Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2013

Flávio Carlos da Silva Rodrigues Silvestre

Detetor Digital Síncrono de Sinal CW Baseado em FPGA

Flávio Carlos da Silva Rodrigues Silvestre

Detetor Digital Síncrono de Sinal CW Baseado em FPGA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Armando Carlos Domingues da Rocha e do Doutor Arnaldo Silva Rodrigues de Oliveira, Professores do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president

Professor Doutor Tomás António Mendes Oliveira e Silva
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Professor José Carlos dos Santos Alves
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Armando Carlos Domingues da Rocha
Professor Auxiliar da Universidade de Aveiro (Orientador)

Professor Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar da Universidade de Aveiro (Co-orientador)

Palavras-chave

Recetor Digital, Propagação de Microondas, Medição de Balizas, Comunicações por Satélite, Rádio definido por Software, Phase Locked Loop.

Resumo

A propagação de sinais acima de 10 GHz na atmosfera sofre de atenuação e despolarização devido à presença de hidrometeoros. A caracterização e o modelamento destas contrariedades exige longas campanhas de medição as quais são efectuados usando receptores dedicados que monitorizam o sinal CW de elevada pureza espectral radiado por satélites geo-estacionário.

Os sinais a medir são habitualmente dois, o denominado copolar (recebido com polarização original) e o crosspolar (polarização ortogonal). A medição da amplitude permite obter a magnitude de atenuação atmosférica, a qual é importante para saber durante quanto tempo do ano uma ligação de satélite estará indisponível.

A medição do tipo de sinais aqui pretendido é sempre problemático pois o sinal recebido apresenta um valor de CNR muito reduzido, algum espalhamento espectral e deriva de frequência. Os métodos de deteção coerente por PLL ou FLL são preferidos neste tipo de aplicações e apresentam uma gama dinâmica superior mas por outro lado a sua implementação no domínio analógico pode ser bastante díficil.

De modo a contornar a inflexibilidade dos métodos analógicos recorreu-se ao conceito de Software Defined Radio (SDR) que procura tratar os sinais de rádio, tanto quanto possível, no domínio digital. Deste modo resultou a ideia para o trabalho desta dissertação, que consistiu na elaboração de um pequeno módulo capaz de detectar digitalmente o sinal proveniente de satélite a uma frequência intermédia (IF) e com recurso a uma FPGA, adicionalmente os resultados da detecção são enviados para um computador hospedeiro.

Com vista à conceptualização do projecto é feita uma caracterização do sinal a detectar e são apresentados alguns métodos de detecção dando-se enfâse aos que fazem uso de malhas de sincronização. Seguidamente é feita uma descrição da placa de conversão analógico-digital construída e da arquitectura do detector síncrono realizado na FPGA: etapas de filtragem, decimação e especificação dos parâmetros de loop. É ainda descrito o processo de transferência de dados para o PC via controlador USB.

Por fim alguns resultados utilizando um gerador RF para sintetizar o sinal de entrada são apresentados e analisados. Foi concluído que o detector adquire o sincronismo para sinais acima de -60 dBm, o que apesar de ser uma situação onde o ruído é muito reduzido, deixa boas indicações para testes efetuados em ambiente real.

Keywords

Digital Receiver, Characterization of Earth-Satellite propagation channel, CW Signal, Satellite Communications, Software Defined Radio, Phase Locked Loop.

Abstract

Earth-Satellite propagation channel modeling requires extensive measurement campaigns at as many sites, frequencies and link parameters as possible. The campaigns are made by estimating the received amplitude of a beacon CW signal radiated from a satellite using beacon receivers.

The receiver measures the signal with the same polarization as the radiated one (copolar) and often the orthogonal (crosspolar) that carries additional information on the channel. As the carrier to noise ratio (CNR) is low a coherent detection is often used and requires the use of tracking loops whose implementation in the analogue domain is harsh.

The use of digital radio hardware and/or software defined radio techniques to handle the tracking and detection is now being used: the down conversion is made within a FPGA, processed, and the data is moved to a host computer for further software processing.

A stand-alone detector unit combining the frequency down conversion and detection in a single board and delivering the Cartesian two channel detected amplitudes to a host PC is described. The advantages would be a general purpose board that could handle, with no more hardware or software developments, measurement of beacon signals in a wide IF input range making also easier the analogue design of the preceding beacon receiver hardware.

Some preliminar results of the developed detector with a clean signal provided by a waveform generator are presented and analysed. It was concluded that the detector can achieve lock with signals as low as -60 dBm in absense of phase noise, which leaves good indications to a real world detection scenario.

The dissertation ends with some ideas for improvement and future work such as the utilization of frequency locked loops.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ADC	Analog to Digital Converter
AFC	Automatic Frequency Control
AGC	Automatic Gain Control
API	Application Programmable Interface
AWGN	Additive White Gaussian Noise
B2B	Board to Board
CIC	Cascaded Integrator Comb
CLBs	Configurable Logic Blocks
CMOS	Complementary Metal-Oxide-Semiconductor
CNR	Carrier to Noise Ratio
CPU	Central Processing Unit
CW	Continuous Waveform
DC	Direct Current
DCM	Digital Clock Manager
DDR	Double Data Register
DDS	Direct Digital Synthesizer
DRSP	Digital Receiver Signal Processor
DSP	Digital Signal Processor
EEPROM	Erasable Programmable Read-Only Memory
EIRP	Equivalent Isotropically Radiated Power
ESA	European Space Agency
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FLL	Frequency Locked Loop

FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPIF	General Programmable InterFace
HDL	Hardware Description Language
I/O	Input/Output
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
IIR	Infinite Impulse Response
IOB	Input/Output Blocks
IP	Intellectual property
ITU	International Telecommunication Union
LED	Light Emitting Diode
LNA	Low Noise Amplifier
LSB	Least Significant Bit
LUT	Lookup Table
LVDS	Low Voltage Differential Signaling
MSB	Most Bignificant Bit
NCO	Numerically Controlled oscillator
NF	Noise Factor
OMT	OrthoMode Transducer
PCB	Printed Circuit Board
PLL	Phase Locked Loop
RF	Radio Frequency
RFD	Ready For Data
ROM	Read Only Memory
SDR	Software Defined Radio
SFDR	Spurious Free Dynamic Range
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface Bus
TDP	Technology Demonstration Payload
TTL	Transistor Transistor logic
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VCO	Voltage Controlled Oscillator
XPD	Cross Polarization Discrimination

Capítulo 1

Introdução

1.1 Enquadramento

Desde o lançamento do primeiro satélite artificial, o *Sputnik 1* em 1957, as comunicações por satélite têm sofrido uma grande evolução a diferentes níveis tecnológicos. Desde os modestos 20 MHz e 40 MHz utilizados como portadoras nos *transceivers* do primeiro satélite, que permitiam larguras de banda da ordem dos kHz, que se tem assistido a um aumento das velocidades de transmissão, bandas de frequência utilizadas e da diversidade de serviços fornecidos. Hoje em dia larguras de banda da ordem dos 500 MHz na banda Ka (20 GHz) estão a ser exploradas para fins comerciais e o interesse começa estender-se para a banda Q.

Por outro lado a propagação de sinais acima de 10 GHz na atmosfera sofre os seguintes efeitos:

- Atenuação (absorção) causada por água líquida (chuva, nuvens, nevoeiro) e vapor de água;
- Despolarização causada por chuva e nuvens de gelo;
- Cintilação causada por turbulência atmosférica devido à variabilidade temporal e espacial do índice de refracção.

Todos os efeitos anteriores, excepto a atenuação por vapor de água, aumentam com a frequência (até pelo menos 90 GHz). A caracterização e a modelação destas contrariedades exige longas campanhas de medição. Estas por sua vez são efectuadas usando receptores dedicados que monitorizam o sinal *Continuous Waveform (CW)*, de elevada pureza espectral e estabilidade radiado por satélites geo-estacionário.

Os sinais a medir são habitualmente dois: o denominado *copolar* (recebido com uma polarização original) e o *crosspolar* (recebido com a polarização ortogonal). A medição da amplitude do primeiro permite obter a magnitude da atenuação atmosférica a qual é importante para saber durante quanto tempo do ano uma ligação de satélite estará indisponível. O segundo tem bastante menos potência e indica a despolarização introduzida pela atmosfera. A relação das amplitudes do canal *copolar* e *crosspolar* (em dB) é denominada de discriminação da polarização cruzada *Cross Polarization Discrimination (XPD)*.

1.2 Motivação

A implementação de uma margem de atenuação para garantir a tradicional disponibilidade de serviço de 99.99% torna-se impraticável na banda Ka (e superiores) sem técnicas de correcção adequadas. No entanto é possível recorrer a diversas técnicas que podem contribuir para minorar

Frequency Band	Uplink Frequency GHz	Downlink Frequency GHz	Major Applications	Bandwidth
UHF Band (military)	0.292 - 0.312	0.250 - 0.270	Military applications	20 MHz
C Band (commercial)	5.9 - 6.4	3.7 - 4.2	Television Broadcast	500 MHz
X Band (military)	7.25 - 7.75	3.7 - 4.2	Mobile (ships, aircrafts)	500 MHz
Ku-Band (commercial)	14 - 14.5	11.7 - 12.2	Broadcast and fixed point service	500 MHz
Ka-Band (commercial)	27 - 30	17 - 20	Broadcast	3 GHz
Ka-Band (military)	20 - 21	11.7 - 12.2	Military	1 GHz

Tabela 1.1: Bandas de Frequência utilizadas para serviços de Satélite. Adaptado de [BC07].

os efeitos de atenuação (*Fade Counter-measuremente methods*). A panóplia de métodos é grande e engloba o controlo de potência, a redução da taxa de transmissão, etc. A obtenção de dados experimentais relativamente ao sinal recebido na superfície terrestre é fundamental para o desenvolvimento de modelos para vários parâmetros do canal físico.

Para meados de 2013 está previsto o lançamento do satélite Alphasat I-XL que fornecerá diversos *Technology Demonstration Payload (TDP)* cujo desenvolvimento está por conta da agência espacial europeia (ESA). Um desses *Payloads* consistirá na emissão de sinal *CW* nas bandas Ka e Q para efeitos de recolha de dados e avaliação do meio de propagação.

No que toca à implementação dos detetores, o recente desenvolvimento dos sistemas de rádio digitais trouxe uma nova janela de oportunidades e um primeiro detetor usando *chips* discretos foi apresentado em [JCo98]. O detetor consistia em *chips Digital Receiver Signal Processor (DRSP)* que efectuavam a conversão e decimação dos sinais amostrados, os quais eram de seguida transferidos para um kit Digital Signal Processor (DSP). O método de detecção baseava-se numa estimativa espectral do sinal.

Os desenvolvimentos seguintes usaram uma abordagem semelhante contudo introduziu-se uma *Phase Locked Loop (PLL)* por software reprogramando os osciladores digitais nos *chips* DRSP [Pir07] [Sou07]. Outro tipo de loops foram superficialmente testados para esta aplicação.

1.3 Objetivos

Face ao exposto o objectivo deste trabalho é a implementação de hardware que permita a medição de sinal CW a partir de uma frequência intermédia (IF) convertida previamente por um andar de conversão de frequência analógico. De entre os requisitos desejáveis do sistema é possível destacar:

- Medição da amplitude e fase relativa dos canais *copolar* e *crosspolar*;

- *Tracking* da frequência central do sinal recebido;
- Rejeição de ruído fora da banda do sinal.

A medição do tipo de sinais aqui pretendido é sempre problemático pois o sinal recebido é invariavelmente fraco ou seja a sua *Carrier to Noise Ratio (CNR)* (dBHz) raramente excede os 55 dB em recetores com antenas até cerca de 1 a 1.5 metros de diâmetro. Outro problema é o espalhamento espectral do sinal (ruído de fase) que leva à necessidade de usar uma largura de banda não nula para ser possível efectuar a sua detecção com a máxima gama dinâmica. A gama dinâmica de medida da atenuação e despolarização é sempre limitada e aproveitar o melhor possível os limites teóricos nas medidas é sempre importante.

Face ao exposto, os objetivos do trabalho resumem-se a:

- Análise e apresentação de várias técnicas de detecção de sinal CW;
- Desenvolvimento de um sistema completamente integrado com *Analog to Digital Converter (ADC)* e uma *Field Programmable Gate Array (FPGA)*, com a capacidade de autonomamente implementar as técnicas de detecção mais comuns e de enviar os resultados às taxas necessárias para o tipo de estudos em vista.

Esta ideia, em termos de hardware, partiu dos kits *Software Defined Radio (SDR)* de uso comum, com a única diferença a residir no facto de o processamento de sinal ser também efetuado dentro da FPGA. A razão para o desenvolvimento desta solução mais especializada prende-se com a pouca flexibilidade de configuração destes kits: um sistema de detecção utilizando o kit *Universal Software Radio Peripheral (USRP)* da *Ettus Research* foi efectuado em [Sou11] contudo constatou-se que era muito complicado utilizar o mesmo *Numerically Controlled oscillator (NCO)* para detetar mais que um canal dentro da FPGA. Para permitir o anterior seria necessário ter acesso e modificar o código *Hardware Description Language (HDL)* dos kits assim como efetuar a reprogramação do mesmo.

Apenas uma solução similar a esta terá sido desenvolvida por uma firma britânica [1] que contudo usava *chips* dedicados à implementação eficiente de FFTs. Não se conhecem desenvolvimentos posteriores.

1.4 Estrutura da dissertação

Este documento encontra-se organizado em seis capítulos. O capítulo atual visa fazer um enquadramento nos sistemas de telecomunicações por satélite, realçar a importância da obtenção de dados experimentais para caracterização física do canal atmosférico e apresentar também os objetivos propostos para a atual dissertação.

- Capítulo 2: É efetuada a caracterização do sinal a detetar e são apresentados vários métodos de deteção de sinal CW, dando-se particular relevância a deteção síncrona usando malhas de sincronização. São apresentados também resultados da simulação efetuada em *Simulink* relativamente a uma PLL digital.
- Capítulo 3: Descrição do sistema físico implementado para a digitalização da IF e que realiza a interface com a FPGA e com o PC anfitrião.
- Capítulo 4: Arquitectura do detetor síncrono implementado em FPGA, sinais de interface com a placa de conversão analógico-digital, blocos instanciados, síntese de filtros digitais e

[1] Signal Processors Limited

dimensionamento da PLL digital.

- Capítulo 5: Comunicação via *Universal Serial Bus (USB)* com o PC anfitrião. É apresentada a estrutura básica do controlador USB e o seu modo de funcionamento, de seguida são apresentados os mecanismos que foram construídos, tanto ao nível da FPGA como do computador hospedeiro, de modo a possibilitar a transferência de informação.

- Capítulo 6: Descrição dos testes realizados ao sistema e análise dos métodos de deteção implementados, domínio da frequência com recurso a Fast Fourier Transform (FFT) complexa e com PLL digital.

- Capítulo 7: Conclusões finais e propostas de trabalho futuro.

- Anexo A: Esquemático, diagrama de blocos e opções de configuração da placa de conversão analógico-digital.

- Anexo B: Diagrama de blocos do sistema implementado em FPGA, ocupação de recursos e configurações utilizadas nos *IP cores* da Xilinx.

- Anexo C: Procedimento para o setup da placa TE0320: instalação de drivers do controlador USB FX2 e programação da PROM.

- Anexo D: Instruções para o desenvolvimento de software utilizando a framework Nokia QT e os drivers do controlador USB FX2 em ambiente Microsoft Visual Studio 2010.

- Anexo E: Descrição da aplicação desenvolvida para interface com a FPGA e recolha de dados.

Capítulo 2

Conceitos Fundamentais

2.1 Introdução

Sendo o objectivo do presente trabalho a realização de um detetor de sinal digital a partir de uma frequência intermédia é do maior interesse saber as características e condições em que chega esse mesmo sinal. A figura 2.1 apresenta o diagrama de blocos geral do sistema. Antes de ser processado digitalmente o sinal de satélite recebido pela antena passa por um bloco analógico (desenvolvido fora do âmbito desta dissertação) cujo objectivo é condicionar o sinal de modo a fornecer uma réplica deste, em amplitude e diferença de fase, a uma frequência mais reduzida, passível de ser digitalizada e processada posteriormente. Para além disto é neste bloco que se efetua a separação do sinal recebido em duas polarizações ortogonais obtendo-se os chamados sinais *copolar* e *crosspolar*. Para além da caracterização do sinal a detectar serão apresentados vários métodos de deteção de sinal CW e alguns resultados decorrentes da implementação de tais métodos em *Simulink*.

