi o ISE da X*ilinx*. A implementação de um sistema numa FPGA passa basicamente por três fases:

- Modelização;
- Simulação;
- Sintetização.

Na fase de modelização para além de se modelizar o funcionamento do sistema que se quer implementar pode-se atribuir também os objectos do programa aos pinos da *board* FPGA. Por exemplo atribuir um porto definido como saída de um *bit* a um *led* ou um porto definido como uma entrada de um *bit* a um *switch*. Existem duas maneiras de executar esta operação. Através de uma das ferramentas existentes no ISE, o *Xilinx* PACE *(Pinout and Area Constraint Editor)*, que é um ambiente gráfico que exporta todos os objectos definidos como portos, quer de entradas quer de saídas, sendo apenas necessário atribuir a cada *bit* de cada porto o pino da *board* FPGA correspondente. Ou através de um ficheiro com extensão ucf onde se editam todas as atribuições necessárias escrevendo uma linha de código para atribuir cada *bit* de cada objecto ao pino da *board* FPGA correspondente. Esta segunda opção raramente se usa pois o próprio PACE actualiza este ficheiro sempre que se atribuem objectos aos pinos da *board* FPGA.

Na fase de simulação verifica-se se o funcionamento do sistema desenvolvido é o desejado. Para tal é necessário construir um código de simulação, na linguagem de descrição de *hardware* que se usou para desenvolver o sistema, onde se modeliza o funcionamento das entradas de modo a se poder verificar se as saídas correspondem às esperadas e também se pode inicializar as variáveis que não tenham sido inicializadas no código principal. Pode-se também verificar se os requisitos temporais do sistema são cumpridos.

Na fase de sintetização do sistema é criado um ficheiro binário ou *bitstream*. Este ficheiro, resultante da sintetização do sistema, é transferido para a FPGA de modo a que esta seja configurada e execute o sistema desenvolvido. Caso a *board* FPGA tenha memória *flash* que possa ser usada pelo utilizador, permite que este ficheiro possa ser lá guardado e caso falhe a alimentação a FPGA não é necessário transferir novamente o ficheiro. Caso contrário quando há falha de alimentação da FPGA, as suas configurações são perdidas e é necessário transferir o ficheiro binário novamente para a FPGA.

Ao longo da aprendizagem da linguagem de descrição de *hardware Verilog* o *System Generator* surgiu como uma opção válida para o desenvolvimento da dissertação pois oferece muitas vantagens destacando-se o facto de esconder detalhes de *hardware* para facilitar a abordagem a processos de processamento digital de sinal, possibilita a geração de código HDL eficiente a partir do modelo desenvolvido e aproveita os recursos de visualização, análise e simulação do *Simulink*.

Esta ferramenta possibilita também a interacção entre uma simulação do sistema desenvolvido no computador e a execução do *hardware* na FPGA simultaneamente.

Uma das desvantagens da ferramenta de projecto S*ystem Generator* é o facto de muitas soluções que esta ferramenta possibilita não são possíveis de implementar na FPGA pois ocupam muitas *slices*. Por exemplo uma FFT com uma janela de 1024 pontos ocupa mais de 1930 slices. Como a *board* de desenvolvimento de FPGAs, ou *board* FPGA, escolhida apenas possui 1930 *slices* uma FFT com estas características não podia ser implementada por falta de *slices*. O mesmo acontece com os filtros FIR de ordem elevada (banda de sintonia estreita), que ocupam muitas *slices* o que obriga à procura de soluções menos dispendiosas em termos de *slices* utilizadas.

## 6.3 - Escolha da *board* FPGA

A escolha da *board* FPGA ideal numa fase embrionária do projecto mostrou-se ser um problema de difícil resolução. Uma das questões essenciais na escolha da *board* FPGA ideal é a sua capacidade, relativamente ao número de *slices* disponíveis para a implementação do projecto desenvolvido. O facto de inicialmente não se conhecer, nem se ter uma vaga ideia, do espaço ocupado pelo projecto não permite uma escolha acertada da *board* FPGA a utilizar. Daí a escolha recaír na utilização de uma plataforma de *hardware* reconfigurável de baixo custo que, para além de ser uma boa plataforma de aprendizagem, permite também ter a percepção das capacidades desta tecnologia sem recorrer ao gasto muito dinheiro. Neste caso a escolha recaíu numa *Spartan3 Starter Kit* da *Xilinx*.


**Figura 6.3.1 – *Board* FPGA *Spartan-3 Starter Kit***

Um factor crucial desta escolha, foi o facto de alguns dos laboratórios da FEUP já possuírem licenças para a utilização das ferramentas de projecto da *Xilinx*, o que permitiu desenvolver o projecto sem recorrer à aquisição de ferramentas de projecto de outros fabricantes de FPGAs. De facto, esta decisão revelou-se bastante acertada pois foram várias as vezes em que as soluções pensadas para o desenvolvimento deste projecto ocupavam demasiado espaço em termos de *slices*.

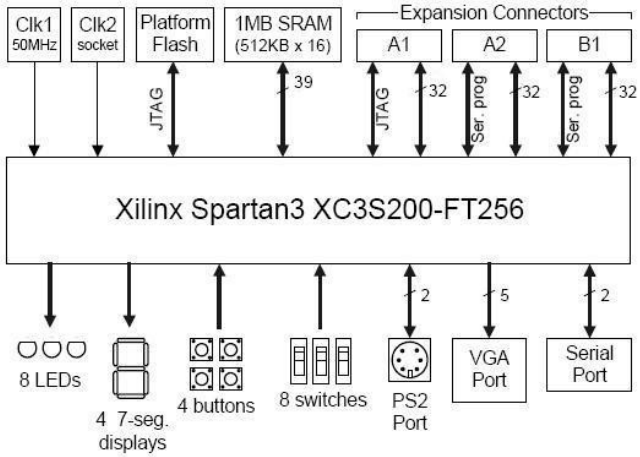



**Figura 6.3.2 – Diagrama de blocos da *Spartan-3 Starter Kit***

A partir do diagrama de blocos podemos ver algumas das características que tornam esta *board* indicada para uma primeira abordagem à resolução do problema em causa. Entre as quais se podem destacar uma plataforma *flash* com uma memória *flash* de 1 *Mbit* onde se podem guardar configurações da FPGA ou o código de uma aplicação do *Microblaze*, três expansores de conexões de 40 pinos onde se podem interligar conversores AD ou placas de expansão como por exemplo uma placa *ethernet* visto que esta *board* não possui nenhuma. Tem também uma velocidade de *clock* interna de 500 MHz embora a velocidade de *clock* relativa ao *hardware* é de 50 MHz no máximo.

## 6.4 – Desenvolvimento do sistema

Este sistema foi desenvolvido no laboratório *Oceansys* e no laboratório I222 ambos pertencentes à Faculadade de Engenheiraria da Universidade do Porto. A implementação do sistema em tecnologia de *hardware* reconfigurável foi elaborada recorredo às ferramentas de projecto System Generator, *Simulink* e Matlab.

### 6.4.1 - Geometria do problema

Para estimar a direcção do VSA foram utilizados dois transdutores, estando afastados um do outro em 20 cm. Na figura seguinte podemos observar a disposição dos transdutores bem como a geometria utilizada para desenvolver as equações usadas na estimação da direcção da fonte sonora, neste caso um VSA.

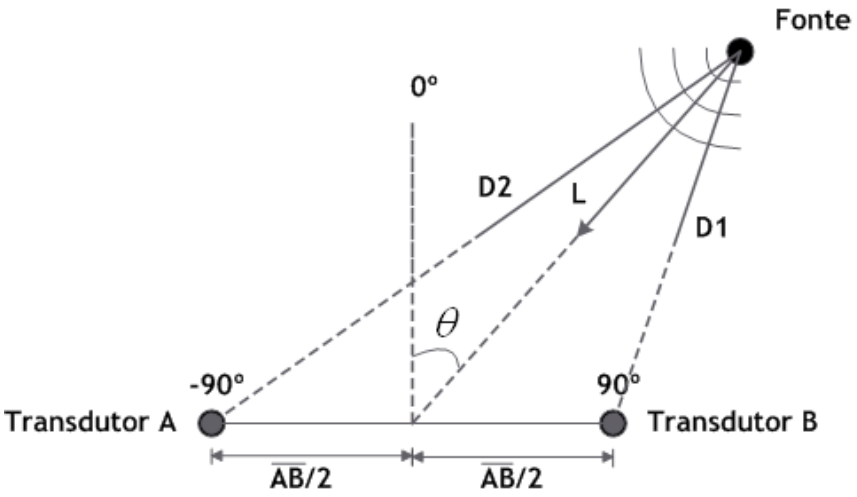



**Figura 6.4.1 – Diagrama geométrico do sistema**

Observando o diagrama da figura 6.4.1 podemos verificar que os valores de $D1$ e $D2$ podem ser obtidos a partir das seguintes equações:

$$D2 = \sqrt{(L\cos(\theta))^2+(\overline{AB}/2+L\sin(\theta))^2} = \sqrt{L^2+\overline{AB}^2/4+\overline{AB}\,L\sin(\theta)} \qquad (6.4.1.1)$$

e

$$D1 = \sqrt{(L\cos(\theta))^2+(L\sin(\theta)-\overline{AB}/2)^2} = \sqrt{L^2+\overline{AB}^2/4-\overline{AB}\,L\sin(\theta)} \qquad (6.4.1.2)$$

onde $\overline{AB}$ é a distância entre os dois transdutores. Como $\overline{AB}$ < 1, podemos aproximar os valores de $D1$ e $D2$ por:

$$D1 \approx \sqrt{L^2-\overline{AB}\,L\sin(\theta)} \qquad (6.4.1.3)$$

e

$$D2 \approx \sqrt{L^2+\overline{AB}\,L\sin(\theta)} \qquad (6.4.1.4)$$

Para valores de $L$ >> $\overline{AB}$ é fácil verificar que $L^2$ >> $\overline{AB}^2$, logo podemos aproximar as equações dos valores de $D1$ e $D2$ por:

$$D1 \approx \sqrt{L^2-\overline{AB}\,L\sin(\theta)} = L\sqrt{1-\frac{\overline{AB}}{L}\sin(\theta)} \qquad (6.4.1.5)$$

e

$$D2 \approx \sqrt{L^2+\overline{AB}\,L\sin(\theta)} = L\sqrt{1+\frac{\overline{AB}}{L}\sin(\theta)} \qquad (6.4.1.6)$$