2.2 Caracterização do sinal a detectar

O satélite Alphasat I-XL terá dois *beacons* (padrões de frequência) coerentes nas bandas Ka e Q, derivados de uma fonte comum de baixo ruído de fase. As características previstas dos *beacons* encontram-se sumariadas na tabela 2.1.

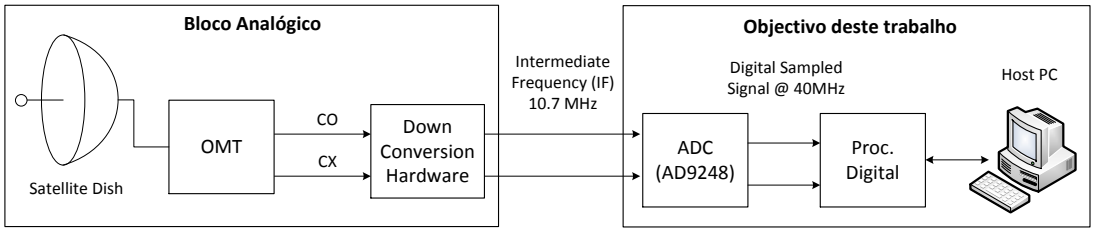


Figura 2.1: Arquitectura geral do detector síncrono baseado em FPGA

2.2.1 EIRP

Define-se por *Equivalent Isotropically Radiated Power (EIRP)* a potência necessária por uma antena isotrópica para produzir a mesma densidade de potência numa dada direção que frequen-

Beacon	**Ka Beacon**	**Q Beacon**
Frequency (GHz)	19.701	39.402
Polarization	Linear Vertical	Linear at 45º
EIRP	21.5	29.3
Frequency stability (3 years) (ppm)	2	2
Phase Noise (dBc) at Amb. Temp		
10 Hz	-48.19	-40.10
100 Hz	-56.00	-51.16
1000 Hz	-68.49	-60.00
10 kHz	-69.95	-62.63

Tabela 2.1: Alphasat I-XL beacon: Características esperadas.

temente é a direção do receptor 2.2. Pode ser calculado através da equação 2.1

$$EIRP = P_T + G_T \quad \text{(dBW)} \tag{2.1}$$

Em que P_T representa a potência transmitida à antena em dBW e G_T o ganho da antena em relação a uma antena isotrópica (dBi). O EIRP toma valores absolutos que podem variar entre 20 e 30 dBW. A ordem de grandeza típica para as variações do EIRP devidos aos ciclos de aquecimento/arrefecimento de periodicidade diária é de algumas décimas de dB.

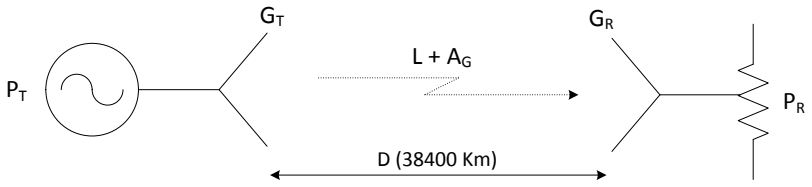


Figura 2.2: Power Budget: conjunto emissor-receptor.

2.2.2 Bloco Analógico

Não é objectivo desta dissertação aprofundar o funcionamento e as características deste bloco pelo que se irá apenas fazer uma breve descrição dos aspectos essenciais.

A arquitetura do bloco é ilustrada na figura 2.3. A antena é do tipo *cassegrain* com um prato de 1 m^2, a separação das polarizações é feita com recurso a um *OrthoMode Transducer (OMT)* com um isolamento de 30 dB entre os canais. Posteriormente é feita uma conversão em frequência seguindo um esquema de desmodulação super-heterodino com vários estágios onde consecutivamente se vai diminuindo a frequência intermédia e a largura dos filtros passa banda.

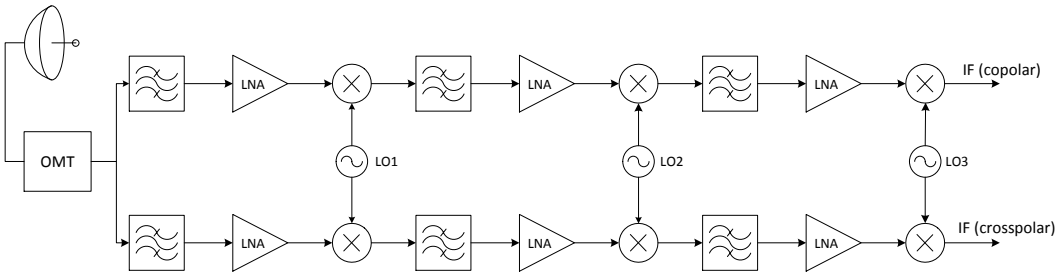


Figura 2.3: Conversão do sinal recebido para a frequência IF.

2.2.2.1 CNR - Link Budget

O *link budget* é calculado relativamente ao *beacon* Ka e para a saída receptor analógico descrito na secção anterior. CNR por definição é o quociente entre potência da portadora e a densidade espectral de ruído que o acompanha, ou a diferença no caso de os respectivos valores estiverem representados em unidades logarítmicas.

$$CNR_0 = P_{recebida} - \eta_{ruído} \quad \text{(dB/Hz)} \tag{2.2}$$

Uma estimativa da potência recebida pode ser calculada através da seguinte equação:

$$P_{in} = EIRP - (L + L_{omt} + A_g) + G_r \quad \text{(dBW)} \tag{2.3}$$

Em que L e L_{omt} representa as perdas em espaço livre devido à distância do satélite (38400 km) e as perdas do OMT, A_g a atenuação devido a partículas atmosféricas e G_r o ganho da antena receptora. Por outro lado a densidade espectral de ruído à saída da unidade analógica $\eta_{ruído}$ é dado por:

$$\eta_{ruído} = k_B\, T_{eq} = k_B\ (T_{sky} + T_{rx}) \quad \text{(dBW/Hz)} \tag{2.4}$$

A temperatura de ruído equivalente do sinal no receptor à frequência intermédia apresenta contribuições da potência de ruído recebida pela antena devido à absorção atmosférica (T_{sky}) e do ruído introduzido pelos atenuadores e amplificadores da cadeia de recepção T_{rx}, k_B representa a constante de *Boltzmann* ($-228.6\ \ (dBW/Hz)\,/K$).

$$T_{rx} = T_0 \left(10^{\frac{L_{omt}}{10}} - 1\right) + \frac{T_{Lna}}{10^{-L_{omt}/10}} \quad \text{(K)} \tag{2.5}$$

$$T_{Lna} = T_0 \left(10^{\frac{NF}{10}} - 1\right) \quad \text{(K)} \tag{2.6}$$

Assume-se que o ganho do primeiro *Low Noise Amplifier (LNA)* é suficientemente grande para se desprezar a contribuição de ruído pelos amplificadores seguintes. NF representa a figura de ruído do primeiro LNA e L_{omt} a figura de ruído do OMT que é idêntico ao seu factor de perdas.

A tabela 2.2 resume os valores envolvidos no cálculo do *link budget* e apresenta o valor esperado para a CNR do sinal em condições normais.

Por último a *Signal to Noise Ratio (SNR)* do sinal obtido pode ser calculada através de:

$$SNR = CNR - 10\log_{10}(B_i) \tag{2.7}$$

onde B_i é a largura de banda do detetor.

Para o valor médio de CNR do sinal (56.4 dB) pode-se verificar que para uma largura do filtro de pré deteção B_i maior que 1 MHz (60 dB) a potência do ruído é superior à do sinal.

Var.	Description	Value (Ka-Band)	Un.
EIRP	Potência isotrópica radiada equivalente	21.5	dBW
G_r	Ganho da antena 1m	44.1	dB
L	Perdas de propagação	210.0	dB
A_g	Perdas Atmosféricas	0.5	dB
L_{omt}	Perdas OMT	0.8	dB
P_{in}	Potência à saída do OMT	-115.7	dBm
T_{sky}	Temperatura de ruído: atmosfera (*Clear Sky*)	32.9	K
NF	LNA Noise Factor	3.0	dB
T_{rx}	Temperatura de ruído: receptor (*Clear Sky*)	438.5	K
CNR	*Carrier to noise ratio* (*Clear Sky*)	56.4	dB/Hz

Tabela 2.2: *Link Budget*: Sistema satélite → recetor terrestre.

2.3 Métodos de deteção

Como foi referido anteriormente, a detecção consiste na estimação da amplitude/potência do sinal de entrada e no caso de dois canais também a fase relativa entre os dois sinais. Nesta secção serão descritas algumas das técnicas mais comuns tendo em consideração que o sinal à entrada do detetor se encontra afetado por ruído *Additive White Gaussian Noise (AWGN)* de banda estreita resultante da filtragem passa banda pelo filtro de pré detecção B_i.

$$s(t) = A_0 \cos\Big(\omega t + \phi_n\big(t\big)\Big) + n_i(t)\cos(\omega t) + n_q(t)\sin(\omega t) \tag{2.8}$$

Onde $n_i(t)$ e $n_q(t)$ são as componentes em fase e quadratura do ruído gaussiano em banda base e $\phi_n\big(t\big)$ o ruído de fase do sinal.

2.3.1 Detecção no domínio do tempo

Provavelmente o método mais simples de implementar (2.4), consiste essencialmente num bloco quadrador seguido de um filtro passa baixo para eliminar a réplica de maior frequência.

A saída do detetor é dada por 2.9.

$$\overline{s(t)^2} = \frac{{A_0}^2}{2} + \eta_0 B_1 \tag{2.9}$$

Como se pode verificar a amplitude detectada inclui um desvio sistemático devido à deteção de ruído não filtrado por B_1. A utilização de um filtro arbitrariamente estreito de modo a tornar o erro desprezável também não é uma solução viável pelos seguintes motivos:

- Em situações reais o sinal a detectar não se encontra, em frequência, contido numa única risca infinitesimal. Tomando o exemplo do *beacon* Ka (2.1) a potência do sinal encontra-se espalhada numa largura de banda de cerca de 50 Hz devido ao ruído de fase dos osciladores do recetor e do sinal.
- A utilização de um filtro largo implica a admissão de ruído adicional no resultado da detecção, por outro lado também não é possível recorrer a um filtro demasiado estreito com risco de se estar a cortar potência do sinal.
- O sinal na prática sofre deslocamentos na frequência central devido ao envelhecimento dos osciladores a cristal tanto na parte de emissão como de recepção.

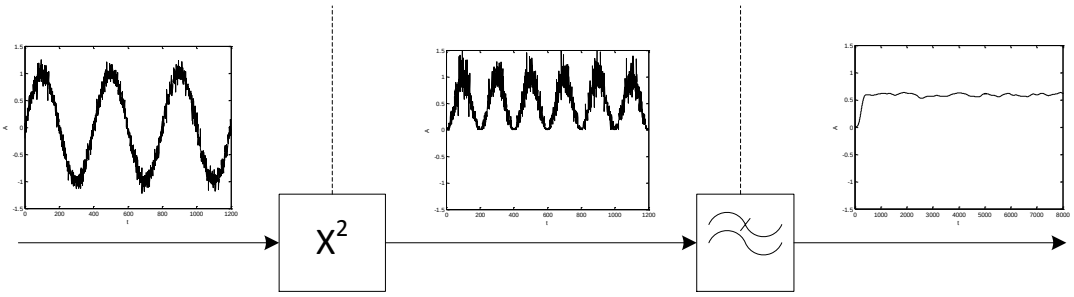


Figura 2.4: Detecção no tempo via quadrador e filtro passa-baixo.

2.3.2 Deteção no domínio da frequência

Um outro método de detecção pode ser implementado utilizando medidas no domínio da frequência. Um dos métodos mais comuns consiste em realizar FFTs em tempo real do sinal, procurar o máximo e somar as riscas adjacentes correspondentes à largura de banda do sinal. Considerando que a potência do sinal se encontra contida entre as riscas r_1 e r_2 (inclusivé) a potência desta banda pode ser calculada recorrendo à fórmula de Parseval.

$$P_b = \sum_{n=r_1}^{r_2} S(n)S(n)^* \tag{2.10}$$

Onde S(n) representa a N-ésima linha espectral da FFT complexa e (*) o operador de conjugado complexo. Naturalmente P_b representa não só a potência do sinal mas também a potência do ruído AWGN. De modo a obter um valor mais preciso pode-se estimar estimar um valor da densidade espectral ruído através das restantes riscas.

$$\eta_0 = \frac{\sum_{n=0}^{r_1} |S(n)|^2 + \sum_{n=r_2}^{N-1} |S(n)|^2 - P_b}{B_n} \tag{2.11}$$

Onde B_n representa a largura de banda do sinal processado pela FFT excluindo as riscas que contêm o sinal. Não é estritamente obrigatório que se utilizem todos os índices da FFT mas uma melhor estimativa resulta se tal for efetuado.

$$B_s = \big(r_2 - r_1 + 1\big)\frac{f_a}{N} \qquad B_n = \Big(N - \big(r_2 - r_1 + 1\big)\Big)\frac{f_a}{N} \tag{2.12}$$

Finalmente um valor mais exato da potência detectada pode ser obtido obtido subtraindo a P_b o valor estimado do ruído para a respectiva banda (B_s) o qual deve ser integrado durante o tempo mais longo possível.

$$P_s = P_b - \eta_0\, B_s \tag{2.13}$$

Naturalmente neste contexto N representa o número de amostras utilizadas para o algoritmo da FFT e f_a a frequência de amostragem do sinal.

2.3.3 Deteção síncrona usando malhas de sincronismo

Um outro método de detecção promissor é o uso de um oscilador local $A_{ref}(t)$ sincronizado com o sinal de entrada (possivelmente com um offset de fase θ_0).

$$A_{ref}(t) = A_{OR}\cos\Big(\omega t + \phi_n\big(t\big) + \theta_0\Big) \tag{2.14}$$

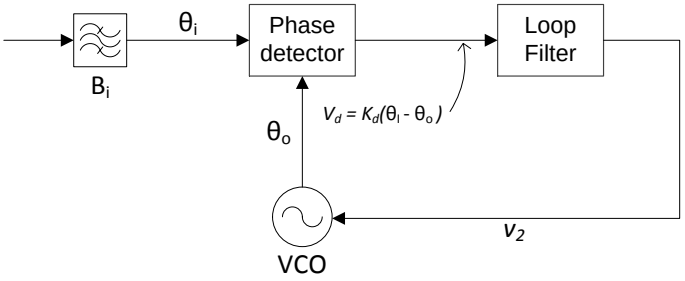


Figura 2.5: Diagrama de blocos genérico de uma PLL.

A obtenção deste sinal síncrono pode ser obtido com malhas de sincronismo como a PLL, *Frequency Locked Loop (FLL)* ou *Costas Loop*, que basicamente utilizam um NCO ou *Voltage Controlled Oscillator (VCO)* inseridos numa malha de realimentação para sintetizar um sinal proporcional em fase ou frequência ao de entrada.

Após a obtenção deste sinal a detecção resume-se basicamente à multiplicação destes dois sinais e à respectiva filtragem passa baixo (ou média/integração). Deve-se ter em atenção que larguras de banda muito reduzidas podem inviabilizar a correta medição de fenómenos de propagação que apresentam frequências mais elevadas.

$$\overline{s(t)_i} = \overline{s(t)\, A_{0r} \cos\Big(\omega t + \phi_n(t) + \theta_0\Big)} = \frac{A_0\, A_{0r}}{2} \cos\,\theta_0 + \frac{A_{0r}\, n_i(t)}{2} \tag{2.15}$$

$$\overline{s(t)_q} = \overline{s(t)\, A_{0r} \cos\left(\omega t + \phi_n(t) + \theta_0 - \frac{\pi}{2}\right)} = \frac{A_0\, A_{0r}}{2} \sin\,\theta_0 + \frac{A_{0r}\, n_q(t)}{2} \tag{2.16}$$

d

Após deteção toda a potência do sinal encontra-se em banda base e a contribuição do ruído de fase para a medida parece ser nula, adicionalmente se: $\theta_0 = k.\pi/2, (k \in \mathbb{Z})$ a componente em quadratura torna-se nula o que permite excluir $n_q(t)$, ou seja, metade do ruído presente na deteção o que contribuí para uma melhoria de 3 dBs da SNR. Na secção seguinte dedicada às PLL será possível verificar que esta condição se verifica sempre que a malha se encontre em sincronismo.

2.4 Malhas de sincronização - PLL

Uma PLL ou malha de seguimento de fase consiste basicamente num sistema de realimentação constituído por três blocos essenciais que se podem observar na figura 2.5: um detetor de fase, um oscilador local controlável e um filtro de malha. O detetor de fase calcula a diferença de fase instantânea entre o sinal de entrada e de saída (VCO) que por sua vez é integrado ao longo do tempo de modo a levar o VCO a sincronizar em frequência com o sinal de entrada. Uma vez em sincronismo qualquer desvio de frequência é compensado com o desenvolvimento de uma tensão de correção no detetor de fase.

Recordando que o sinal é corrompido por ruído aditivo AWGN e ruído de fase é necessário que a PLL esteja convenientemente dimensionada de modo a rejeitar o mais possível o ruído aditivo, sem no entanto, comprometer a qualidade e a gama dinâmica da deteção.

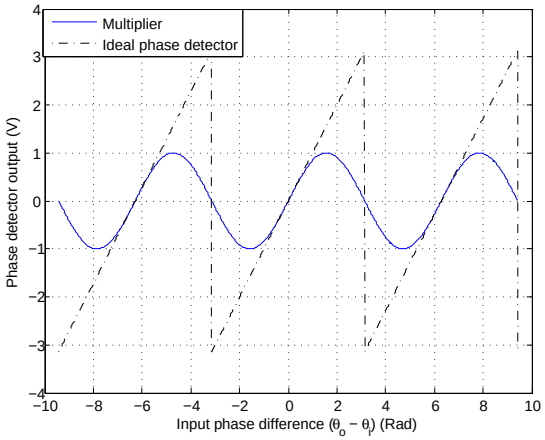


Figura 2.6: Comparação entre a saída do detector de fase ideal e um multiplicador, considerando entradas sinusoidais.

2.4.0.1 Detetor de Fase

Idealmente um detetor de fase tem uma saída proporcional à diferença de fase entre duas entradas:

$$v_d = K_d\,(\theta_i - \theta_o) \tag{2.17}$$

Onde K_d é o ganho do detetor de fase e tem dimensões de V/rad. Muito frequentemente este componente é um simples multiplicador, assumindo que os sinais de entrada se encontram inicialmente dessincronizados:

$$y_i(t) = A\sin\Big(\omega\,t + \phi_i(t)\Big) \tag{2.18}$$

$$y_o(t) = B\cos\Big(\omega\,t + \phi_0(t)\Big) \tag{2.19}$$

A frequência angular dos dois sinais não é necessariamente igual uma vez que a diferença pode ser incluída em $\phi_i(t) - \phi_o(t)$. A saída do multiplicador, desprezando a componente de alta frequência, resulta num sinal do tipo:

$$u_1(t) = \frac{A\,B}{2}\sin\Big(\phi_i(t) - \phi_0(t)\Big) \tag{2.20}$$

Um dos problemas do multiplicador em relação ao detetor ideal é a não linearidade do mesmo (figura 2.6), o que traz consequências no comportamento dinâmico da PLL como se vai ver mais adiante. Contudo para pequenos valores de $\Big(\phi_i(t) - \phi_0(t)\Big)$ é possível fazer a aproximação $\sin(\theta_d) \approx \theta_d$

2.4.0.2 Filtro de Malha

O filtro de malha é bastante influente sobre o comportamento dinâmico da PLL. O filtro é de natureza passa-baixo de modo a eliminar as componentes de alta frequência à saída do detetor, e naturalmente também ruído. Um estudo exaustivo dos vários tipos de filtros de malha é efetuado em [Gar66] e [Bla76], contudo apenas vai ser abordado o filtro de 2ª ordem com integrador perfeito e correção de avanço de fase representado na figura 2.7.

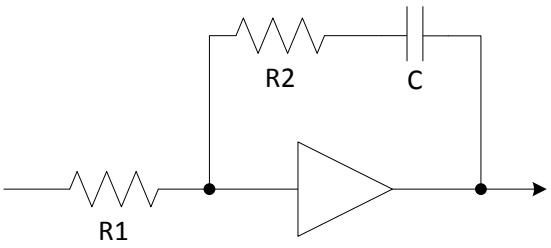


Figura 2.7: Filtro de malha de 2ª ordem com integrador e correcção de avanço de fase.

A saída do filtro em termos de transformada de Laplace vem dada por:

$$F(s) = \frac{s\,CR_2 + 1}{s\,CR_1} = \frac{s\,\tau_2 + 1}{s\,\tau_1} = \frac{\tau_2}{\tau_1} + \frac{1}{s\,\tau_1} \tag{2.21}$$

Onde τ_1 e τ_2 representam as constantes de tempo, podendo-se concluir que a função de transferência apresenta um fator proporcional τ_2/τ_1 e um fator integrador $1/\tau_1$.

Para além da filtragem de ruído o filtro tem como objectivo integrar ou fazer a média da saída do detetor de fase. Isto significa que para o sinal de entrada do VCO estabilizar é necessário que o sinal de entrada do filtro seja em termos médios igual a zero o que significa, por sua vez, que as frequências dos sinais de entrada da PLL são idênticas.

2.4.0.3 VCO

Como o próprio nome indica são osciladores controlados por tensão. A saída é uma sinusóide de amplitude constante cuja frequência $\Delta\,\omega_o$ em torno de uma frequência central é função da tensão de controlo de entrada v_c. O desvio do VCO em relação à frequência central é dada então por:

$$\Delta\,\omega_o = K_o\,v_c \tag{2.22}$$

Uma vez que a frequência é a derivada em ordem ao tempo da fase a equação pode ser reescrita como:

$$\frac{d\theta_o}{dt} = K_o\,v_c \tag{2.23}$$

Ou aplicando a transformada de Laplace:

$$s\,\theta_o(s) = K_o\,V_c(s) \Rightarrow \theta_o(s) = \frac{K_o\,V_c(s)}{s} \tag{2.24}$$

2.4.0.4 Função Transferência

Na figura 2.8 está representado um diagrama da malha da PLL, cujas funções de transferência de cada bloco estão de acordo com as respectivas equações 2.21 e 2.24. Como se pode observar a função de transferência em malha aberta é dada por:

$$G(s) = \frac{K_d\,K_o\,F(s)}{s} \tag{2.25}$$

e logo, a função de transferência em malha fechada é dada por:

$$H(s) = \frac{G(s)}{1 + G(s)} = \frac{K_d\,K_o\,F(s)}{s + K_d\,K_o\,F(s)} \tag{2.26}$$

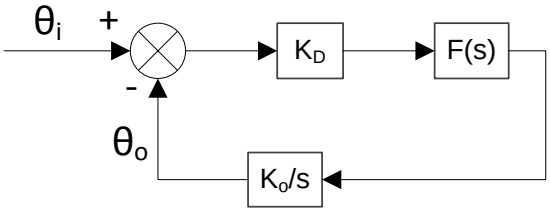


Figura 2.8: Diagrama de blocos da PLL no domínio de Laplace.

Outro parâmetro útil para a análise do sistema é o ganho em Direct Current (DC) em malha aberta dado por:

$$K_v = \lim_{s \to 0} s\, G(s) = K_d\, K_o\, F(0) \tag{2.27}$$

Substituindo $F(s)$ pelo filtro apresentado na seção 2.4.0.2, a função transferência fica finalmente:

$$H(s) = \frac{K_d\, K_o\, (s\, \tau_2 + 1)\, /\tau_1}{s^2 + s\, (K_d\, K_o\, \tau_2/\tau_2) + K_d\, K_o/\tau_1} \tag{2.28}$$

podendo ser reescrita usando a notação comum para sistemas lineares de 2ª ordem,

$$H(s) = \frac{2\zeta\, \omega_n\, s + {\omega_n}^2}{s^2 + 2\, \zeta\, \omega_n\, s + {\omega_n}^2} \tag{2.29}$$

onde ω_n é a frequência natural e ζ é o coeficiente de amortecimento, dados por:

$$\omega_n = \sqrt{\frac{K_d\, K_o}{\tau_1}} \qquad\qquad \zeta = \frac{\tau_2}{2} \sqrt{\frac{K_d\, K_o}{\tau_1}} \tag{2.30}$$

2.4.0.5 Largura de Banda de ruído

Considerando que o sinal à entrada da PLL é caracterizado por uma densidade espectral de ruído de fase $S_{ni}(f)$ este vai ser alterado como resultado da actuação da PLL, recorrendo à teoria dos sistemas lineares invariantes no tempo e considerando o ruído em causa um processo estacionário é possível calcular o ruído à saída do sistema.

$$S_{no}(f) = |H(f)|^2\, S_{ni}(f) \tag{2.31}$$

A potência de ruído $\overline{{\theta_{no}}^2}$ ou jitter de fase à saída resulta da integração matemática de S_{no} sobre a banda do filtro de pré-deteção B_i.

$$\overline{{\theta_{no}}^2} = \int_0^{B_i/2} |H(f)|^2\, S_{ni}(f)\, \mathrm{d}f \approx S_{ni}(f) \int_0^{\infty} |H(f)|^2\, \mathrm{d}f \tag{2.32}$$

A aproximação efectuada considera que o filtro de pre-deteção é muito mais largo que a largura de banda de $H(f)$ o que em geral se verifica sempre devido aos filtros de pre-deteção terem de acomodar as variações na Intermediate Frequency (IF).

Deste modo, e observando a equação 2.32, é possível definir B_L como a largura de banda equivalente que um filtro passa baixo perfeito precisaria de ter para se obter a mesma potência de ruído à saída da PLL.

$$B_L = \int_0^{\infty} |H(f)|^2\, \mathrm{d}f \tag{2.33}$$

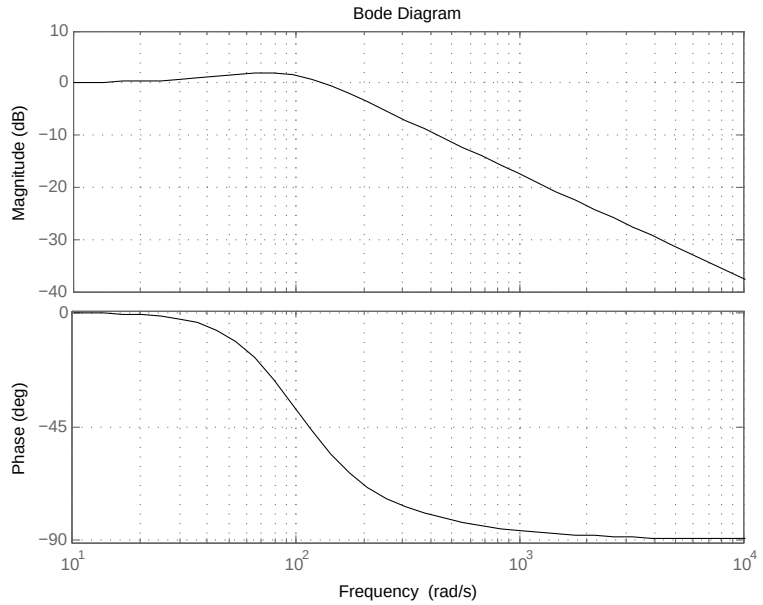


Figura 2.9: Diagrama de Bode da PLL com filtro de malha ativo de 2ª ordem.

Para o caso particular que temos vindo a considerar de um filtro de segunda ordem, o valor 2.33 encontra-se tabelado em [Gar66] e é dado por:

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\,\zeta}\right) \quad \text{(Hz)} \tag{2.34}$$

A curva é apresentada na figura 2.10 e é possível verificar que a largura de banda de ruído apresenta um mínimo para $\zeta = 0.5$.

2.4.0.6 Potência de ruído

De acordo com [Gar66] é possível calcular a potência do ruído de fase de um sinal afectado por AWGN. De facto a adição de ruído na amplitude causa incerteza nas passagens por zero. A relação é estabelecida por:

$$\overline{\theta_{ni}{}^2} = \frac{1}{2SNR_i} \tag{2.35}$$

e consequentemente a potência do ruído de fase à saída da PLL é dada por:

$$\overline{\theta_{no}{}^2} = \frac{B_L}{2CNR_i} \tag{2.36}$$

que é um resultado intuitivo. A potência de ruído à saída da PLL é tanto menor quanto maior for a CNR do sinal de entrada e é proporcional à largura de banda de ruído (B_L), contudo existem também problemas decorrentes de diminuir demasiado este parâmetro:

- A diminuição de B_L implica diminuir a largura de banda do filtro de malha (passa-baixo). Se a diferença de frequência do sinal de entrada em relação ao VCO for demasiado elevada o sinal à saída do detetor de fase é fortemente atenuado o que por sua vez implica que a tensão de correção do VCO pode não ser suficiente para corrigir a sua frequência;
- Da mesma forma, se o sistema se encontrar inicialmente em sincronismo, pequenos desvios na frequência que ocorrem em situações reais serão dificilmente corrigidos pela PLL. Uma

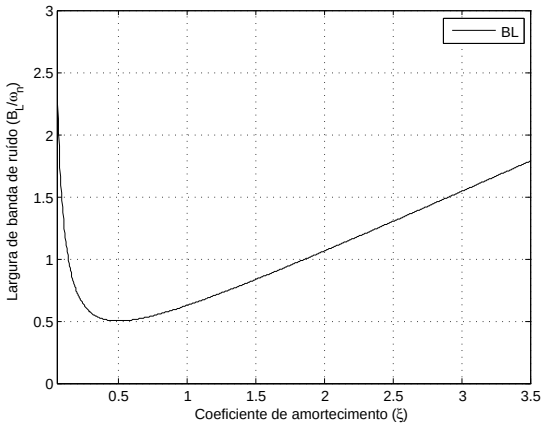


Figura 2.10: Largura de banda de ruído da PLL em função do coeficiente de amortecimento, considerando filtro de malha ativo de 2ª ordem.

outra forma de ver as coisas é constatar que diminuir B_L em geral é conseguido através da diminuição da frequência natural ω_n, o que implica situar os pólos do sistema mais próximo da origem o que por sua vez se traduz em tempos de resposta mais longos relativamente a alterações do sinal de entrada.

2.4.0.7 Manutenção e aquisição de sincronismo

Em termos de sistema, inicialmente, é de esperar que as frequências do VCO e do sinal de entrada sejam diferentes. Consoante a dimensão deste desvio e dos parâmetros da PLL. Se o detetor fosse ideal, independentemente das condições iniciais dos dois sinais, a PLL chegaria sempre ao sincronismo uma vez que desenvolveria sempre uma tensão proporcional à diferença de fase entre os dois sinais porém, como foi visto na secção 2.4.0.1, a utilização de multiplicadores ou outros circuitos de deteção de fase possuem curvas de saída diferentes das ideais. Este facto provoca que em alguns casos a PLL atinja o sincronismo mais lentamente se a diferença de frequências dos sinais de entrada for elevada ou mesmo tornar impossível a aquisição de sincronismo. De acordo com [Gar66] é possível distinguir três gamas de frequências:

- Gama de *Lock-In*: $|\Delta\omega| < K_v$
  O sincronismo é quase imediato, uma vez que a tensão desenvolvida à entrada do VCO nos primeiros ciclos é suficiente para corrigir a frequência do mesmo;
- Gama de *Hold-In*: $K_v < |\Delta\omega| < 2\sqrt{\zeta\omega_n K_v}$
  O sincronismo não é imediatamente atingido, mas é desenvolvida uma tensão de média não nula à saída do detetor de fase (*beatnote*) que ao longo do tempo vai corrigindo a frequência do VCO;
- Gama em que não é atingido sincronismo: $2\sqrt{\zeta\omega_n K_v} < |\Delta\omega|$

2.4.1 PLL: Simulação em Simulink

Com o objectivo de aprofundar o conhecimento na área das PLLs e para efeitos de estudo do sistema a implementar foi realizado um modelo parametrizável em *Simulink*. O diagrama de blocos encontra-se na figura 2.11, essencialmente trata-se de uma PLL simples em que a saída do VCO é utilizada para obter as componentes em fase (I) e quadratura (Q) de um sinal sinusoidal de entrada. Para além disto foram introduzidas várias não idealidades como a adição de ruído de fase à entrada do sistema e offset de tensão à saída dos multiplicadores.