Sabendo que:

$$\sqrt{1+x} \approx 1+x/2 \tag{6.4.1.7}$$

Podemos aproximar novamente as equações dos valores de $D1$ e $D2$ por:

$$D1 \approx L(1+\frac{\overline{AB}}{2\mathrm{L}}\sin(\theta)) \tag{6.4.1.8}$$

e

$$D2 \approx L(1-\frac{\overline{AB}}{2\mathrm{L}}\sin(\theta)) \tag{6.4.1.9}$$

Subtraindo $D2$ por $D1$ obtemos:

$$D2 - D1 \approx \overline{AB}\sin(\theta) \tag{6.4.1.10}$$

Este mesmo resultado poderia ser facilmente obtido considerando que para distâncias muito grandes da fonte a frente de onda pode ser localmente aproximada por um plano (aproximação de onda plana).

Sabendo que uma variação de tempo é equivalente à divisão de uma variação de distância por uma velocidade, obtemos:

$$\Delta T \approx \frac{D2-D1}{c} \approx \frac{\overline{AB}}{c}\sin(\theta) \tag{6.4.1.11}$$

onde, $c$ é a velocidade do som na água e $\Delta T$ a diferença de tempo da passagem de um sinal acústico entre os trandutores A e B. O valor de $\Delta T$ pode ser estimado através da medição da diferença entre os tempos de detecção do sinal acústico pelos transdutores A e B. Esta diferença pode ser determinada a partir da seguinte equação:

$$\Delta T \approx n_s T_s \tag{6.4.1.12}$$

onde, $n_s$ é a quantidade de amostras que ocorrem entre a detecção de um sinal acústico nos transdutores A e B, e $T_s$ é o período de amostragem. Se igualarmos as equações 6.4.1.10 e 6.4.1.11 obtemos:

$$n_s T_s \approx \frac{\overline{AB}}{c}\sin(\theta) \qquad (6.4.1.13)$$

Podemos então concluir que a direcção do VSA é dada por:

$$\theta \approx \arcsin\left(\frac{n_s T_s c}{\overline{AB}}\right) \qquad (6.4.1.14)$$

Esta aproximação é apenas válida para distâncias da fonte muito maiores do que as distâncias entre os transdutores A e B ( $\overline{AB}$ ), tipicamente pelo menos dez vezes maiores (como o que acontece nos sistema USBL) e apenas fornece informação azimutal acerca do VSA num plano.

Apenas com os transdutores A e B, só temos a possibilidade de estimar a direcção do VSA entre $\pm 90º$. Daí que, para eliminar a ambiguidade dos valores estimados, seja necessário orientar os transdutores na direcção do VSA quando este esteja próximo de $\pm 90º$. Preferencialmente, a orientação dos transdutores deverá ser feita quando o VSA se aproxima de $\pm 60º$, pois é a partir desta gama de valores que a função arcsin( $\theta'$ ) se torna não linear como se pode verificar na figura seguinte.

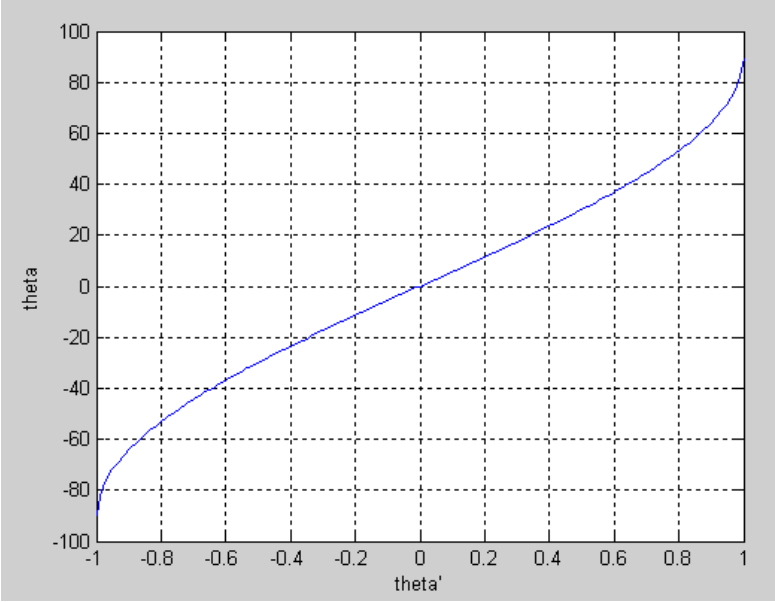



**Figura 6.4.2 – Função arcsin( $\theta'$ )**

Esta característica da função arcsin( $\theta'$ ) implica que haja uma diminuição da precisão na estimativa da orientação do VSA com o aumento do ângulo $\theta$. De facto, se atendermos à derivada de $\theta$ em ordem a $\Delta T$, da equação 6.4.1.11, que nos mostra o modo como varia o ângulo $\theta$ em função da variação de $\Delta T$, verificamos que quando o ângulo $\theta$ tende para $\pm 90º$ o $\Delta T$ tende para $\pm\infty$, pois depende do inverso de um coseno.

$$\frac{d\theta}{d\Delta T} = 1/(\frac{d\Delta T}{d\theta}) \approx \frac{c}{\overline{AB}\cos(\theta)} \quad (6.4.1.15)$$

Isto significa que à medida que os ângulos se aproximam de $\pm 90^{o}$ são necessárias cada vez maior número de amostras para mantermos a mesma resolução na estimativa da direcção do VSA, como isso não é possível há uma deterioração na precisão da estimativa.

## 6.4.2 - Detecção de um sinal pseudo-aleatório

De modo a se estimar o tempo de atraso entre a detecção de uma onda acústica entre os dois transdutores é necessário encontrar um modo eficaz de o fazer. Após algumas experiências com tentativas de implementar FFTs e filtros digitais do tipo FIR chegou-se à conclusão que estes últimos permitiam soluções com ocupação de um muito menor número de *slices*.

Uma maneira simples de detectar o atraso de um sinal acústico é correlacionando o sinal detectado com o sinal que se espera detectar. Para realizar a correlação dos sinais foi utilizado um filtro FIR na sua forma directa configurado com os coeficientes do sinal que se quer detectar de modo a que o filtro funcione como um correlacionador. A equação de um filtro do tipo FIR é:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \quad (6.4.2.16)$$

Onde $x(n)$ e $y(n)$ são as entradas e saídas do filtro, respectivamente, e $h(k),\ k=0,\ 1,\ldots,\ N-1,$ são os coeficientes do filtro. Para um dado filtro os valores dos seus coeficientes são únicos e determinam as características do filtro. Olhando para a equação do filtro, verifica-se que de facto a saída do filtro é a convolução do sinal de entrada com a resposta impulsional do filtro no domínio dos tempos. Se invertermos uma das sequências, por exemplo a ordem dos coeficientes do filtro, temos de facto uma correlação entre o sinal de entrada, que é o nosso sinal detectado, e a resposta impulsional do filtro que é o nosso sinal esperado. Este tipo de filtragem é também conhecida como *matched filtering* [15].

A grande vantagem deste tipo de filtragem, para além de ser de fácil implementação, é o facto de permitir detectar, de forma óptima, o instante de chegada de um dado sinal que se conhece. De facto, ao correlacionar o sinal recebido com uma cópia do sinal que se espera receber obtém-se um sinal que terá um valor máximo no instante correspondente ao instante de detecção do sinal.

Para que se possa ter uma boa imunidade ao ruído no processo de correlação escolheu-se utilizar um sinal pseudo-aleatório, o que garante que a correlação tenha um pico aguçado, quando os dois sinais estão correlacionados, obtendo-se assim uma grande precisão no instante de detecção e com bastante imunidade ao ruído. A utilização da correlação entre dois sinais pseudo-aleatórios permite que se possa verificar a existência de correlação entre os dois sinais mesmo quando o sinal detectado tem bastante ruído, o

que é frequente acontecer em ambientes acústicos, e que o erro na estimativa de $\Delta T$ seja minizado, pois apenas utilizando um detector de máximo podemos marcar com precisão o instante em que um sinal é detectado.

## 6.4.3 - Modulação de fase de uma sequência pseudo-aleatória

O primeiro passo na criação do sinal pseudo-aleatório passível de ser utilizado pelo sistema de transmissão acústico do VSA é a criação de uma sequência pseudo-aleatória, por exemplo através do MATLAB. De modo a podermos gerar e emitir um sinal pseudo-aleatório a partir do transdutor do VSA é necessário modular essa sequência.