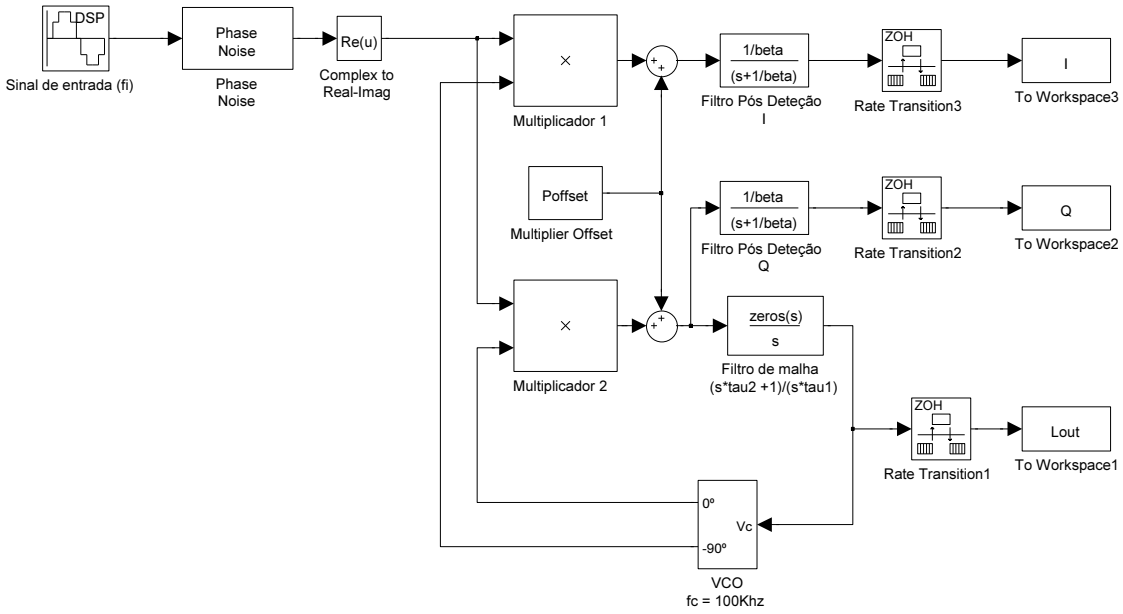


Figura 2.11: Circuito de simulação da PLL realizado em Simulink.

O ruído de fase sintetizado apresenta um decaimento na forma de $1/f$ em torno da frequência central da portadora, como se pode observar na figura 2.15.

De modo a avaliar o impacto de certos parâmetros do loop, foram efetuados alguns testes com este modelo, nomeadamente:

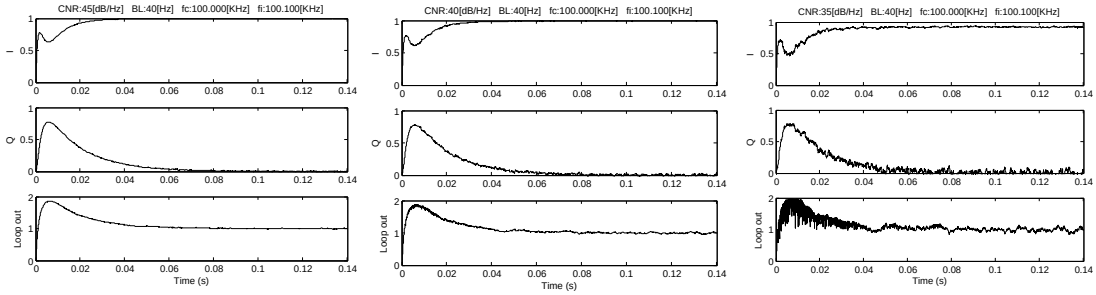
1. Variação da CNR do sinal de entrada, para um desvio inicial de frequência de 100 Hz em relação à frequência inicial do VCO *(lock-in)*;
2. Variação da frequência de entrada para um valor de CNR médio (40 dB/Hz);
3. Variação da largura de banda de ruído.

Os resultados destes testes, nomeadamente as componentes I/Q detetadas e a tensão de entrada do VCO *(L_{out})*, estão representadas nas figuras seguintes. O propósito principal destes testes foi avaliar qualitativamente o impacto dos parâmetros do loop pelo que a análise será da mesma natureza:

1. figura 2.12 - Verifica-se que um aumento do ruído leva a uma maior variância nas componentes I/Q como seria de esperar;
2. figura 2.13 - **(a)** o desvio de frequência do sinal de entrada para a frequência central do VCO encontra-se dentro da gama de *lock-in* pelo que a aquisição de sincronismo é quase imediata, fora desta gama **(a e b)** o sincronismo é muito mais demorado, havendo deslizamento das duas frequências o que leva ao aparecimento das oscilações observadas nas componentes I/Q;
3. figura 2.14 **(a)** a largura de banda do *loop* é muito reduzida (10 Hz) pelo que o sinal de entrada é fortemente atenuado o que impede o VCO a adquirir o sincronismo rapidamente, **(c)** largura de banda excessiva, o *loop* adquire rapidamente o sincronismo mas as componentes I/Q são bastante afetados pelo ruído fora da banda do sinal, **(b)** situação intermédia em que existe o compromisso entre rapidez da resposta dinâmica e variância de ruído à saída.

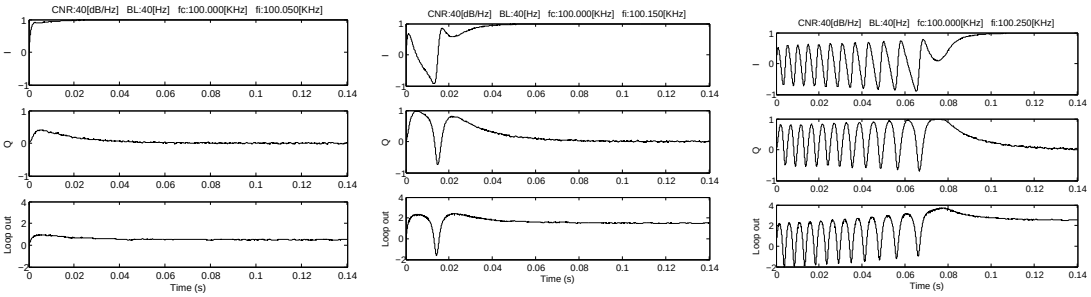
2.5 Sumário

Neste capítulo foi descrito o bloco analógico prévio ao detetor digital, o link budget foi calculado à saída do mesmo para valores típicos de ruído atmosférico obtendo-se uma valor de cerca de 56 dB para a CNR esperada do sinal. O espalhamento espectral da potência do sinal a detectar e o desvio em frequência ao longo do tempo do mesmo torna pouco viáveis os métodos de deteção com base em quadrador e filtragem. Os métodos tipo PLL que utilizam uma portadora sincronizada com o sinal de entrada resolvem este problema pelo que será o método a implementar no detetor digital, contudo, é necessário dimensionar corretamente os parâmetros da malha de modo a haver um bom compromisso entre a variância de ruído à saída e rapidez da resposta dinâmica.



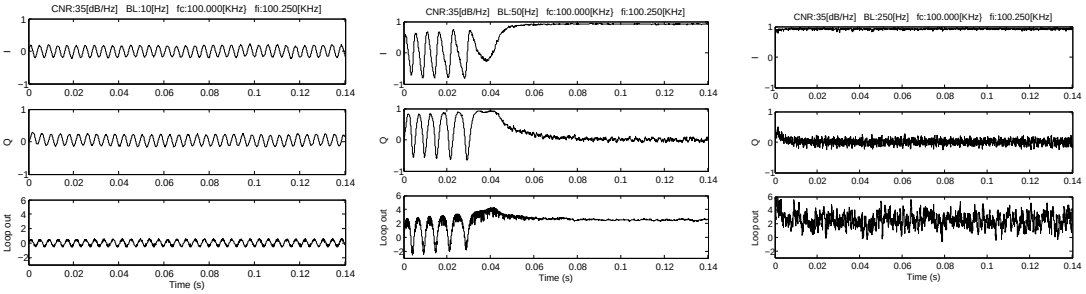
(a) CNR - 45 dB/Hz (b) CNR - 40 dB/Hz (c) CNR - 35 dB/Hz

Figura 2.12: Simulações com variação de CNR. Δf =100 Hz, B_L =40Hz, $\zeta = 0.707$.



(a) Δf - 50 Hz (b) Δf - 150 Hz (c) Δf - 250 Hz

Figura 2.13: Simulações com variação do desvio de frequência. CNR=40 dB/Hz, B_L =40Hz, $\zeta = 0.707$.



(a) B_L - 10 Hz (b) B_L - 50 Hz (c) B_L - 250 Hz

Figura 2.14: Simulações com variação da largura de banda de ruído. CNR=35 dB/Hz, Δf =250 Hz, $\zeta = 0.707$.

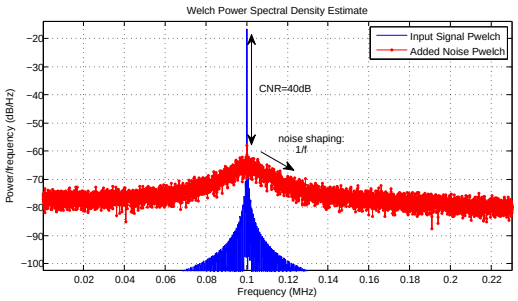


Figura 2.15: Espectro do sinal de entrada simulado, com adição de ruído.

Capítulo 3

Plataforma de Hardware do Detetor

3.1 Introdução

Este capítulo tem como objectivo descrever e justificar as escolhas efetuadas a nível de hardware para a implementação do detetor de sinal a partir da sua frequência intermédia. Como foi referido no capítulo anterior a frequência do sinal encontra-se situada nos 10.7 MHz e é necessário efetuar a sua conversão para o domínio digital de modo a poder ser processada pelo detetor. Quanto ao processamento digital são descritas brevemente as duas opções mais comuns, nomeadamente os *Digital Signal Processors* (DSPs) e as FPGAs.

3.2 Field Programmable Gate Arrays (FPGA's)

Uma FPGA é, na sua essência, um circuito integrado que contêm uma matriz de elementos programáveis que podem ser interconectados de forma flexível para implementar um circuito digital específico. Os elementos lógicos podem-se dividir basicamente em três categorias:

- Blocos Lógicos - tipicamente *Configurable Logic Blocks (CLBs)*, os blocos básicos de construção do sistema digital compostos geralmente por *Look up Tables* (LUT), registos (Flip Flops) e *multiplexers*.
- Blocos I/O - Permitem a conexão de sinais internos da FPGA aos pinos de I/O e vice-versa, condicionamento de sinal (níveis de tensão) e configurar o drive de corrente.
- *Switch Matrixes* - Responsáveis pela configuração das ligações entre elementos lógicos.

A configuração dos elementos lógicos é geralmente efetuada a partir de linguagens de descrição de *hardware* que descrevem a nível comportamental ou a nível das próprias ligações o circuito desejado.
Uma das vantagens de efetuar processamento de sinal em FPGA decorre do facto de o processamento ser inerentemente paralelo o que permite a realização de vários etapas de tratamento de sinal simultaneamente sem que para isso sejam necessárias frequências de relógio muito elevadas (quando comparadas com uma implementação em DSP).

Deste modo quase se pode dizer que a FPGA é um meio termo entre um Application-specific integrated circuit (ASIC) e uma DSP, permitindo a realização de sistemas complexos e específicos para a aplicação em causa e ao mesmo tempo permitindo a reconfigurabilidade do mesmo.

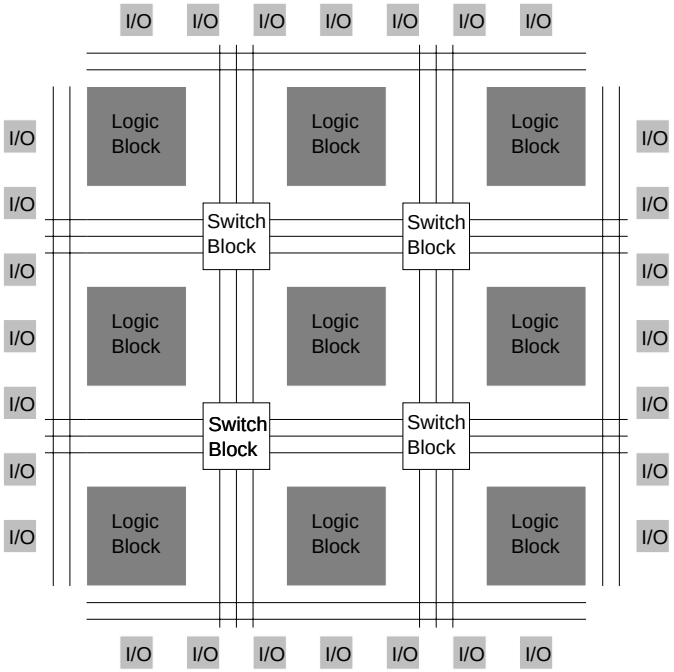


Figura 3.1: FPGA: Arquitectura em forma de matriz.

3.2.1 Comparação entre FPGAs e DSPs

Os algoritmos de processamento digital podem ser implementados também através de DSPs, sendo os mesmos, microprocessadores otimizados para o efeito. A otimização consiste essencialmente numa arquitetura que facilita a realização de produtos internos e operações aritméticas que permitem maior eficácia em operações comuns de processamento de sinal, nomeadamente: convolução, correlação, FFT, cálculos matriciais, etc.

As DSPs são também bastante flexíveis e podem ser programados de forma fácil e rápida com recurso a linguagens de alto-nível, como por exemplo C. Contudo em comparação com as FPGAs a flexibilidade é menor decorrente da arquitetura fixa das DSPs, em contrapartida os tempos de desenvolvimento são muito menores. A tabela 3.1 resume as vantagens e desvantagens das duas tecnologias.

Tendo em conta a referida tabela e o esquema de um detetor com recurso a malha de sincronização as vantagens são evidentes. Dificilmente uma DSP de baixa-média gama teria capacidade para processar continuamente e em tempo-real um sinal amostrado a uma frequência da ordem dos MHz. Para além disso, quer as necessidades de I/O do sistema (2 canais de entrada, e envio em tempo real da detecção para um computador) quer a natureza das malhas de sincronismo, fazem valer a paralelização de operações inerente à implementação em FPGA. Esta escolha permite também implementar simultaneamente os vários tipos de detecção apresentados no capítulo 2.

3.3 Módulo FPGA

Na escolha da FPGA optou-se por adquirir um módulo de modo a poupar esforços de desenvolvimento do hardware de suporte ao *chip* da FPGA. Os módulos industriais da *Trenz Electronic* com FPGAs de baixa-gama (*Spartan 3* e *Spartan 6*) [Ele13c] [Ele13d] revelaram-se bastante interessantes quer pelo tamanho compacto da placa, preço reduzido e quantidade de funcionalida-

	FPGA	**DSP**
Aplicação em sistemas de SDR	✓ IP cores facilitam a implementação de multiplicadores, divisores, FFT e operações trigonométricas (CORDIC). ✓ NCO de fácil síntese através de blocos de propriedade intelectual parametrizáveis. Implementação eficaz de filtros CIC tirando partido da paralelização de operações. ✗ Maior esforço de desenvolvimento na implementação de operações matemáticas, cálculos em vírgula flutuante dispendiosos em termos de recursos.	✗ Implementação de NCO e filtros CIC pouco eficientes a frequências relativamente elevadas, reduzindo significativamente o tempo disponível para o processamento a jusante. ✓ Maior facilidade na implementação de operações matemáticas, especialmente em vírgula flutuante. ✓ Implementação dos algoritmos em linguagens de alto nível (C) facilitam e permitem reduzir o tempo de desenvolvimento.
Flexibilidade	✓ Reconfiguração da FPGA permite a readaptação dos recursos às necessidades do sistema. ✗ Sistemas com um número elevado de operações condicionais são mais díficeis de implementar. Alocação de recursos para cada configuração.	✗ Arquitetura fixa das DSPs torna mais criteriosa a escolha da mesma para a aplicação em causa. ✓ Mudança de contexto facilitada pelo salto do processador para diferentes zonas do programa, re-utilização em run-time dos recursos de processamento.
Desempenho	• Melhor desempenho para sistemas que podem tirar partido da paralelização de operações. • Frequências de relógio menores devido aos elementos de switching da FPGA que aumentam os tempos de propagação das linhas, compensado pela paralelização de operações.	• A arquitectura centralizada torna-a mais eficiente em aplicações que não tirem partido de paralelização. • Frequências de relógio mais elevadas mas o tempo do processador é dividido em execução de instruções, acesso à memória e execução de rotinas de interrupção I/O.