A modulação de sinais é um processo extremamente comum em processamento digital de sinal, pois os sinais digitais raramente são transmitidos através de longas distâncias ou guardados em grandes quantidades na sua forma original por questões relacionadas com as características em frequência do canal de transmissão. Os sinais são normalmente modulados para aproximar as suas características às características dos dispositivos de transmissão ou de armazenamento de modo a minimizar distorções de sinal, utilizar a largura de banda disponível eficientemente ou para assegurar que os sinais têm as características desejadas. O processo de modulação envolve normalmente modificar uma propriedade de um sinal de alta frequência, conhecido como portadora, em consonância com o sinal que queremos transmitir ou guardar, chamado de sinal modulado. Os mais básicos processos de modulação digital de sinais mais são o ASK *(Amplitude Shitf Keying),* PSK *(Phase Shift Keying) e* FSK *(Frequency Shift Keying)* [15].

Devido ao facto de os sistemas acústicos operarem em bandas estreitas de frequência e apenas poderem transmitir sinais sinusoidais, foi necessário modular a sequência pseudo-aleatória de maneira a criar um signal acústico pseudo-aleatório.

De maneira a que seja possível transmitir um sinal pseudo-aletório a partir de uma sequência pseudo-aletória é necessário definir a forma da onda a transmitir para cada valor binário da sequência pseudo-aleatória. Por exempo para cada 0 (zero) da sequência pseudo-aleatória transmite-se um perído de um sinal equivalente a um seno e para cada 1 (um) transmite-se um período de um sinal equivalente a um -seno. Finalmente a sequência pseudo-aleatória é obtida atribuindo o valor binário 1 à arcada positiva e atribuindo o valor binário -1 à arcada negativa do sinal sinosóidal transmitido. De notar que, ao amostrar esta sequência de valores binários de 1s e -1s à frequência de amostragem utilizada obtemos os coeficientes do *matched filter*.

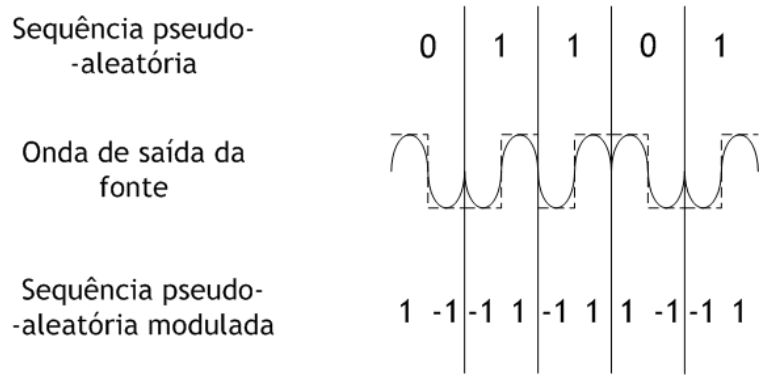



**Figura 6.4.3.1 – Processo de modulação de uma sequência pseudo-aleatória**

Tipicamente os sistemas acústicos têm circuitos emissores que permitem a transmissão de sinais sinusoidais cujo comando é feito através de FETs, ou outro tipo de transistores, controlados por microcontroladores. Para criar fisicamente o sinal pseudo--aleatório é necessário modular a sequência pseudo-aletória. Para tal é necessário passar da sequência pseudo-aleatória de banda base para a banda do sistema acústico, o que requer a modulação de uma portadora. Para modular a portadora é necessário gerar a sequência de comando dos FETs que controlam o circuito emissor do transdutor do sistema acústico.

No caso específico do VSA utilizado no laboratório de sistemas oceanográficos (*Oceansys)* basicamente o que é feito é emitir uma onda quadrada através do comando dos FETs (FETs on/off), comando este feito através de um microcontrolador, que comuta o circuito primário de um transformador para gerar no seu secundário um aumento de tensão à mesma frequência da onda quadrada no seu primário, sendo o transdutor responsável pela filtragem dessa onda. A onda filtrada aproxima-se da componente fundamental, pois o transdutor funciona como um filtro passa banda com um factor de qualidade (Q) elevado. O circuito da figura 6.4.3.2 exemplifica de modo detalhado o circuito de emissão acústica do VSA.

A sequência de comando dos FETs (Q1 e Q2 da figura 6.4.3.2) é formada a partir da sequência pseudo-aleatória inicial onde cada valor binário igual a 0 desta passa a ser 01 e cada valor binário igual a 1 passa a 10 formando uma sequência de comando dos FETs com o dobro do tamanho da sequência pseudo-aleatória.

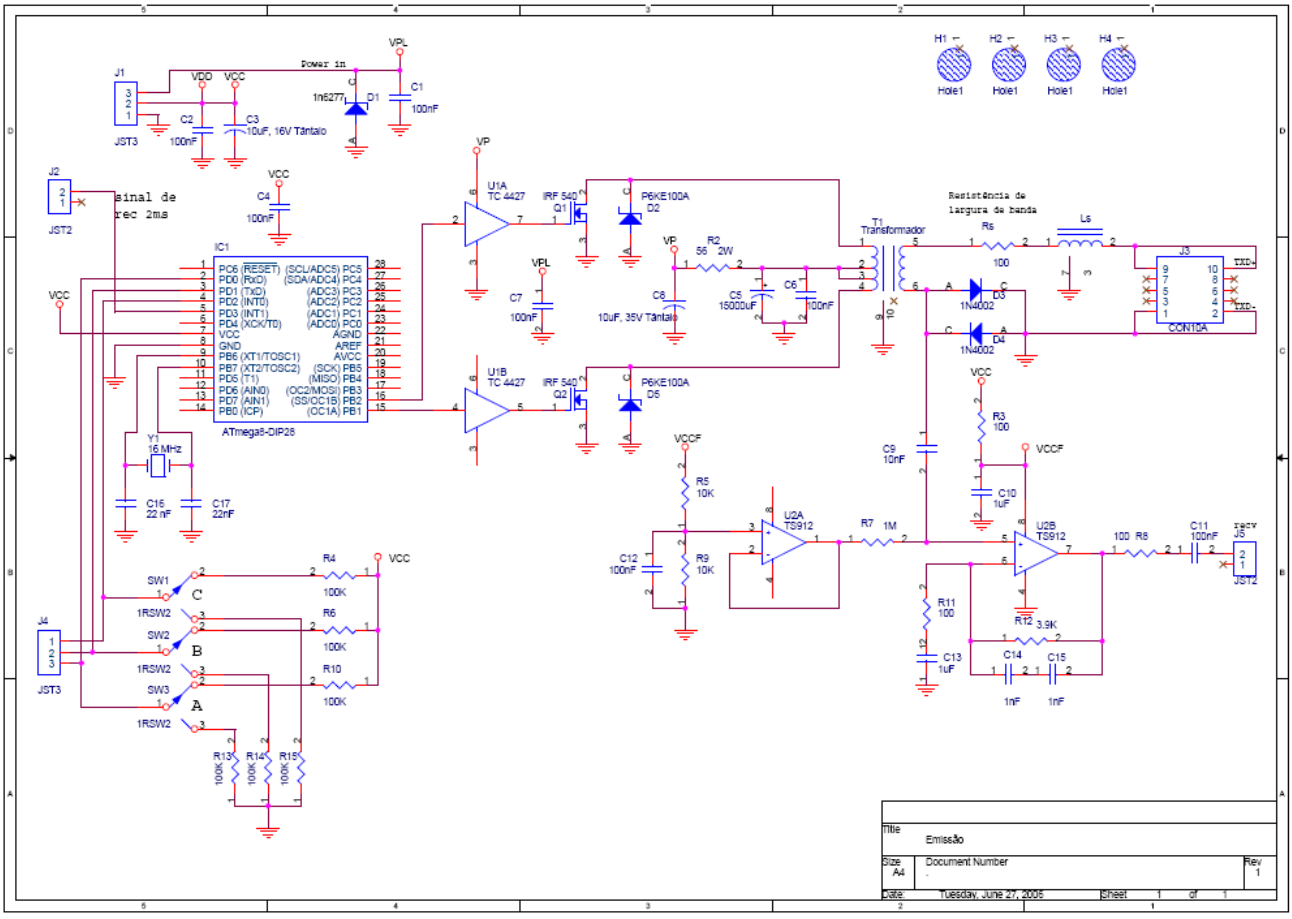



**Figura 6.4.3.2 – Circuito de emissão acústica**

A escolha de uma modulação de fase, deve-se ao facto de esta ser facilmente implementável. Estando esta facilidade relacionada com o drive dos FETs, ou seja, com a sequência de valores binários de 1s e -1s que pode ser directamente utilizada para criar o sinal modulado.

É necessário, no entanto, utilizar uma função de *treshold* na parte de detecção do sinal para termos uma onda quadrada na entrada do filtro que varie entre -1 e 1 de modo a que não existem valores ambiguos que sejam diferentes de -1 e 1, o que poderia deteriorar o funcionamento do filtro.

### 6.4.4 – Implementação do sistema em *hardware* reconfigurável

A implementação do conjunto de algoritmos descritos anteriormente foi executada através de tecnologia de *hardware* reconfigurável recorrendo à ferramenta de projecto *System Generator* e às capacidades de simulação, análise e visualização das ferramentas *Simulink* e MATLAB.

Após decidir que a escolha acertada era a utilização de *matched filters* para conseguir determinar o atraso das detecções de um sinal acústico entre os dois transdutores de modo óptimo, especificaram-se as características dos filtros para que correlacionassem dois sinais pseudo-aleatórios e simulou-se o seu comportamento.

Para obtermos uma resolução de 1º na estimação da direcção do VSA é necessário, no mínimo, utilizar uma frequência de amostragem de 429 kHz. Este valor é obtido utilizando as seguintes equações:

$$\Delta T \approx \frac{\overline{AB}}{c}\sin(\phi) \approx \frac{0.2}{1500}\sin(1^\circ) \qquad (6.4.4.1)$$

e

$$f_s = \frac{1}{\Delta T} \qquad (6.4.4.2)$$

Sendo $\phi$ o valor da resolução que pretendida e $f_s$ o valor da frequência de amostragem necessária para se obter a resolução pretendida. Para este sistema, os valores de $\overline{AB}$, $c$ e $\phi$ são de 0.2 m, 1500 m/s e 1º respectivamente. O valor da velocidade do som na água ($c$), é um valor aproximado pois admitiu-se que a gama operação, em termos de profundidade, do VSA não ultrapassaria os 40 m de profundidade. Observando a figura 2.3.1, pode-se verificar que até esta profundidade a velocidade do som na água pode ser aproximada por 1500 m/s. Atribuindo estes valores às diferentes variáveis obtém-se $\Delta T \approx$ 2.33(3) µs e $f_s \approx$ 429 kHz. Como os ADs, tipicamente, apenas permitem variar a sua frequência de amostragem em alguns múltiplos da sua frequência de amostragem máxima decidiu-se utilizar 500 kHz como frequência de amostragem do sistema. O que garante uma resolução aproximada de 0.86º, que é melhor do que a resolução pretendida.

Ao utilizarmos 500 kHz como frequência de amostragem estamos já a definir o número de coeficientes dos *matched filters* como sendo 500 coeficientes e o tamanho da sequência pseudo-aletória inicial como sendo de 250 elementos. Para criar esta sequência pseudo-aleatória, utilizou-se a função do MATLAB randn configurada para criar 250 valores aleatórios com média 1 e desvio padrão 2. Esta função, por si só, não cria um vector com valores binários entre 0 e 1. Para tal, criou-se um vector auxiliar que foi preenchido com valores binários entre 0 e 1 que dependem do facto de os valores do vector inicial serem maiores ou não do que zero. Sendo que, para valores maiores que zero é colocado um valor binário 1 e para valores menores que zero é colocado um valor binário 0. Após criação da sequência pseudo-aleatória com valores binários entre 0 e 1 é utilizado uma porção de código MATLAB para gerar a sequência de sinais *on/off* para a comutação dos FETs do circuito emissor a partir da sequência pseudo-aleatória. A partir da sequência de comando dos FETs é utilizada uma porção de código MATLAB para gerar os coeficientes do *matched filter*. Estes três vectores são guardados em três ficheiros de texto diferentes para serem utilizados sempre que é necessário reiniciar o sistema. Ao reiniciar o sistema os filtros são configurados automaticamente carregando os seus coeficientes a partir do ficheiro de texto respectivo. Os dois outros ficheiros são utilizados para configurar o comando dos FETs por parte do circuito de emissão do VSA podendo ser utilizados como *backup* pois pode-se recuperar os coeficientes a partir da sequência pseudo-aleatória ou do vector dos comandos dos FETs e vice versa, não sendo necessário actualizar os dois sistemas. Na parte de anexos deste relatório estão as porções de código que implementam todo este algoritmo de criação das sequências pseudo-aleatórias, de comando dos FETs e dos coeficientes dos filtros.

Após se verificar que o comportamento dos *matched filters* era o esperado implementaram-se dois filtros, um para cada transdutor, e todos com as mesmas características. Estes filtros foram implementados a partir de dois blocos DSP de nome DAFIR (*distributed arithmetic FIR)* que implementam filtros do tipo FIR através de aritmética distribuída. Os coeficientes dos filtros são carregados a partir de um ficheiro de texto, colocado no mesmo directório do modelo *Simulink* onde estão implementados, previamente configurado com os coeficientes que modulam o sinal pseudo-aleatório e configuram o bloco DAFIR de modo a este implementar uma correlação entre o sinal de entrada e o sinal pseudo-aleatório modulado. As duas figuras seguintes mostram como foi implementado o atraso nos filtros e mostram também como foram utilizados os filtros.

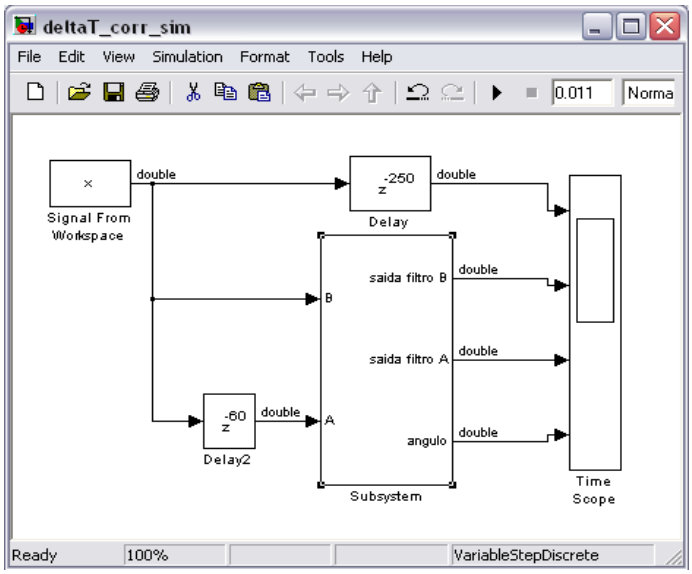



**Figura 6.4.4.1 – Sistema implementado**

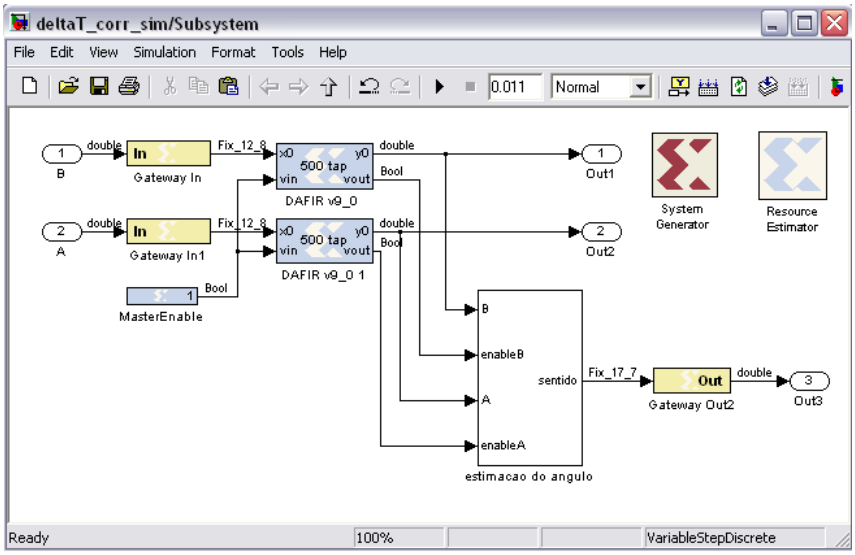



**Figura 6.4.4.2 – Implementação dos filtros**

A direcção é calculada através da estimação do atraso entre a saída dos dois filtros. Esse atraso é estimado contando o número de amostras entre o maior valor de saída da correlação de cada filtro através de um detector de valor máximo implementado recorrendo a blocos de lógica reconfigurável, tais como registos, ANDs, Ors, constantes, etc. As figuras seguintes mostram como foram implementados os detectores de máximo através do subsistema estimação do atraso. Na parte de anexos deste relatório está uma figura mais detalhada deste subsistema de estimação do atraso.