Tabela 3.1: Vantagens e desvantagens entre FPGA e DSP.

Módulo	FPGA	Família	Eq.Cells	BRAM	DSP	#IOs	Preço
TE0320-EV02	XC3SD1800A	Spartan-3A DSP	37 K	1.5 Kb	84	109	177€
TE0320-EV02B	XC3SD3400A	Spartan-3A DSP	53 K	2.2 Kb	109	469	201€
TE0630-00	XC6SLX45	Spartan-6 LX	43 K	2 Kb	58	110	190€
TE0630-00IBF	XC6SLX75	Spartan-6 LX	74 K	3 Kb	132	110	272€
TE0630-00IV	XC6SLX150	Spartan-6 LX	147 K	4.8 Kb	180	110	332€

Tabela 3.2: Comparativo dos módulos FPGA da Trenz Electronic.

des/periféricos incluídos. A tabela 3.2 reflete as opções mais apropriadas para o projeto em causa. Todos estes módulos apresentam um controlador USB e respectivo conector em forma "mini" facilitando as comunicações com o computador hospedeiro com taxas de transferência elevadas.

Dos módulos apresentados na tabela todos apresentam um número suficiente de pinos I/O de uso geral de modo a realizar a interface com a ADC. O critério de escolha seguinte recaíu nos recursos da FPGA e no preço dos módulos. Os primeiros três módulos na tabela 3.2 apresentam um número equivalente de células lógicas e à partida consideradas suficientes para a implementação do sistema. No final optou-se pelo modelo `TE0320-EV02B` pela abundância de blocos DSP e BRAMs que se revelaram bastantes úteis. A diferença de preços entre *Spartan 3* e *Spartan 6* para um número semelhante de células equivalentes não justificou a aquisição de um modelo *Spartan 6*, a principal vantagem da utilização de *Spartan 6* adviria da utilização de LUTs de 6 entradas que possibilitam a diminuição do número de níveis lógicos e consequentemente aumentar a frequência de funcionamento do sistema no entanto, considerando as frequências baixas envolvidas (ordem dos 40 MHz) optou-se por *Spartan 3A*.

3.3.1 Trenz Electronic TE0320-EVO2B

Como foi referido na secção anterior a escolha do módulo recaiu sobre este modelo. Para além do *chip Spartan 3A-DSP* o módulo apresenta diversos componentes que auxiliam a configuração da FPGA e extendem as funcionalidades da mesma, a destacar:

- Controlador USB 2.0 *device* e respectivo conector para interface com o exterior;
- 2x módulos de memória *DDR SDRAM* de 512 *Mbit* para armazenamento de dados da FPGA;
- Conector JTAG para fins de depuração (através da aplicação *Chipscope*) e envio do *bitstream* de configuração para a FPGA;
- Memória Serial Peripheral Interface Bus (SPI) para armazenamento do *bitstream* da FPGA;
- EEPROM (armazenamento para o *firmware* do controlador USB FX2);
- Osciladores de onda quadrada CMOS/TTL: 100 MHz (FPGA) e 24 MHz (Controlador USB e FPGA);
- 2x conectores B2B disponibilizando vários pinos de I/O e configuração para o exterior;
- LEDs e *push buttons* de uso geral;
- Reguladores de corrente/tensão para todos os componentes listados e circuitos de filtragem de ruído.

A placa e o respectivo diagrama de blocos encontram-se nas figuras 3.2 e 3.3. O módulo é alimentado através dos conectores B2B para uma tensão de entrada de 5V (gama entre 4V e 7V),

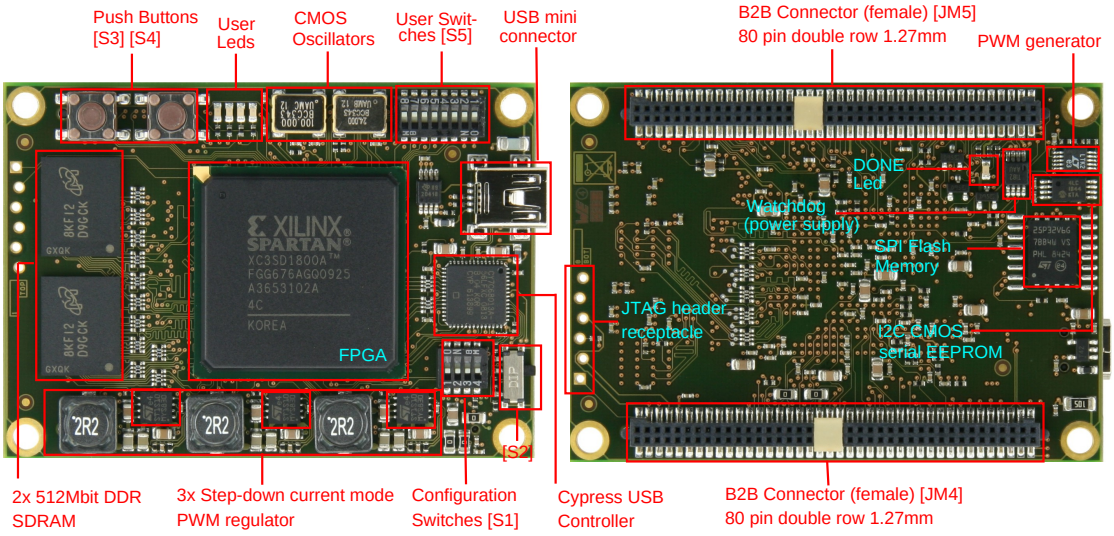


Figura 3.2: Módulo TE0320-EVO2 (Trenz Electronic) e identificação de elementos. [Ele13a]

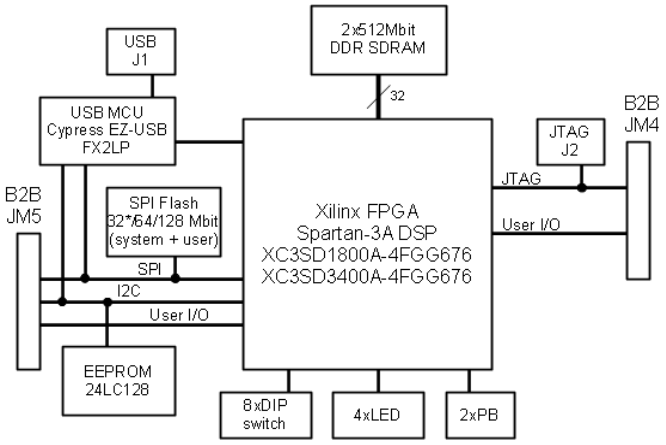


Figura 3.3: Diagrama de Blocos do módulo TE0320 da Trenz Electronic. [Ele13b]

o *bitstream* da FPGA é armazenado na memória SPI sendo a FPGA configurada após *power-on* ou *reset*. A configuração é assinalada pelo sinal `DONE` que se encontra ligada ao *LED* da *bottom layer* do módulo. Informação relativa ao processo de escrita do *bitstream* na memória SPI do módulo pode ser encontrada no Anexo C.

Os conectores Board to Board (B2B) disponibilizam para o exterior grande parte (109) dos pinos I/O de uso geral da FPGA. A ideia seguinte consistiu em realizar uma Printed Circuit Board (PCB) que permitisse realizar a conversão analógico-digital do sinal IF e fornecer as alimentações para o módulo da FPGA.

3.4 ADC - AD9248

Para a escolha da ADC o critério mais importantes é a frequência de amostragem que deverá permitir uma amostragem do sinal à frequência IF (neste caso 10.7 MHz) sem *aliasing* e com

número de bits suficientes para não reduzir significativamente a CNR do sinal. Pela facilidade de obtenção e pelos bons resultados obtidos em [Sou07] e [Pir07] foi utilizado o modelo AD9248 da *Analog Devices*. Trata-se de uma ADC de 2 canais, tipo *pipelined*, de 14 *bits* e com frequência de *sampling* de 40 MHz. O princípio de funcionamento deste tipo de conversores digitais é apresentado na figura 3.4.

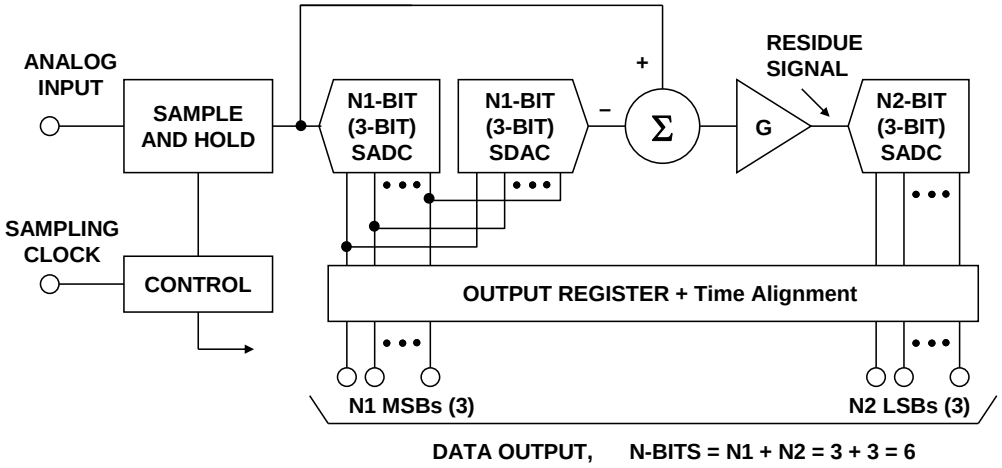


Figura 3.4: Arquitectura geral de uma ADC pipelined. Exemplo para 6 bits. [Kes13]

Como se pode observar a arquitetura é uma sequência de blocos constituídos por uma ADC de baixa precisão seguida de uma DAC que converte a saída digital da primeira, este valor depois é subtraído à tensão de entrada de modo a calcular o erro remanescente que será amplificado e processado pelo bloco ADC/DAC seguinte, obtendo-se assim, os *bits* menos significativos. Este tipo de arquitetura permite frequências de amostragem elevadas (< 200 MS/s) que as tornam bastante apropriadas para *sampling* de IF. Geralmente o maior inconveniente associado a estas ADCs é a latência elevada de conversão, que para o caso em questão de um detetor de sinal se torna um parâmetro com muito pouca relevância.

O AD9248 é constituído por duas ADCs *pipelined* (dois canais) idênticas que operam a uma frequência máxima de 40 MS/s definida pelos respectivos *clocks* entrada (exteriores ao *chip*) e possuem uma resolução de 14 *bits* disponibilizados sob a forma de barramento paralelo para o exterior.

A entrada do AD9248 é diferencial, apresentando duas entradas (V+ e V-) para cada canal, permitindo a conversão de valores negativos. A saída digital a '0' corresponde a uma situação em que ambas as entradas apresentam um valor idêntico de tensão. Ambas as entradas V+ e V- deverão estar contidas entre os valores de alimentação interna da ADC (REFT e REFB) que determinam a amplitude máxima de conversão.

De modo a adaptar o sinal IF à entrada da ADC foi necessário incluir transformadores em cada canal de modo a transformar o sinal *single ended* num sinal balanceado em torno de um valor de tensão médio (neste caso metade da tensão de alimentação da ADC). Por fim, e de modo a completar a interface com a FPGA, foi construída uma placa de circuito impresso com a ADC e conetores B2B de modo a permitir o acoplamento com o módulo TE0320. A placa realizada apresenta-se na figura 3.6, no Anexo A o respectivo esquemático e as opções de configuração da ADC.

3.5 Arquitectura do Detetor e Aspecto Físico

O diagrama físico do sistema é apresentado na figura 3.5. Como foi referido anteriormente os transformadores (*baluns*) efetuam a conversão de *singled ended* para diferencial. O sinal de relógio da ADC é fornecido pela FPGA (40 MHz), sendo este obtido através da divisão na frequência do sinal de 100 MHz fornecido pelo oscilador da placa TE0320. Uma vez que a ADC não apresenta nenhum pino de saída que assinale a presença de uma nova amostra nos barramentos, e uma vez que a conversão de dados ocorre ao mesmo ritmo (40 MS/s) que o relógio fornecido, este sinal de 40 MHz também é utilizado internamente na FPGA para capturar as amostras de entrada.

A interface com o computador hospedeiro é feita através do controlador USB, a operar a uma frequência de 48 MHz, permitindo uma velocidade máxima de transmissão de dados de 48 MB/s desde que o computador hospedeiro tenha capacidade para operar em *High Speed Mode* (a partir de USB 2.0). A arquitetura do detetor implementado em FPGA vai ser apresentada nos próximos capítulos, o capítulo 4 é dedicado ao processamento de sinal propriamente dito e o capítulo 5 à comunicação via USB.

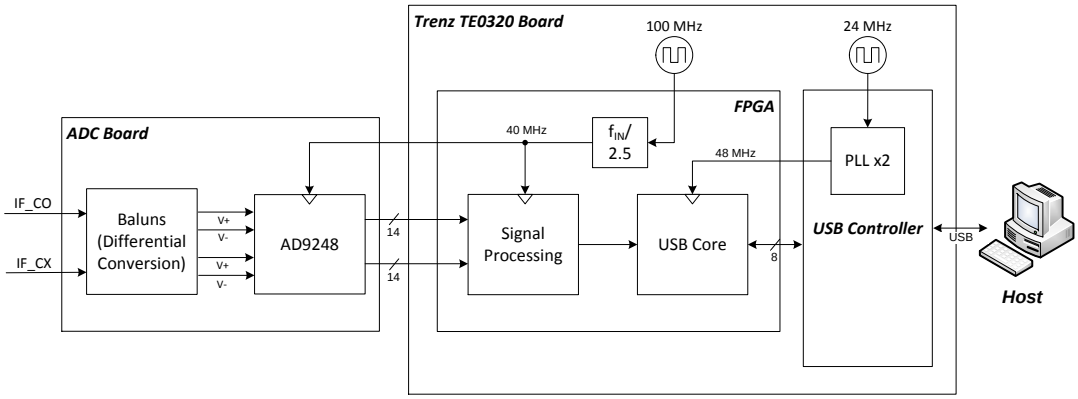
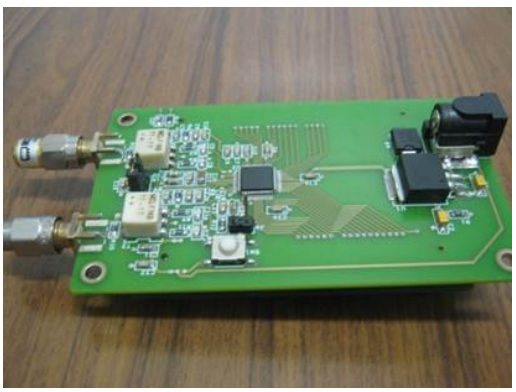
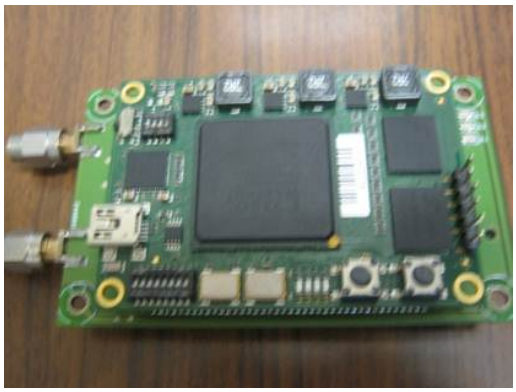


Figura 3.5: Diagrama de blocos do hardware utilizado para a detecção do sinal IF.



(a) Face superior.



(b) Face inferior com o módulo TE0320.

Figura 3.6: Placa de conversão analógico-digital.

Capítulo 4

Arquitectura do detetor síncrono

4.1 Introdução

O diagrama de blocos do detetor síncrono para um canal encontra-se na figura 4.1, dois métodos de detecção foram implementados: detecção síncrona baseada em PLL e detecção em frequência com recurso a um bloco de FFT.

Como foi referido no capítulo 3 cada canal é composto por 14 *bits* que dão entrada nos Input/Output Blocks (IOB) da FPGA. O sinal digital proveniente da ADC passa primeiro por um bloco de condicionamento dentro da FPGA responsável pela amostragem dos *bits* e pela correção de um possível *offset* de amplitude. Posteriormente o sinal é multiplicado por ambas as saídas do NCO e passa pelos blocos de filtragem e decimação (CIC/FIR) que reduzem grandemente a largura de banda do sinal (40 MHz para 4.9 kHz) de modo a reduzir a potência de ruído. Finalmente num último estágio o sinal a baixa frequência passa pelo filtro de malha da PLL e a sua saída é aplicada ao NCO para sincronização (em condições normais) da frequência do mesmo com a do sinal de entrada.

As componentes resultantes da detecção (fase e quadratura) são novamente filtradas por filtros Finite Impulse Response (FIR) para uma frequência de 25 S/s e armazenadas em FIFO. Conjuntamente é também armazenado o resultado da FFT efectuada sobre as mesmas componentes. Posteriormente o conteúdo dos FIFOs é encaminhado para o controlador USB e para o computador hospedeiro.

4.2 Domínios de clock

O sistema implementado apresenta três domínios de relógio o que significa que todos os elementos sequenciais dentro de cada uma das regiões representadas na figura 4.1 operam a essa frequência.

- *40 MHz* - frequência de operação da ADC e do ritmo de entrada dos *bits* de informação na FPGA. Tendo em conta que as entradas do multiplicador têm de ser síncronas com o próprio de sinal relógio deste, tanto o NCO como o multiplicador acabam por operar a esta frequência.
- *100 MHz* - Devido a uma questão de poupança de recursos da FPGA decidiu-se utilizar uma frequência mais elevada para o processamento de sinal, permitindo reduzir significativamente a área utilizada por alguns *IP cores* (essencialmente os filtros *Cascaded Integrator Comb (CIC)* e o bloco de FFT).

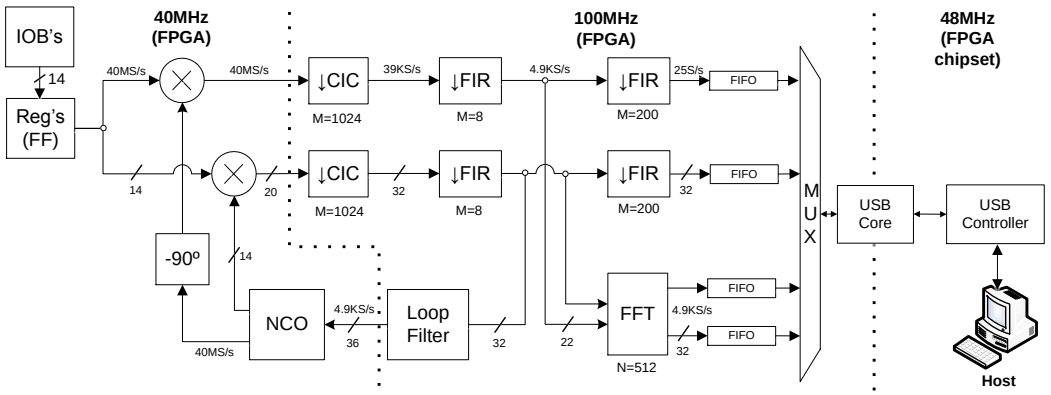


Figura 4.1: Diagrama de blocos do detector síncrono para um canal.

- *48 MHz* - É a frequência de operação do controlador USB, todos os sinais de controlo assim como os dados transferidos são amostrados síncronamente pelo mesmo sinal de relógio fornecido pelo controlador Fx2.

A síntese do sinal de 40 MHz é efetuada através de um *Digital Clock Manager (DCM)*, estes blocos são primitivas da FPGA que permitem a manipulação de sinais relógio, entre outras funções suportam:

- multiplicar ou dividir a frequência de um sinal de relógio de entrada;
- compensar tempos de propagação dentro da *clock network* da FPGA;
- assegurar duty-cycles de 50% (importante para sistemas *double data rate*).

4.2.1 Síntese do clock para ADC

Para além de alimentar internamente os registos e *cores* da FPGA, o relógio de 40 MHz é também fornecido à ADC. Para este fim foi utilizada o *IP core* `clock forwarding` configurado a partir do `Xilinx Clock Wizard`. Esta unidade é composta essencialmente de um DCM seguido de um *Double Data Register (DDR)* como se pode observar na figura 4.2. O sinal de relógio externo (100 MHz) é dividido por um factor de 2.5 e a saída é utilizada como sinal de *toggle* para o DDR. A saída do mesmo é posteriormente encaminhada para os IOB da FPGA e para a ADC.

A inclusão do DDR por parte do *wizard* prende-se apenas por razões de precisão de *duty cycle* uma vez que o sinal de saída é gerado também com a saída de 180° do DCM garantido uma maior precisão do flanco descendente. Este aspecto é essencial para a interface de sistemas *double data rate* ou *Low Voltage Differential Signaling (LVDS)* a frequências elevadas (>150 MHz com respeito à família *Spartan 3A*) mas para o caso do AD9248-40MSPs será provavelmente irrelevante, contudo como esta implementação funcionou corretamente optou-se por manter este esquema.

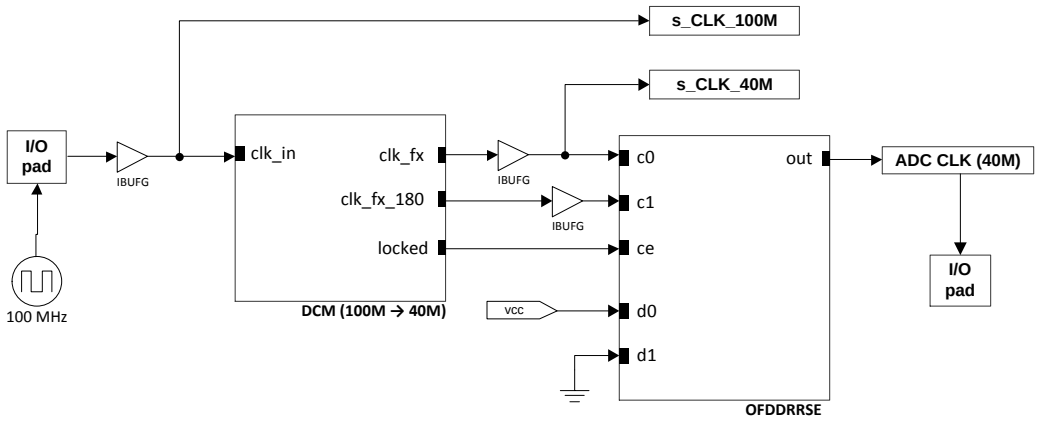


Figura 4.2: DCMs sintetizados, sinais e respectivas ligações.

4.3 Interface com ADC e Computador Hospedeiro

O esquema de interface da FPGA pode ser encontrado na figura 4.3. Como foi referido o sinal de *clock* de 40 MHz é fornecido a ambos os canais da ADC e os dois barramentos de 14 *bits* dão entrada numa sequência de registos dentro da FPGA. Os sinais de OTR (*out of range*) indicam quando existe *overflow* à saída da ADC decorrente de um sinal de entrada fora da gama dinâmica de amplitude. Estes sinais dão entrada na FPGA mas apenas são encaminhados para os LEDs exteriores de modo a fornecer uma indicação visual. O sinal de DCS é utilizado para activar ou desactivar o estabilizador de *duty cycle* dentro da ADC, uma vez que as ADCs *pipelined* são síncronas em ambos os flancos de relógio.

A interface com o computador hospedeiro é feita com recurso ao controlador USB através de um barramento síncrono de 8 *bits* e vários sinais de controlo ambos síncronos pelo relógio de 48 MHz (IFCLK).

4.3.1 Entrada de Dados da ADC

Devido à ausência de sinais que assinalem uma nova amostra disponível nos pinos de entrada da FPGA um dos problemas que surgem é a impossibilidade de prever se os sinais respeitam os tempos de *setup* e *hold* no primeiro registo de entrada dentro da FPGA o que pode trazer consequências nefastas quanto à integridade do sinal e à ocorrência de metaestabilidade dentro dos circuitos lógicos. De modo a minimizar estes possíveis efeitos foi introduzido uma sequência de registos (*flip flops* D).

A figura 4.4 ilustra o *hardware* sintetizado onde, após a passagem pelos registos, é adicionado um valor constante de modo a corrigir o *offset* de amplitude observado à saída da ADC.

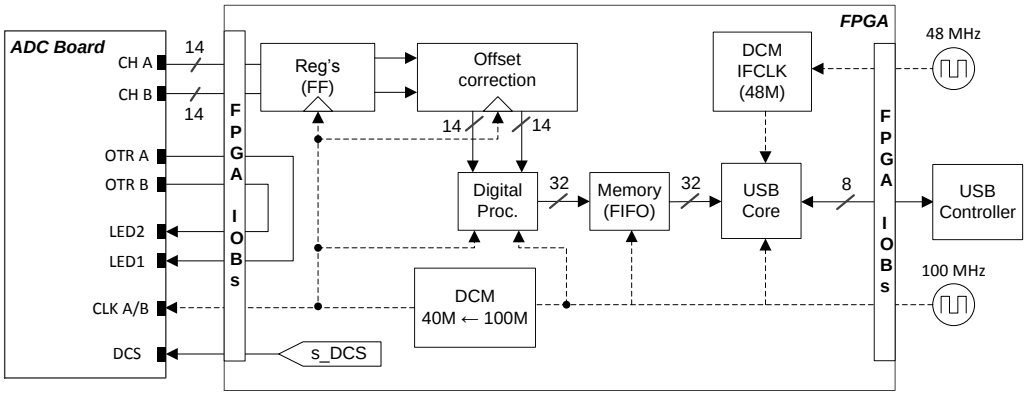


Figura 4.3: Sinais de interface ADC-FPGA: OTR (*out of range indicator*) indica *overflow* da saída da ADC, o sinal é apenas encaminhado para os LEDs da placa como indicador visual. O sinal de DCS (*duty cycle stabilizer*) para a ADC está sempre activo.

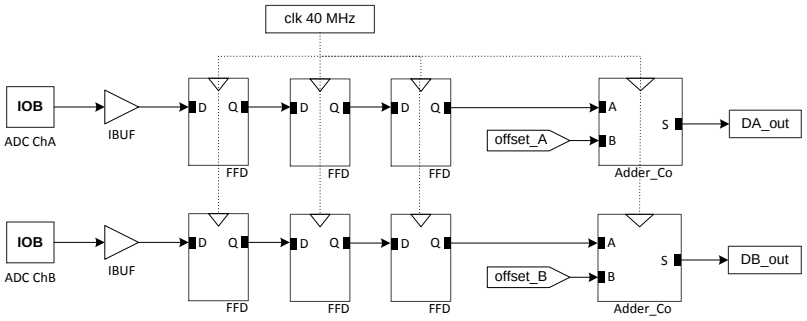


Figura 4.4: Sincronismo do sinal de entrada e correção do offset da ADC com recurso a somadores.

4.4 PLL digital

A figura 4.5 representa o diagrama de blocos simplificado da PLL digital implementada em *VHDL* (referente a apenas um canal por motivos de simplificação, uma esquema completo pode ser encontrado no anexo B). O sinal proveniente da ADC é multiplicado por ambas as saídas do NCO sendo o sinal de saída do mesmo proporcional à diferença de fase entre os dois sinais, depois de removidas as componentes de alta frequência e redução da largura de banda de sinal pelos filtros decimadores CIC e FIR o sinal passa pelo filtro digital onde é integrado ao longo do tempo. Este valor dá entrada no NCO definindo o incremento de fase do mesmo (e consequentemente a frequência), em condições de sincronismo este valor assumirá um valor constante e proporcional ao desvio da frequência central do NCO.

A grande maioria dos blocos do sistema foram sintetizados através da ferramenta `Xilinx Core Generator` podendo-se encontrar no anexo B.2 uma descrição de todos os parâmetros utilizados na síntese dos mesmos. Nas seguintes secções apenas vai ser apresentado o que se considera essencial.

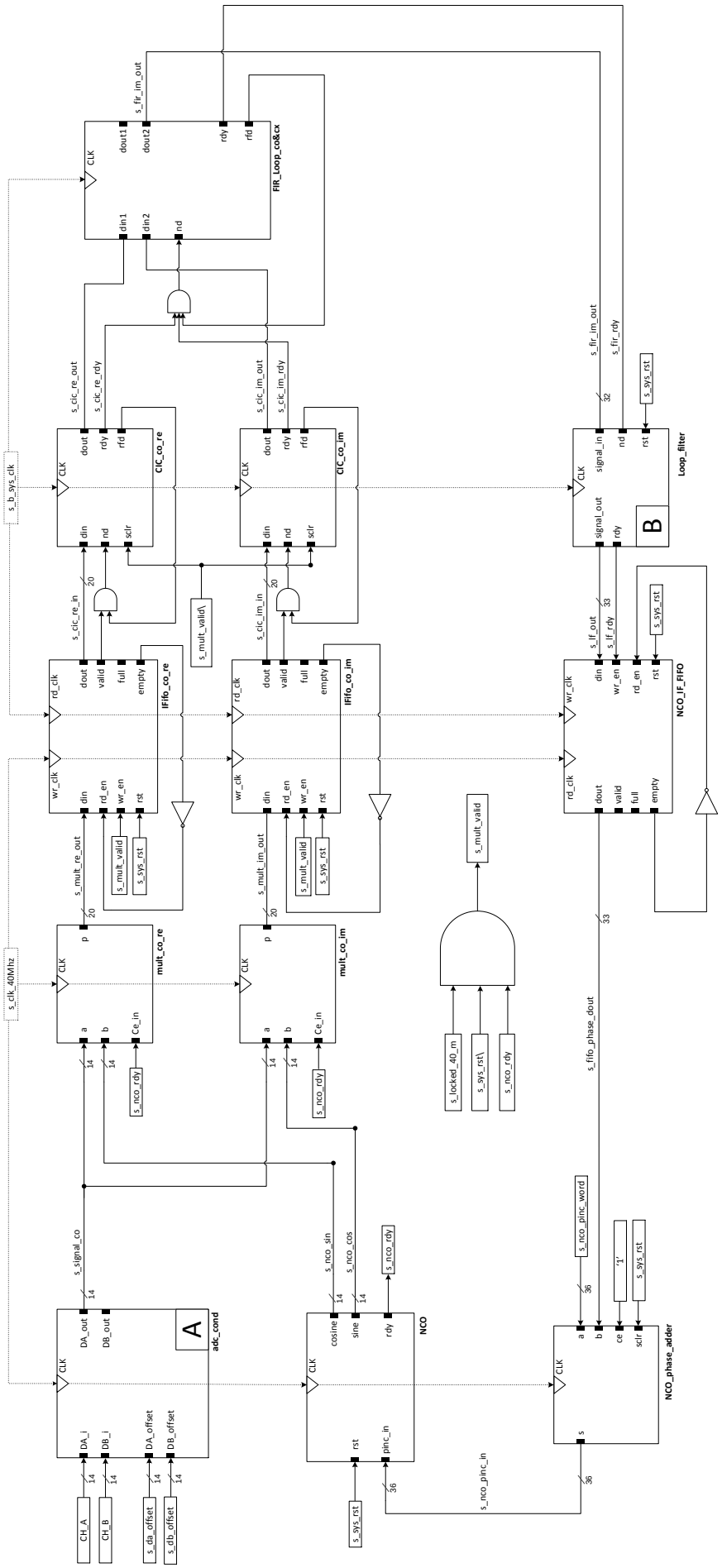


Figura 4.5: Diagrama de blocos da PLL digital sintetizada em VHDL.

4.4.1 NCO

O NCO é um gerador de sinais digitais que cria uma representação em tempo discreto de uma forma de onda, habitualmente uma sinusóide, tratando-se nesse caso do equivalente digital ao VCO. O NCO é composto por dois blocos principais:

- Um acumulador de fases: soma ao valor interno de fase (Θ) o incremento de fase ($\Delta\theta$) presente à entrada do bloco;
- Um conversor de fase para amplitude: usa o valor à saída do acumulador para endereçar uma *Lookup Table (LUT)* que contêm os valores de amplitude correspondentes.