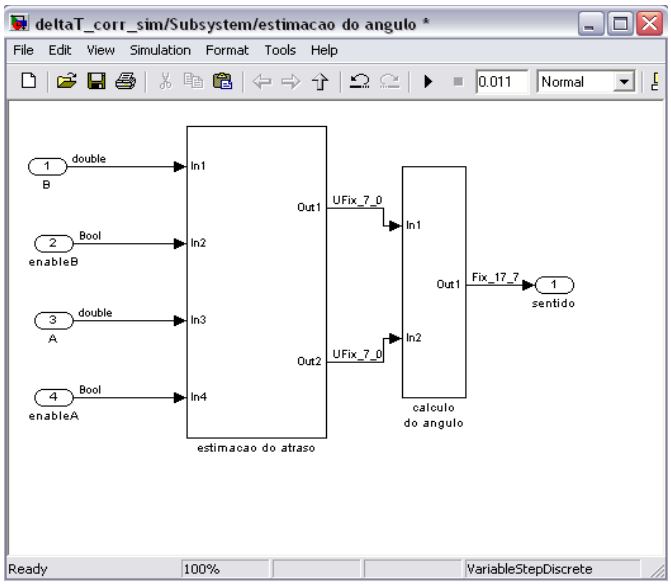



**Figura 6.4.4.3 – Estimação do ângulo**

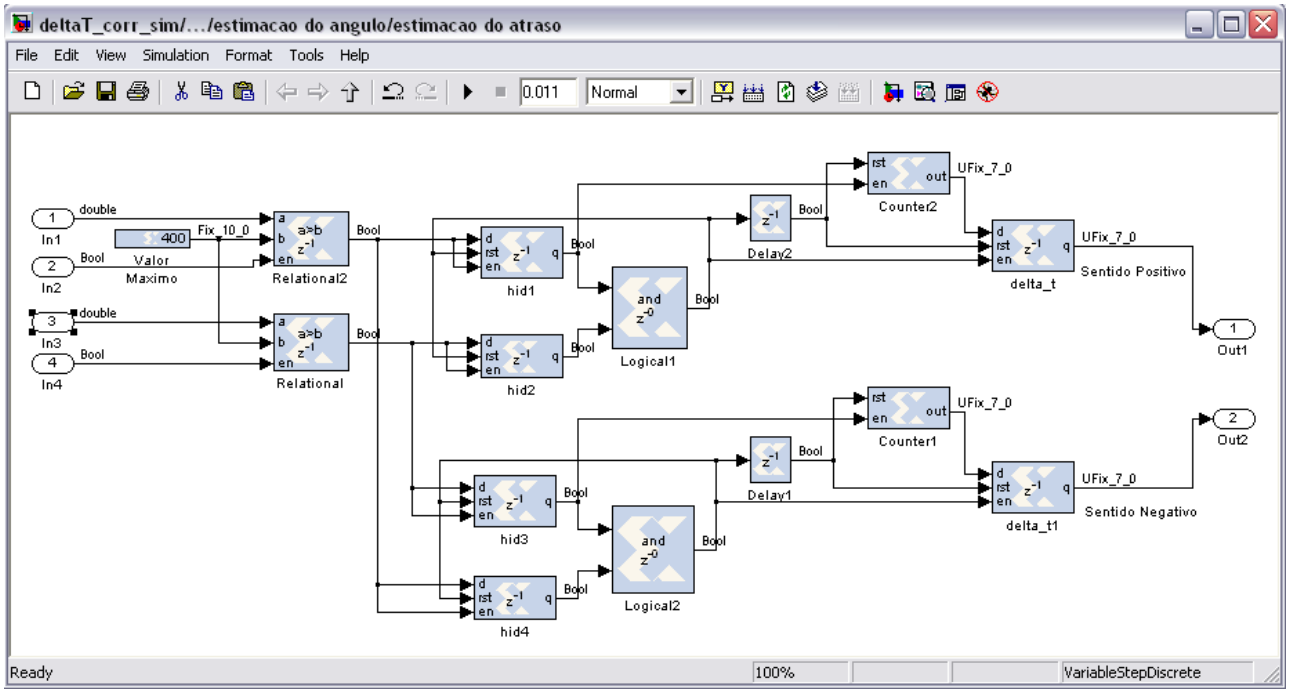



**Figura 6.4.4.4 – Estimação do atraso**

Os blocos *Relational* e *Relational2* determinam qual o filtro cuja saída atingiu um valor superior a 400 (nível máximo de detecção escolhido que pode ser utilizado para aumentar ou diminuir a sensibilidade do filtro) e faz o *enable* (através de um dos pares de registos hid) para que o respectivo *counter* inicie a contagem do número de amostras. Quando é detectado um valor da saída do segundo filtro superior a 400 o respectivo bloco AND (Logical1 ou Logical2), ligado a um par de registos hid que estão neste momento os dois ao nível lógico 1, faz *enable* ao respectivo registo (delta_t ou delta_t1) e o valor do *counter* é guardado neste registo sendo o *counter* no instante de *clock* seguinte desligado. Este valor do *counter* é passado ao subsistema cálculo do ângulo para ser utilizado como *index* do bloco ROM de modo a obter-se o valor do ângulo correspondente ao atraso entre a detecção do valor máximo da saída dos dois filtros.

Os detectores de valor máximo foram utilizados para obter uma estimativa do atraso entre os diferentes filtros, de modo a podermos obter o valor do números de amostras ( $n_s$ ) entre o valor máximo da saída de cada filtro. O valor de $\Delta T$ é obtido multiplicando $n_s$ pelo inverso da frequência de amostragem ( $f_s$ ). Como neste caso a frequência de amostragem é de 500 kHz obtemos:

$$\Delta T = n_s \frac{1}{f_s} = 2n_s \ (\mu s) \qquad (6.4.4.3)$$

Para a estimação do ângulo a partir do atraso da saída dos filtros é necessário proceder ao calculo do ângulo através de uma função arcsin. Visto que a ferramenta de projecto *System Generator* não possui nenhum bloco que faça esta operação foi necessário utilizar um bloco ROM com valores discretos do resultado da função arcsin entre 0 e 90º. Atendendo ao facto de o cálculo do ângulo ser através da equação:

$$\theta \approx \arcsin\left(\frac{c}{\overline{AB}} \frac{1}{f_s} n_s\right) = \arcsin(0.015\, n_s) \qquad (6.4.4.4)$$

Podemos concluir que o cáculo do ângulo depende apenas do número de amostras medidas entre o valor máximo da saída dos filtros. Como, de facto, o números de amostras é um número inteiro implica que a estimação do ângulo é obtida a partir de valores inteiros, logo é possível utilizar o valor de $n_s$ como *index* relativo da posição de memória de um bloco ROM que contém os valores da função arcsin para cada valor de $n_s$ entre 0 e 66 que oferece uma gama para $\theta$ de 0 a aproximadamente 82º. Este método de estimação do ângulo não nos permite estimar ângulos superiores a 82º porque para o *index* seguinte o ângulo seria maior que 90º e a geometria do nosso problema não o permite. No entanto, como à partida esta gama de valores para a estimação do ângulo não seria utilizada, pois estamos na zona fortemente não linear da função arcsin e a resolução neste zona da valores do ângulo é muito fraca, este facto não deteriora o funcionamento do global do sistema. A figura seguinte mostra como foram utilizados os blocos ROM no cálculo do valor do ângulo.

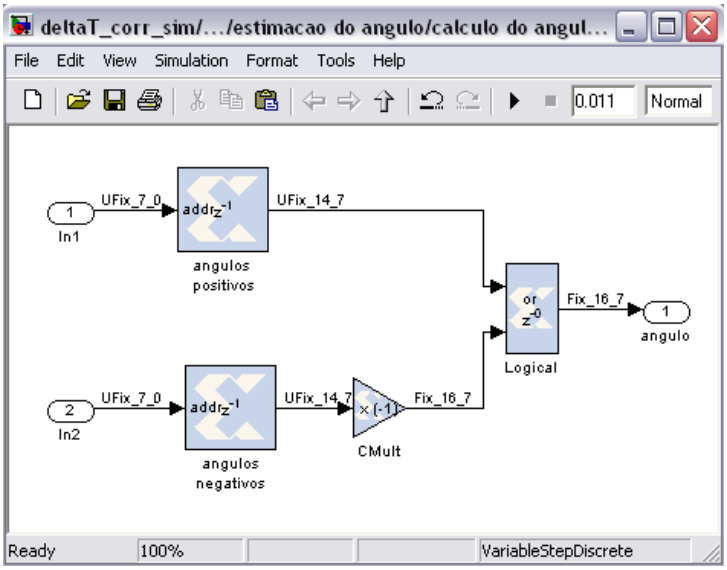



**Figura 6.4.4.5 – Cálculo do ângulo**

Capítulo 7

# Simulações

Neste capítulo serão apresentadas algumas das simulações executadas no decorrer da implementação do sistema. Estas simulações foram feitas recorrendo à ferramenta de projecto MATLAB numa primeira fase e depois recorrendo à ferramenta de projecto *Simulink* já utilizando blocos da ferramenta System Generator.

## 7.1 – Simulação dos *matched filters*

As simulações dos *matched filters* numa primeira fase foram feitas através da ferramenta de projecto MATLAB. As figuras seguintes simulam a correlação de dois sinais pseudo-aleatórios sendo a um deles acrescentado bastante ruído. A primerira figura mostra uma sequência pseudo-aleatória com valores binários entre -1 e 1. A segunda figura é a sequência pseudo-aleatória anterior com a adição de outra sequência pseudo-aleatória com valores que variam entre -5 e 5 de modo a simular a existência de ruído a que a onda recebida estará sujeita. Finalmente a terceira figura simula o resultado da correlação entre estas duas duas sequências pseudo-aletórias. Pode-se verificar pelo resultado da correlação entre as duas sequências pseudo-aleatórias que mesmos existindo bastante ruído numa das sequências pseudo-aleatórias é possível detectar a existência de correlação entre elas.

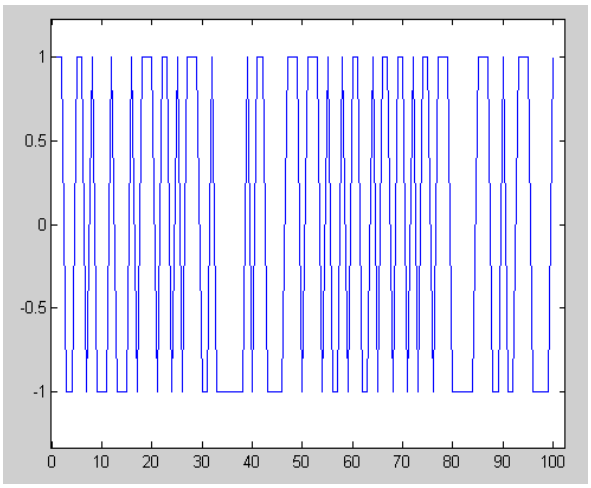



**Figura 7.1.1- Sequência pseudo-aleatória**

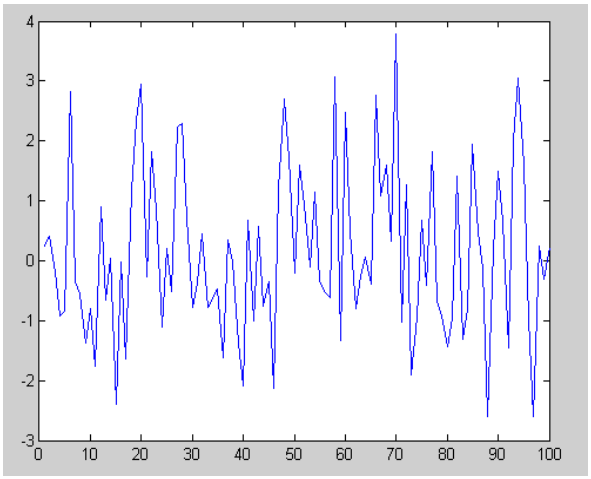



**Figura 7.1.2 – Sequência pseudo-aleatória com ruído**

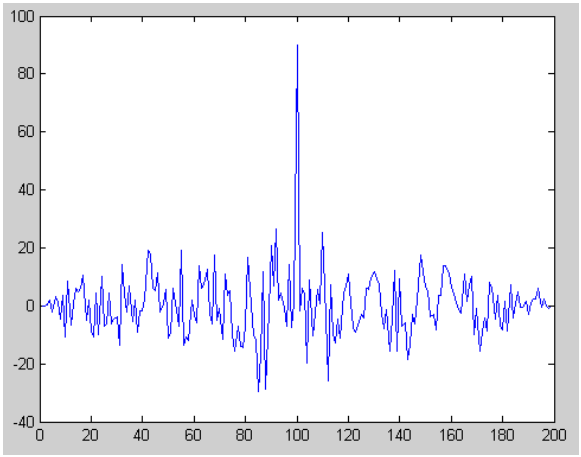



**Figura 7.1.3 – Correlação entre duas sequências pseudo-aleatórios**

O passo seguinte foi implementar os filtros através dos blocos da ferramenta de projecto *System Generator* e simulá-los através da ferramenta de projecto *Simulink*. As duas figuras seguintes mostram a implementação dos filtros, o sinal de entrada dos filtros e o atraso aplicado ao sinal de entrada do filtro A.