O número de *bits* utilizados para o acumulador de fase (B_Θ) e para o incremento de fase ($B_{\Delta\theta}$) são idênticos e definem a resolução de frequência. O número de *bits* utilizados para a representação da amplitude (B_s) é, regra geral, inferior ao número de *bits* da fase e encontra-se relacionado com a pureza espectral Spurious Free Dynamic Range (SFDR)) do sinal sintetizado.

A frequência de saída do sinal é então dada por:

$$f_{out} = \frac{f_{clk}\,\Delta\theta}{2^{B_\Theta}} \tag{4.1}$$

O NCO sintetizado com base no `Xilinx Core Generator` apresenta os seguintes sinais de interface:

- `PINC_IN` - O incremento de fase $\Delta\theta$;
- `SCLR` - *Reset* Síncrono;
- `SIN, COS` - As saídas respectivas ao sinal sintetizado;
- `RDY` - Indica que as saídas `SIN/COS` são válidas (as saídas apenas não são válidas no processo de inicialização do core após um `reset`).

4.4.2 Detetor de fase (Multiplicadores)

O multiplicador digital funciona sincronamente e multiplica duas entradas *signed* representadas em complemento para dois. As entradas de cada multiplicador consistem num dos canais da ADC (14 *bits*) e numa das saídas do NCO (14 *bits*) de modo a obter as componentes em fase e quadratura para cada canal. O resultado final é truncado a 20 *bits* uma vez que é o número de *bits* máximo de entrada do filtro CIC que se encontra logo à saída do multiplicador.

O único sinal de controlo destes blocos é o sinal de `Clock Enable (CE)`. Este uma vez activo, permite que ambas as entradas `A` e `B` sejam multiplicadas a cada ciclo de relógio sendo o respectivo resultado disponibilizado à saída `(P)` após N ciclos de latência (configurável).

O diagrama temporal com respeito à interface entre NCO e os multiplicadores encontra-se na figura 4.6 e pode ser descrito nos seguintes pontos:

1. Inicialmente (ou após `reset` do sistema) o *clock* de 40 MHz não é válido (representado pelo estado `X`), isto deve-se ao facto de o respectivo `DCM` não ter atingindo o sincronismo.
2. O `DCM` atinge o sincronismo, este evento é sinalizado pela subida do sinal `locked_out`, que por sua vez activa o NCO através do sinal `sclr`;
3. A cada ciclo de relógio o NCO soma à fase interna o valor presente em `pinc_in`. Após N ciclos de latência os valores convertidos pela LUT aparecem à saída, sinalizados pela subida do sinal `RDY` e acompanhados do respectivo valor de fase `phase_out`;

4. O sinal de `RDY` do NCO corresponde ao `clock enable (CE)` dos multiplicadores, os sinais `signal_in_co` e `SIN/COS` são multiplicados a cada ciclo de relógio a partir desse momento.

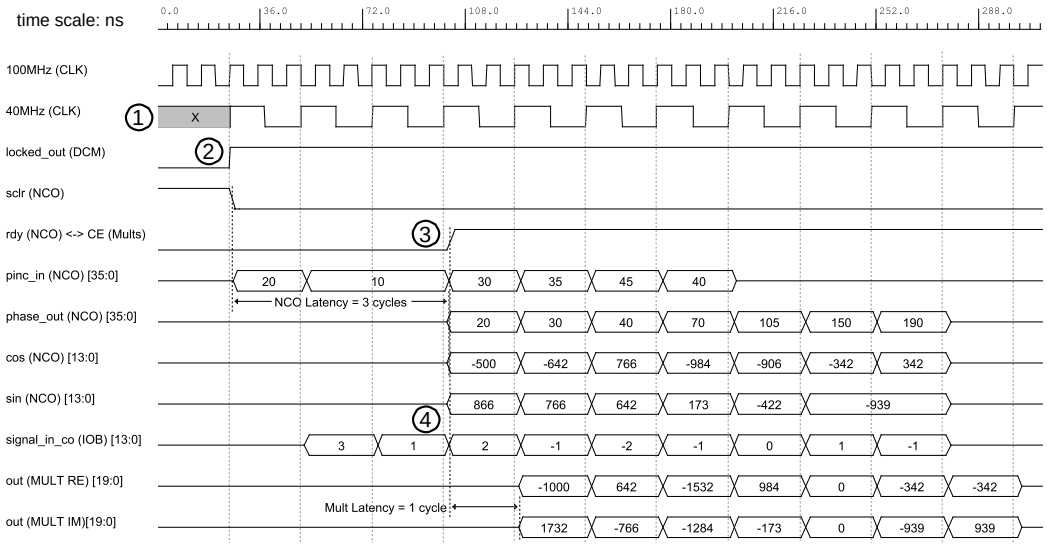


Figura 4.6: Diagrama temporal: DCM(40MHz), NCO, multiplicador

4.4.3 FIFOS de Interface

A saída do multiplicador dá entrada primeiro nos filtros CIC e posteriormente nos filtros FIR. Antes disso, e uma vez que se optou por fornecer a estes blocos um *clock* de frequência mais elevada, foram incluídos FIFOs com *clocks* de leitura (100 MHz) e escrita (40 MHz) independentes para efectuar a transição. Mais uma vez na figura 4.5 é possível observar o esquema de sinais de controlo utilizados:

- `din/dout` - portos de entrada e saída do FIFO respectivamente;
- `wr_clk/rd_clk` - *clocks* respectivos às operações de escrita e leitura (decorrente da configuração com clocks independentes);
- `rd_en` - *strobe* de leitura do FIFO. O próximo valor é lido para a saída `dout` quando o sinal se encontra activo e na transição ascendente de `rd_clk`, uma operação de leitura quando o FIFO se encontra vazio é não destrutiva;
- `wr_en` - *strobe* de escrita do FIFO, a palavra no barramento `din` é escrita no FIFO quando este sinal se encontra activo e no flanco ascendente de `wr_clk`;
- `empty/full` - *flags* de indicação de FIFO cheio ou vazio.

4.4.4 Filtros CIC

Os filtros CIC são uma classe de filtros com resposta impulsional finita cujo objectivo é a implementação eficaz do algoritmo de *moving average* que consiste em efetuar a média ponderada das últimas N entradas do sistema. Este tipo de filtros é particularmente eficaz quando a par da filtragem é necessária uma redução da largura de banda do sinal (através de decimação), situação onde os filtros CIC permitem reduzir em boa quantidade o número de operações aritméticas a realizar.

O esquema de um filtro CIC de decimação pode ser encontrado na figura 4.7. O filtro é composto por `N` estágios que correspondem ao número total de blocos integradores e blocos *COMB*, `R` corresponde ao factor de decimação do filtro e `M` representa o atraso diferencial que corresponde ao número de unidades de atraso na blocos *COMB*.

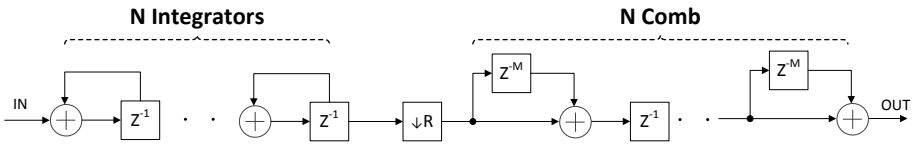


Figura 4.7: Esquema de um filtro CIC decimador de factor R.

As funções de transferência dos filtros no domínio de Z e no domínio da frequência são dadas respectivamente por:

$$H(z) = \frac{\left(1 - z^{-R\,M}\right)^N}{\left(1 - z^{-1}\right)^N} \qquad |H(f)| = \left[\frac{\sin\left(\pi R M f\right)}{\sin\left(\pi f\right)}\right]^N \tag{4.2}$$

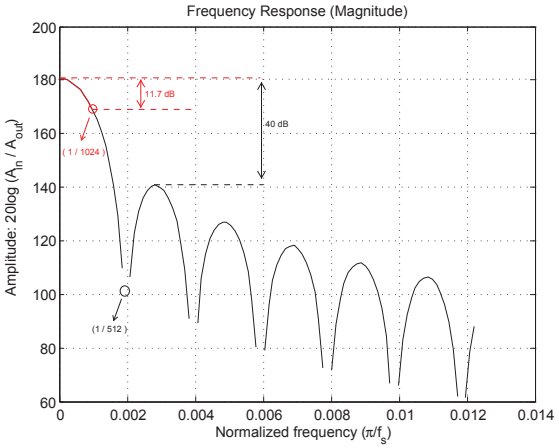


Figura 4.8: Resposta em magnitude do filtro CIC implementado. (R=1024, N=3, M=1)

A resposta do filtro implementado, com um factor de decimação de 1024, encontra-se na figura 4.8. A diferença entre valores máximo e mínimo na banda resultante à saída (a vermelho) é elevada (11.7 dB), no entanto como a largura de banda vai ser reduzida pelo filtro FIR seguinte (R=8) a atenuação máxima imposta pelo filtro CIC na banda final resultante é de apenas 0.16 dB.

Os sinais relevantes para o controlo do bloco `Xilinx LogiCore CIC filter`:

- `DIN/DOUT` - Portos de entrada/saída de dados;
- `ND (New Data)` - A cada transição ascendente de *clock* é carregado o valor de entrada `DIN` para o filtro desde que `ND` se encontre ativo;
- `RFD (Ready For Data)` - Indica a disponibilidade do *core* para o carregamento de novos valores;
- `RDY` - A cada `R` amostras que dão entrada no filtro, o resultado da filtragem é disponibilizada em `dout` e sinalizado com a subida do sinal `RDY` durante um ciclo de relógio.

A figura 4.9 representa a evolução temporal dos sinais a partir da saída do detetor de fase até à saída do(s) filtro(s) CIC:

1. O sinal de `write_enable` do FIFO é activo quando se verificam as condições seguintes: o DCM de 40 MHz se encontra sincronizado (`locked_out = '1'`), as saídas do NCO são válidas (`rdy = 1`) e na ausência de sinal de *reset* do sistema. O sinal `s_mult_valid` é o *and* lógico destas três condições.
2. Sempre que o FIFO não esteja vazio (`empty=0`) é efectuada uma operação de leitura pelo flanco do *clock* de 100 MHz;
3. No ciclo seguinte à operação de leitura a palavra aparece no barramento de saída e é sinalizada através da subida do sinal `fifo_valid`;
4. O sinal de `cic_nd` indica ordem para o carregamento de um novo valor para o filtro CIC e resulta do *and* lógico entre o sinal `fifo_valid` e `cic_rfd`;
5. O sinal `cic_rfd` indica a disponibilidade do *core* para a recepção de um novo valor e o seu comportamento é determinado pelo parâmetro `Input Sampling Frequency` passado para os valores de parametrização do *core*. A definição deste parâmetro com o valor de 40 MHz garante que o core esteja sempre disponível (`cic_rfd=1`) cada vez que o sinal de `valid` sobe;
6. A cada 1024 (factor de decimação do filtro) amostras o filtro CIC disponibiliza um novo resultado no barramento de saída assinalando o evento com a subida do sinal `cic_rdy`.

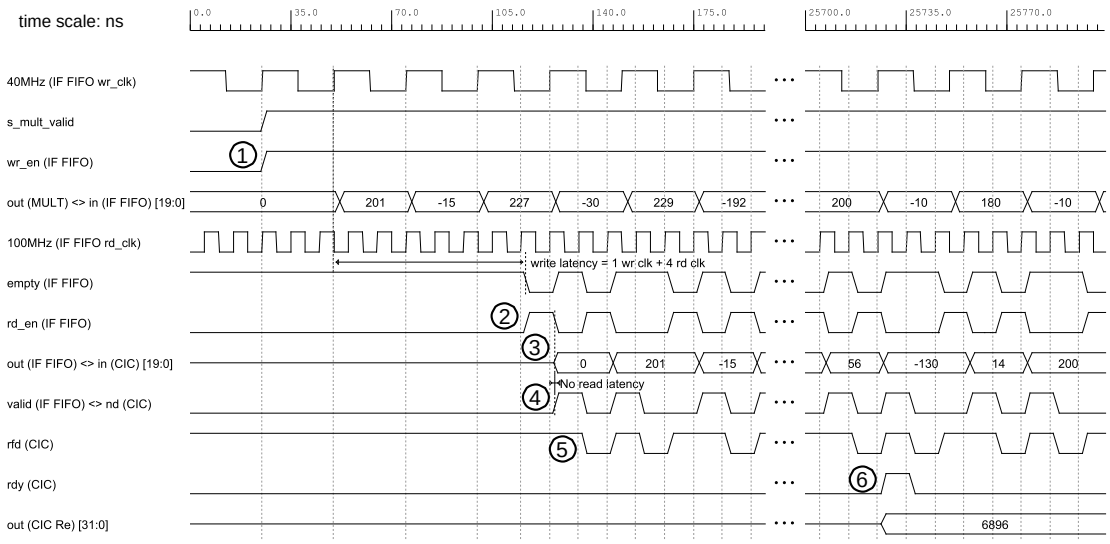


Figura 4.9: Diagrama temporal: Interface multiplicador/filtros CIC.

4.4.5 FIR (Primário)

Em processamento de sinal, um filtro FIR é um tipo de filtro cuja resposta impulsional é de duração finita, uma vez que se torna nula após um determinado tempo, ao contrário dos filtros Infinite Impulse Response (IIR) onde é feita realimentação da saída do filtro. Um filtro FIR com (N+1) coeficientes diz-se de N-ésima ordem uma vez que um dos coeficientes é sempre respeitante à amostra mais recente.

Apesar de os filtros IIR requererem menor número de coeficientes e de operações aritméticas

para filtragens semelhantes, estes apresentam também as seguintes desvantagens:

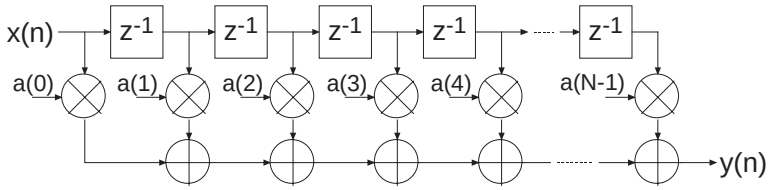


Figura 4.10: Implementação de filtros FIR com linha de atraso. [Xil12]

- Resposta em fase não linear. O atraso que o filtro impõe depende da frequência do sinal de entrada. Por outro lado o atraso imposto pelos filtros FIR é constante e igual a $(N/2)\,F_s$ sendo F_s a frequência de amostragem do sinal;
- São inerentemente instáveis devido ao facto de existir realimentação da saída.

Para evitar possíveis distorções do sinal, decorrentes da variação em frequência do sinal IF optou-se pela implementação de filtros FIR com resposta em fase linear. Para isso basta fazer os coeficientes dos filtros simétricos e em número par (ordem ímpar) de modo a permitir a simetria. A resposta dos filtros implementados encontra-se na figura 4.11, detalhes da implementação podem ser encontrados no Anexo B.

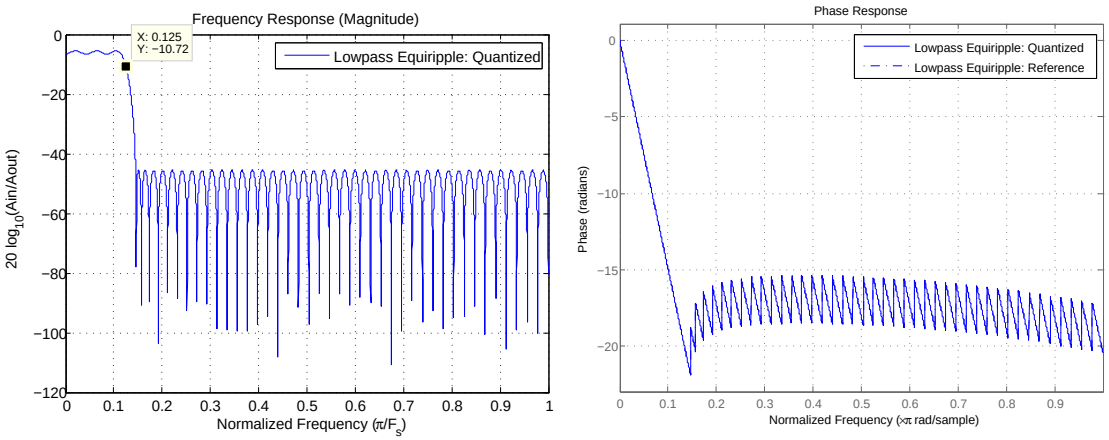


Figura 4.11: Filtro FIR (primário), resposta em amplitude e fase.