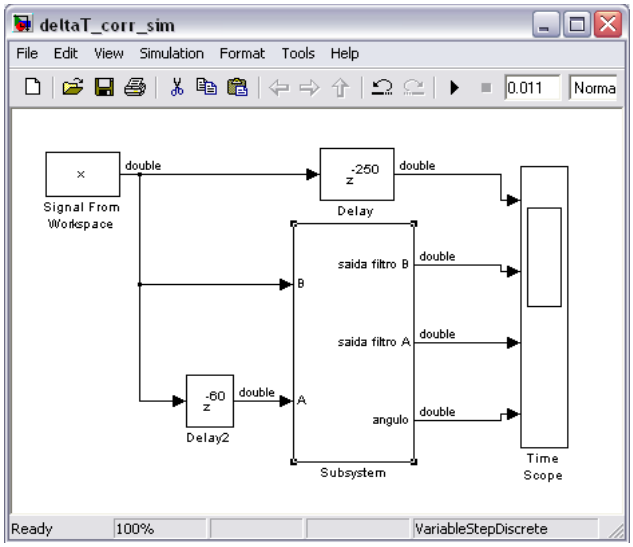



**Figura 7.1.4 – Simulação dos sinais de entrada com e sem atraso**

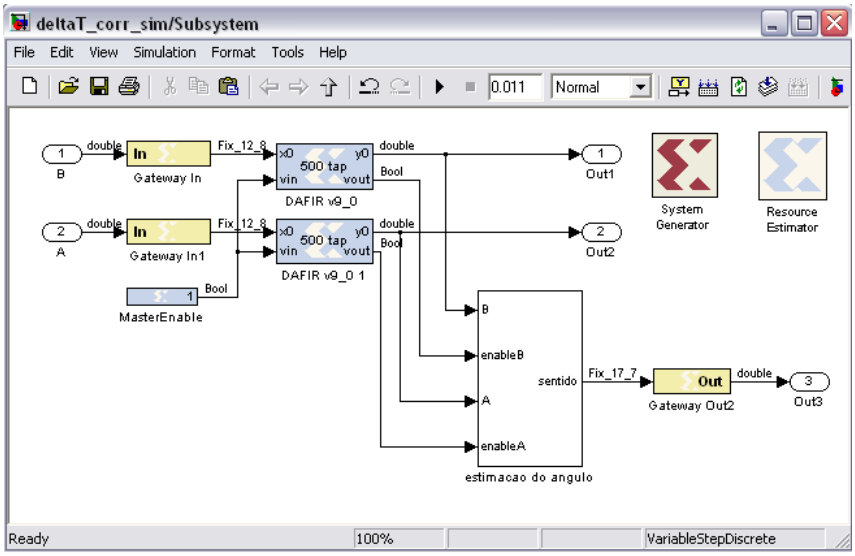



**Figura 7.1.5 – Simulação dos filtros**

A figura seguinte mostra os resultados obtidos nas saídas dos *matched filters*, recorrendo ao *Time Scope* da ferramenta *Simulink*. O sinal à entrada tem duas componentes como se pode verificar na primeira janela da figura 7.1.6. A primeira é uma sinal pseudo-aleatória com valores binários que variam entre -1 e 1 que corresponde a uma sequência pseudo-aleatória modulada e a segunda o sinal pseudo-aleatório com a adição de uma sequência pseudo-aletória que simula o sinal pseudo-aleatório detectado com ruído ganho na propagação do sinal num meio subaquático. A segunda janela é a saída do filtro B e a terceira janela é a saída do filtro A.

Pode-se verficar que, para além de existir correlação nas duas situações acima descritas verifica-se também que o valor máximo da saída dos dois filtros tem um atraso que é equivalente ao atraso provocado pelo bloco de *delay* colocado à entrada do filtro A de modo a simular o tempo que o sinal pseudo-aleatório demora a percorrer os dois trandutores.

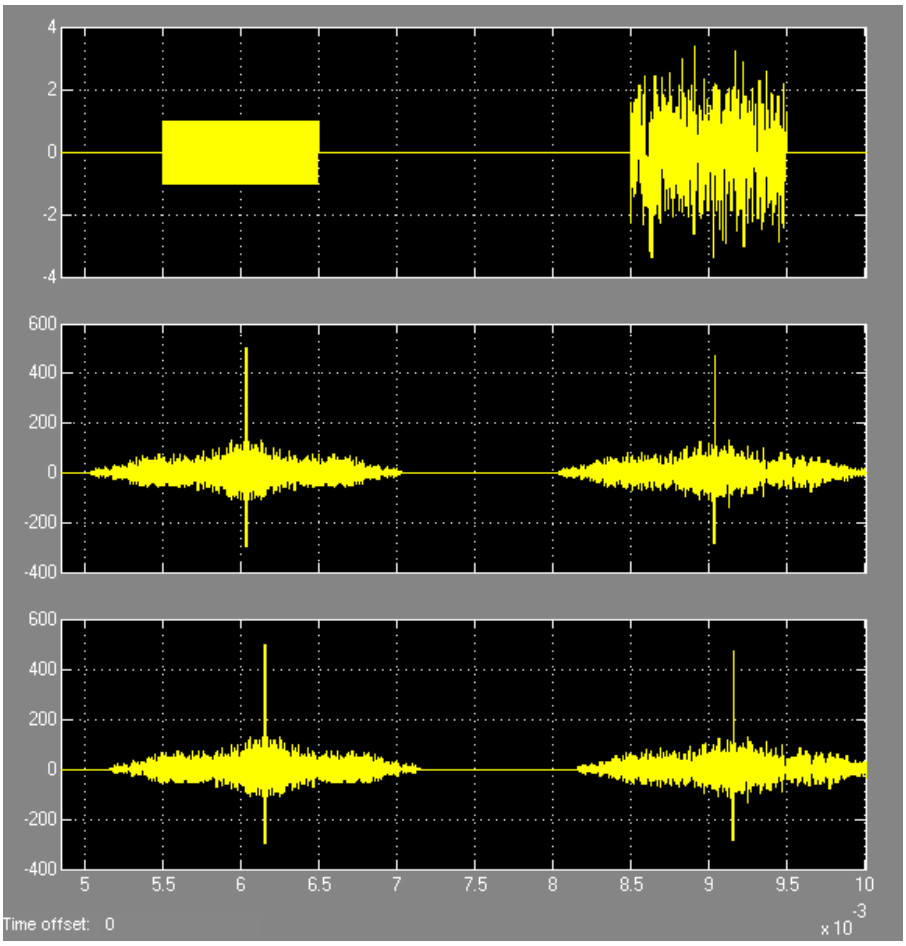



**Figura 7.1.6 – Resultados das simulações dos *matched filters***

## 7.2 - Simulação da estimação do atraso

A partir da figura 7.1.5 verifica-se que o atraso aplicado ao sinal de entrada do filtro A é de 60 amostras, sendo este atraso utilizado para simular o tempo que o sinal pseudo-aleatório demora a percorrer os dois trandutores. Para se verificar se o valor do atraso era o esperado adicionou-se a saída do subsistema estimação do atraso à saída do sistema estimação do ângulo e conectou-se essa saída um *Time Scope* da ferramenta *Simulink*, como se pode ver nas duas figuras seguintes.

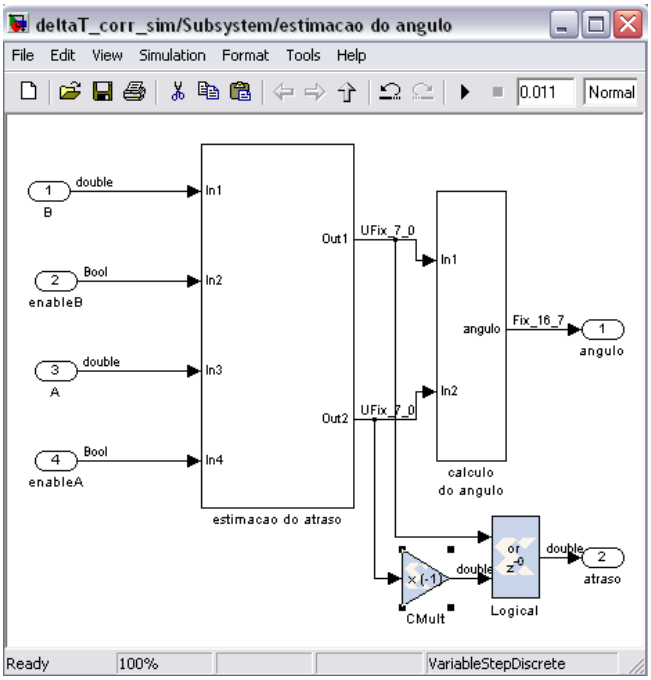



**Figura 7.2.1 – Simulação do atraso**

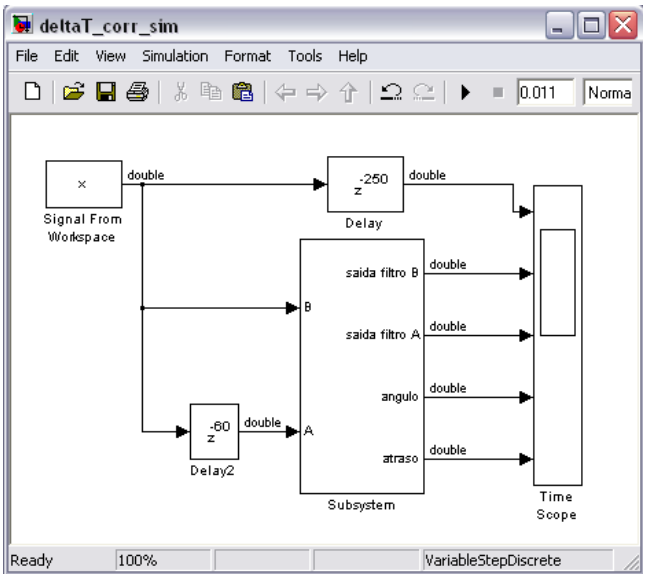



**Figura 7.2.2 – Simulação dos sinais com e sem atraso**

Analisando os resultados obtidos através do *Time Scope* (figura 7.2.3) podemos verificar que o resultado obtido, através do subsistema estimação do atraso, é de 60 amostras. Este valor corresponde ao atraso aplicado ao sinal de entrada do filtro A e pode ser verificado analisando a última janela da figura 7.2.3. A primeira janela da figura 7.2.3 mostra o sinal de entrada enquanto que a segunda e terceira janelas mostram as saídas dos *matched filters* B e A respectivamente. Sendo as características dos sinais as mesmas que foram referidas no capítulo anterior.

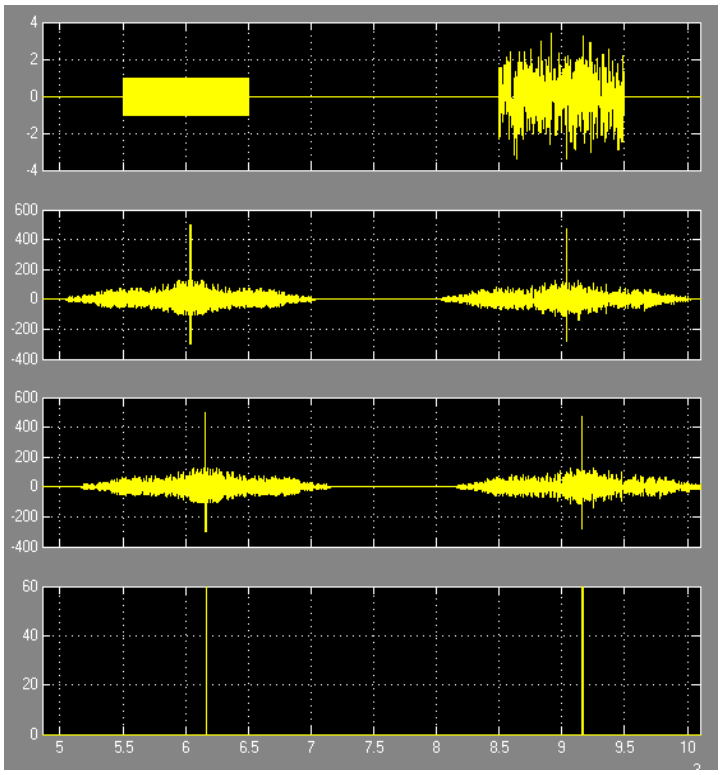



**Figura 7.2.3 – Resultados da simulação do atraso**

## 7.3 - Simulação do cálculo do ângulo

Após obtermos um valor para o atraso entre a detecção do valor máximo da saída nos dois filtros, este valor é utilizado como *index* relativo da posição de memória dos blocos ROM, representados na figura seguinte, que contêm os valores do ângulo cujo argumento da função arcsin varia conforme o valor desse *index*. Estes dois blocos ROM têm guardados valores que foram importados a partir de um ficheiro de texto com os valores da função arcsin para cada valor do *index* entre 0 e 66. Na parte de anexos deste relatório está descrita uma lista com os *indexs* e respectivos valores da função arcsin relativos a cada *index* que foram importados para os blocos ROM.

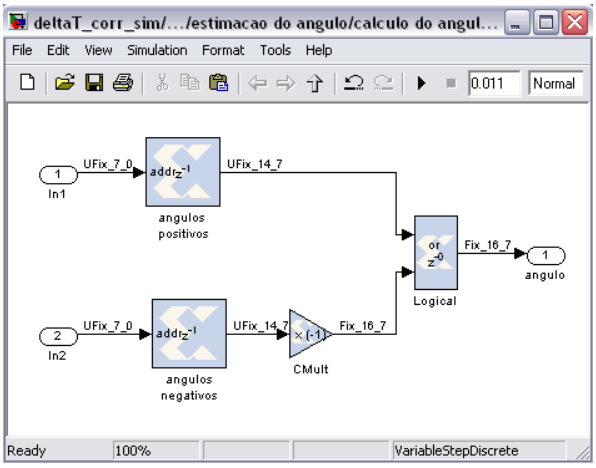



**Figura 7.3.1 – Simulação do ângulo**

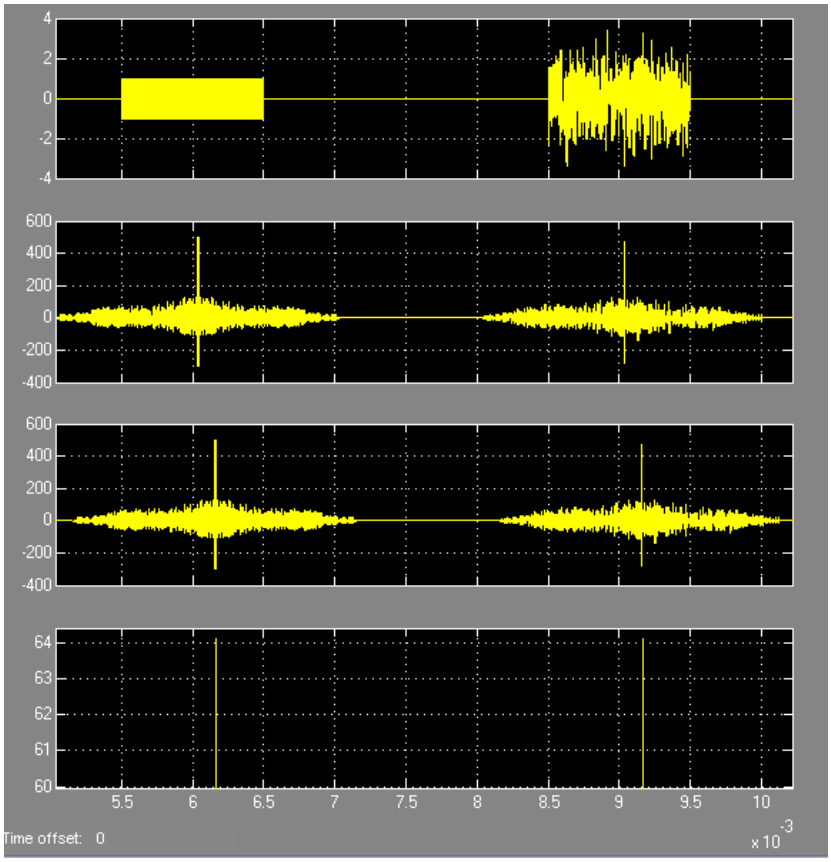



**Figura 7.3.2 – Resultados da simulação do ângulo**

A figura 7.3.2 mostra uma estimativa do valor do ângulo para um atraso entre a detecção do valor máximo da saída dos dois filtros correspondente a 60 amostras. Sendo neste caso o valor do ângulo $\theta$ de aproximadamente 64º.

Capítulo 8

# Conclusões e trabalho futuro

Uma das vantagem da utilização de tecnologia de *hardware* reconfigurável baseada em FPGAs para implementação deste sistema deveu-se ao facto de ser altamente paralelizável permitindo implementar vários subsistemas em que todos funcionam paralelamente. Por exemplo, todos os subsistemas nesta dissertação funcionam todos em paralelo. A partir do momento em que é detectado um máximo à saída de um dos filtros o subsistema de estimação do atraso começa a contar o número de amostras entre o valor máximo de saída do outro filtro e após se obter o valor do atraso é cálculado o valor do ângulo. Estas operações acontecem, mesmo antes de os filtros finalizarem a totalidade da correlação entre sinais. O facto de se poder utilizar todas as potencialidades das ferramentas de projecto MATLAB e *Simulink* é outra vantagem, pois permite projectar, simular e modificar facilmente as características dos filtros.

Uma das desvantagens desta tecnologia é o facto de muitas das possibilidades oferecidas pelas ferramentas de projecto não poderem ser realizáveis por falta de espaço em termos de *slices*. Sendo a ferramenta de projecto *System Generator* bastante recente existem algumas operações que ainda não estão implementadas. Esta ferramenta tem a possibilidade de calcular funções como senos, cosenos, tangentes e cotangentes mas, por incrível que pareça, ainda não possibilita o cálculo de funções do tipo arcsin ou arccos.

Para além da implementação e teste do *hardware* relativo à interligação dos conversores A/D com a *board* FPGA de modo a que os sinais obtidos através dos transdutores possam ser digitalizados e processados, apresentam-se ainda algumas sugestões para continuação do trabalho. Como primeira e segunda sugestões ficam a interligação com o sistema computacional principal dos veículos ou das bóias para o envio dos dados obtidos, através da porta série ou de um protocolo IP e a configuração dos coeficientes do *mactched filter* para a resposta do circuito emissor à sequência de ataque aos FETs. Como terceira e última sugestão para um futuro trabalho fica a utilização de outros sinais e técnicas de modulação tais como modulação de frequência, utilização de *chirps,* etc.

# Anexos

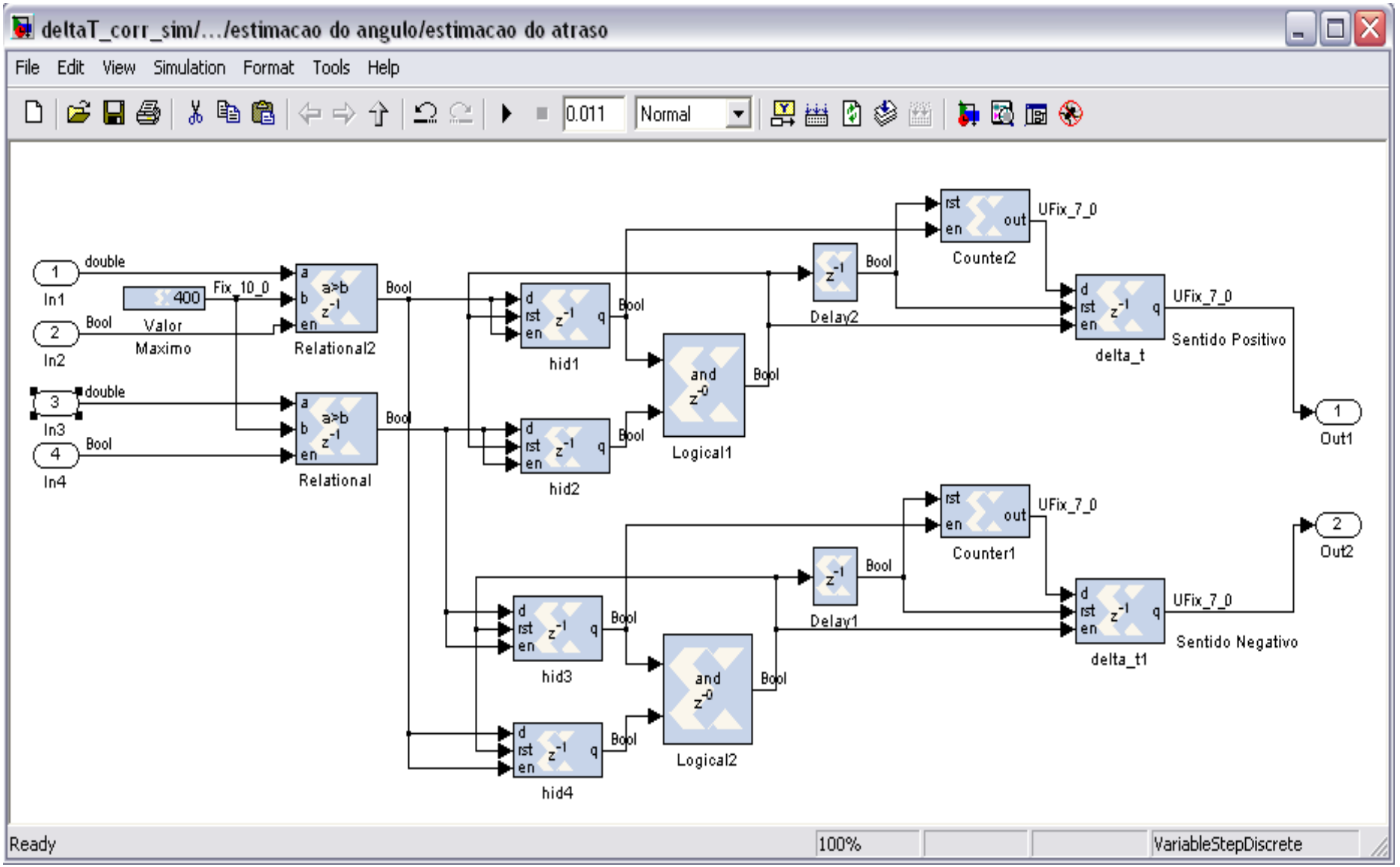
```
% cálculo dos coeficientes do matched filter para detecção de uma sequência
% pseudo aleatória que modula a fase de uma portadora
% parâmetos de entrada:
% b - sequência pseudo aleatória  de comprimento N
% valores de saída
% c - sequência de sinais de on-off para a comutação dos transistores do
%      circuito emissor (esta sequência tem comprimento 2N)
% tcoefs - coeficientes do matched filter



b_aux=1+2.*randn(250,1);
b=b_aux>0;
N=length(b);
c=zeros(2*N,1);
t=zeros(2*N,1);
for ii=0:N-1
   if b(ii+1)==1
      c(2*ii+1) = 1;
      c(2*ii+2) = 0;
   else
      c(2*ii+1) = 0;
      c(2*ii+2) = 1;
   end
end

for ij=0:2:2*N-2
   if (c(ij+1)==1) && (c(ij+2)==0)
      t(ij+1) = -1;
      t(ij+2) = 1;
   else
      t(ij+1) = 1;
      t(ij+2) = -1;
   end
end

tcoefs=t(end:-1:1);
```


deltaT_corr_sim/.../estimacao do angulo/estimacao do atraso
File Edit View Simulation Format Tools Help
0.011
Normal
1
In1
double
2
In2
Bool
400
Valor
Maximo
Fix_10_0
3
In3
double
4
In4
Bool
a
b
a>b
en
Relational2
Relational
Bool
d
rst
en
hid1
hid2
hid3
hid4
and
Logical1
Logical2
Delay2
Delay1
rst
en
out
Counter2
Counter1
UFix_7_0
delta_t
delta_t1
Sentido Positivo
Sentido Negativo
1
Out1
2
Out2
Ready
100%
VariableStepDiscrete

## Lista de *indexs* e respectivos valores da função arcsin

Columns 1 through 10

0 0.8595 1.7191 2.5792 3.4398 4.3012 5.1636 6.0272 6.8921 7.7586

Columns 11 through 20

8.6269 9.4972 10.3698 11.2447 12.1224 13.0029 13.8865 14.7736 15.6643 16.5588

Columns 21 through 30

17.4576 18.3608 19.2688 20.1818 21.1002 22.0243 22.9545 23.8911 24.8346 25.7853

Columns 31 through 40

26.7437 27.7102 28.6854 29.6698 30.6638 31.6682 32.6836 33.7107 34.7502 35.8030

Columns 41 through 50

36.8699 37.9519 39.0501 40.1657 41.2999 42.4542 43.6301 44.8295 46.0545 47.3072

Columns 51 through 60

48.5904 49.9070 51.2606 52.6553 54.0959 55.5885 57.1401 58.7597 60.4586 62.2515

Columns 61 through 67

**64.1581** 66.2057 68.4348 70.9089 73.7398 77.1614 81.8904

# Referências


[1] Paul C. Etter, *Underwater Acoustic Modelling and Simulation, 3rd Edition*, Taylor & Francis, 2003.

[2] http://www.dosits.org/science/sndmoves/3c.htm

[3] Nicholas Paul Fofonoff, *Physical properties of seawater: A new salinity scale and equation of state for seawater*, 1985.

[4] John Ralph Apel, *Principles of ocean physics*. In:, Academic Press, London (1987).

[5] Leonid Maksimovich Brekhovskikh, *Fundamentals of Oceans Acoustics, 3rd Edition,* Springer, 2003.

[6] Malcolm J. Crocker, *Handbook of Acoustics,* Wiley-IEEE, 1998.

[7] http://www.sonardyne.com/Support/PositioningTechniques/index.html

[8] L. A. Madureira, *Sistema de Navegação Acústica para Múltiplos Veículos Subaquáticos,* Tese de Mestrado, Faculdade e Engenharia Universidade do Porto, Dezembro 2004

[9] Keith Vickery, *Acoustic Positioning Systems: A Practical Overview of Current Systems,* Sonardyne, Inc (1998).

[10] http://www.sonardyne.co.uk/Products/PositioningNavigation/systems/wideband_lusbl.html

[11] http://pt.wikipedia.org/wiki/Jean-Baptiste_Joseph_Fourier

[12] http://en.wikipedia.org/wiki/Dirichlet_conditions

[13] Francisco José de Oliveira Restivo, *Processamento Digital de Sinal - Tópicos,* Ano Lectivo 1998/99, Faculdade de Engenharia da Universidade do Porto, Dezembro de 1998

[14] Sanjit K. Mitra, *Digital signal processing: a computer-based approach*, McGraw-Hill, 1998.

[15] Digital Signal Processing *A practical Approach,* Emmanuel C. Ifeachor, Barrie W. Jervis.

[16] Virtex™ 2.5 V Field Programmable Gate Arrays product specification.

[17] EDK Concepts, Tools, and Techniques - A Hands-On Guide to Effective Embedded System.