A interface entre os CIC e FIR cores é bastante simples bastando carregar uma amostra para este último quando é disponibilizada uma nova pelo primeiro, deste modo o sinal `fir_nd` é o resultado do *'and'* lógico entre `cic_rdy` e `fir_rfd`. A cada 8 amostras (factor de decimação) carregadas, o FIR calcula um novo resultado, disponibiliza os dados à saída e assinala o acontecimento através do sinal `fir_rdy` o que acontece a cada 8*25.6 μs.

4.4.6 FFT

O bloco *IPLogiCore* FFT implementa o algoritmo *Cooley-Tukey*, um método computacionalmente eficaz para calcular DFTs:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-jnk2\pi} \tag{4.3}$$

O *core* permite o cálculo de DFTs complexas apresentando entradas para as partes real e imaginária do sinal. Os sinais de controlo e modo de operação dependem do tipo de arquitetura escolhida no *FTT compiler* existindo fundamentalmente dois tipos:

- *Streaming I/O* - Permite a execução em paralelo das diferentes fases de execução, o carregamento de novas amostras é contínuo ao mesmo tempo que é efectuado o cálculo dos coeficientes e o *unload* dos resultados. Deste modo é a arquitectura que permite maior débito de dados, porém também é a mais exigente em termos de recursos utilizados ocupando até oito vezes mais *XtremeDSP slices* que uma arquitectura *Burst I/O*.
- *Burst I/O* - As fases de carregamento de dados, cálculo dos coeficientes e `unload` ocorrem em série e em janelas temporais contíguas. Em termos de sinais de controlo apresenta uma entrada adicional (unload) para controlar a saída de um *set* de dados.
- em ambas as arquiteturas tanto os dados de entrada como de saída são apresentados em série.

Uma vez que a taxa de dados proveniente dos Filtros FIR é relativamente baixa (4.9 KS/s) optou-se pela arquitectura *Radix-2 Lite Burst I/O*, a mais económica em termos de recursos. O procedimento para processar um *set* de dados através do *FFT IP Core* (Burst I/O) encontra-se descrito nos pontos seguintes com o respectivo diagrama temporal na figura 4.12:

1. Activar sinal de *clock enable* `ce`, dar início ao carregamento de dados `(fft_start = 1)` esperar pelo sinal de *ready for data*; `fft_rfd=1`;
2. Após *start* e a cada tique de relógio (100 MHz) são carregados os valores (real e complexo) à entrada do bloco desde que `ce=1` e `rfd=1`. A saída `xn_index` indica o índice actual;
3. No final da fase de *Load* quando são carregadas N valores a saída `rfd` desce e o sinal `busy` sobe indicando o cálculo da FFT, a saída `edone` indica a conclusão do processamento;
4. Activar a entrada `unload`, o sinal `dv` *(data_valid)* indica a presença de dados válidos no barramento `fft_xk_(re/im)`.

De notar que ao contrário dos CIC/FIR Cores não existe uma entrada para indicar novos dados, o controlo neste caso deverá ser feito com recurso ao `ce` que deverá ser apenas activo quando existir nova amostra à saída do bloco FIR (`FIR_RDY=1`).

A gestão do bloco da FFT, tal como foi apresentado, foi implementada através de uma máquina de estados finita cujo diagrama se encontra na figura 4.14. O diagrama de interface dos sinais do *core* encontra-se na figura 4.13.

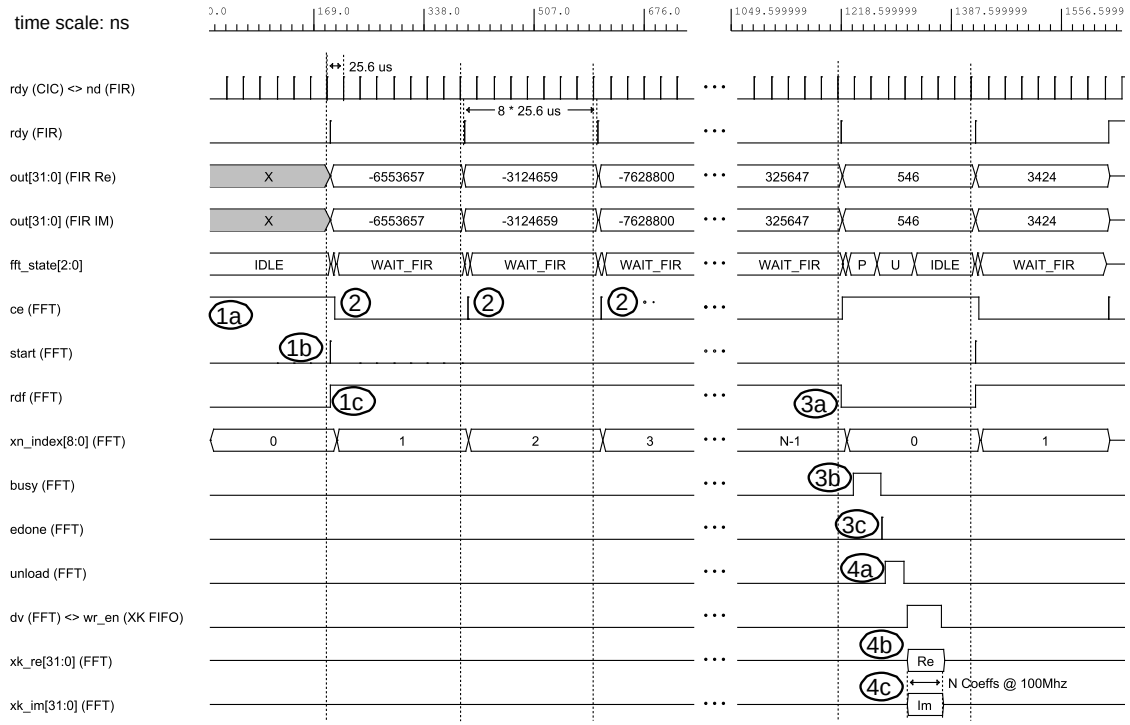


Figura 4.12: Diagrama temporal: Interface entre filtro FIR e *Core* FFT. (`fft_states:` **(S)**ample; wait **(P)**rocessing; **(U)**nload)

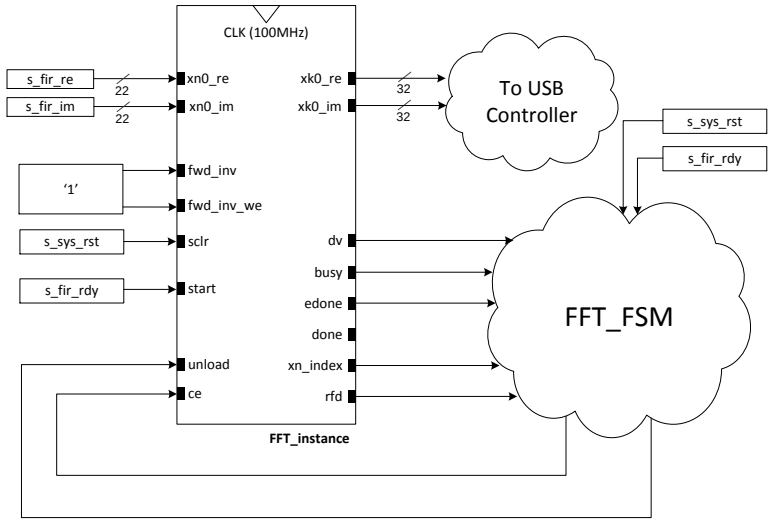


Figura 4.13: Interface do bloco FFT.

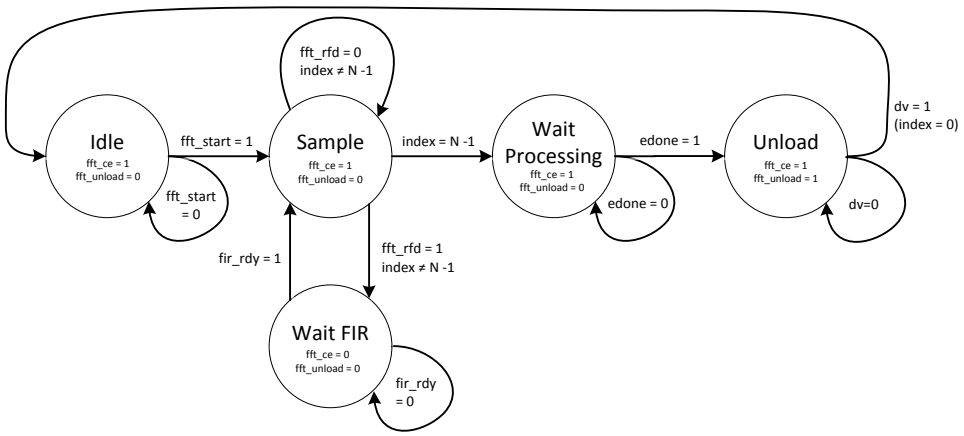


Figura 4.14: Diagrama de transições da máquina de estados implementada para o FFT IP core.

4.5 Dimensionamento da PLL - Conversão para o domínio de Z

O filtro implementado foi abordado na seção 2.4.0.2 (2ª ordem com integrador perfeito e correcção de avanço de fase) e apresenta a seguinte função no domínio de `s`.

$$F(s) = \frac{s\tau_2 + 1}{s\tau_1} \tag{4.4}$$

Como estamos perante uma implementação digital é necessário converter a equação anterior para o domínio de Z. A relação entre os dois domínios é não linear contudo é possível recorrer à série de Taylor e obter a seguinte aproximação.

$$z^{-1} = e^{-Ts} \approx 1 - sT \tag{4.5}$$

Onde T_s representa o período dos dados à entrada do filtro (0.205 ms), substituindo em 4.4 obtêm-se:

$$F(z) = \frac{\tau_2}{\tau_1} + \frac{T_s}{\tau_1}\left(\frac{1}{1-z^{-1}}\right) = \frac{1}{C_1} + \frac{1}{C_2}\left(\frac{1}{1-z^{-1}}\right) \tag{4.6}$$

Uma realização física possível e conveniente para este filtro encontra-se na imagem 4.16, a saída é a soma de um ramo proporcional com um ramo integrativo com ganhos diferentes.

Para o cálculo das constantes dos divisores C_1 e C_2 é necessário ter em conta o ganho de todos os blocos que fazem parte do *loop*. A tabela 4.1 apresenta os valores resultantes da implementação realizada.

Considerando uma largura de banda de ruído de 50 Hz (B_L) suficiente para acomodar em frequência toda a largura do sinal a detectar e um coeficiente de amortecimento de 0.707 (ζ) os coeficientes podem ser calculados recorrendo ao seguinte conjunto de equações:

Parâmetro	Valor	Parâmetros do `IPcore` / comentários
A_{in}	$2^{13}-1$	Amplitude do sinal de entrada (digital)
B_L	50 Hz	
ζ	0.707	
w_n	94.28 rad/s	
G_{mult}	16 Ain	$A_{NCO} = 2^{12}$, $B = 8$
G_{cic}	4096	R=1024, $N = 3$, $M = 1$, $B = 18$
G_{fir}	0.5	$G_{coeff} = 2^{17}$, $B = 18$
k_d	536168365	
k_o	0.582 mHz	Phase bits = 36, Amplitude bits = 14 (Unit Circle)
τ_1	35.1102 s	
τ_2	0.0150 s	
C_1	2341	Divisor do ramo proporcional
C_2	171440	Divisor do ramo integrativo

Tabela 4.1: Dimensionamento da PLL efetuada para uma amplitude do sinal de entrada de 8191 (valor máximo para a ADC utilizada).

$$\tau_1 = \frac{K_o\, K_d}{{\omega_n}^2} \qquad K_d = G_{mult} \cdot G_{cic} \cdot G_{fir}$$

$$\tau_2 = \frac{2\zeta}{\omega_n} \qquad K_o = \frac{f_{nco}}{2^{B_{\theta(n)}}}$$

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right)$$

K_o representa o ganho de fase do NCO, f_{NCO} o *clock* de entrada e $B_{\theta(n)}$ o número de *bits* utilizados para representação da fase interna do NCO. A constante K_d neste caso não representa apenas o ganho do detetor de fase mas também dos filtros CIC e FIR nas respectivas bandas de passagem, de um modo geral os respectivos valores podem ser calculados através de:

$$G_{mult} = \frac{A_{nco}\, A_{in}}{2^B} \qquad G_{cic} = \frac{(RM)^N}{2^B}\frac{1}{2} \qquad G_{fir} = \frac{G_{coeff}}{2^B} \tag{4.7}$$

Em que B representa o número de *bits* truncados em cada um dos blocos e G_{coeff} o ganho total do filtro FIR dependente dos coeficientes usados. O factor de 1/2 no cálculo de G_{cic} é referente à rejeição da frequência imagem resultante da mistura no multiplicador, R/M e N são os parâmetros do filtro CIC. O ganho do multiplicador depende de ambos os operadores de entrada o que significa que os parâmetros do *loop* dependem da amplitude do sinal a detectar, por sua vez isto implica que na ausência de um *Automatic Gain Control (AGC)* que mantenha constante o sinal de entrada as características do loop (w_n,B_L) variam conforme a atenuação do sinal. Este facto representa um aspecto indesejável numa situação real de detecção.

Deste modo o dimensionamento da PLL é apenas válido para a amplitude de entrada considerada nos cálculos (8191, amplitude máxima da ADC).

4.5.1 Filtro de malha

O diagrama de blocos da última parte do *loop* encontra-se na figura 4.16. À saída do filtro é adicionado o valor de fase constante que define a frequência central e inicial do NCO.

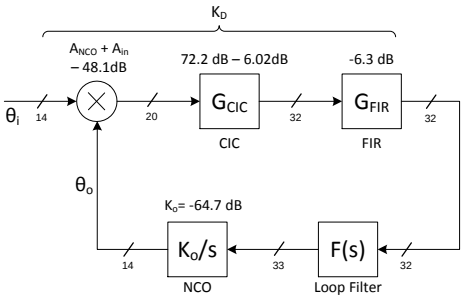


Figura 4.15: PLL digital em FPGA: Ganhos.

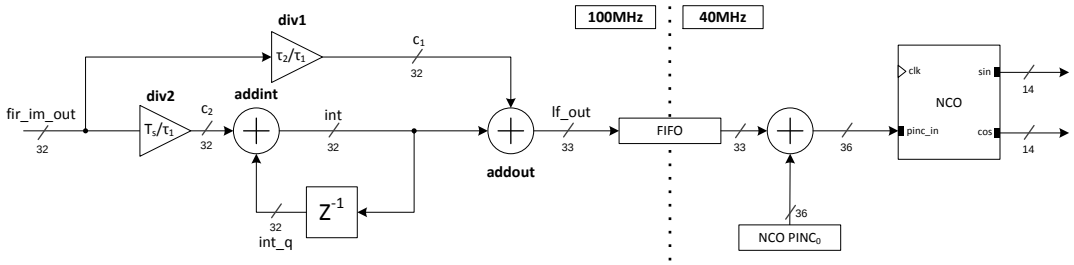


Figura 4.16: Diagrama de blocos simplificado - Filtro de malha PLL.

1. Sinal `fir_rdy` indica nova amostra presente no barramento;
2. A saída do bloco de atraso z^{-1} é actualizada com valor da entrada, os sinais de `new_data (ND)` para os divisores são activos;
3. É aguardado pelo resultado dos divisores (`dif_rdy=1`);
4. O sinal de `clock enable` do somador `addint` é activo durante um ciclo de relógio, o resultado da soma fica disponível no ciclo seguinte (latência=1);
5. Mesmo procedimento que (4) mas para o somador `addout`;
6. É enviado para o exterior a indicação de nova amostra disponível (`lf_rdy=1`).

O diagrama detalhado resultante da implementação em VHDL, diagrama temporal dos sinais e diagrama da máquina de estados encontram-se nas figuras 4.17, 4.19 e 4.18, respectivamente.

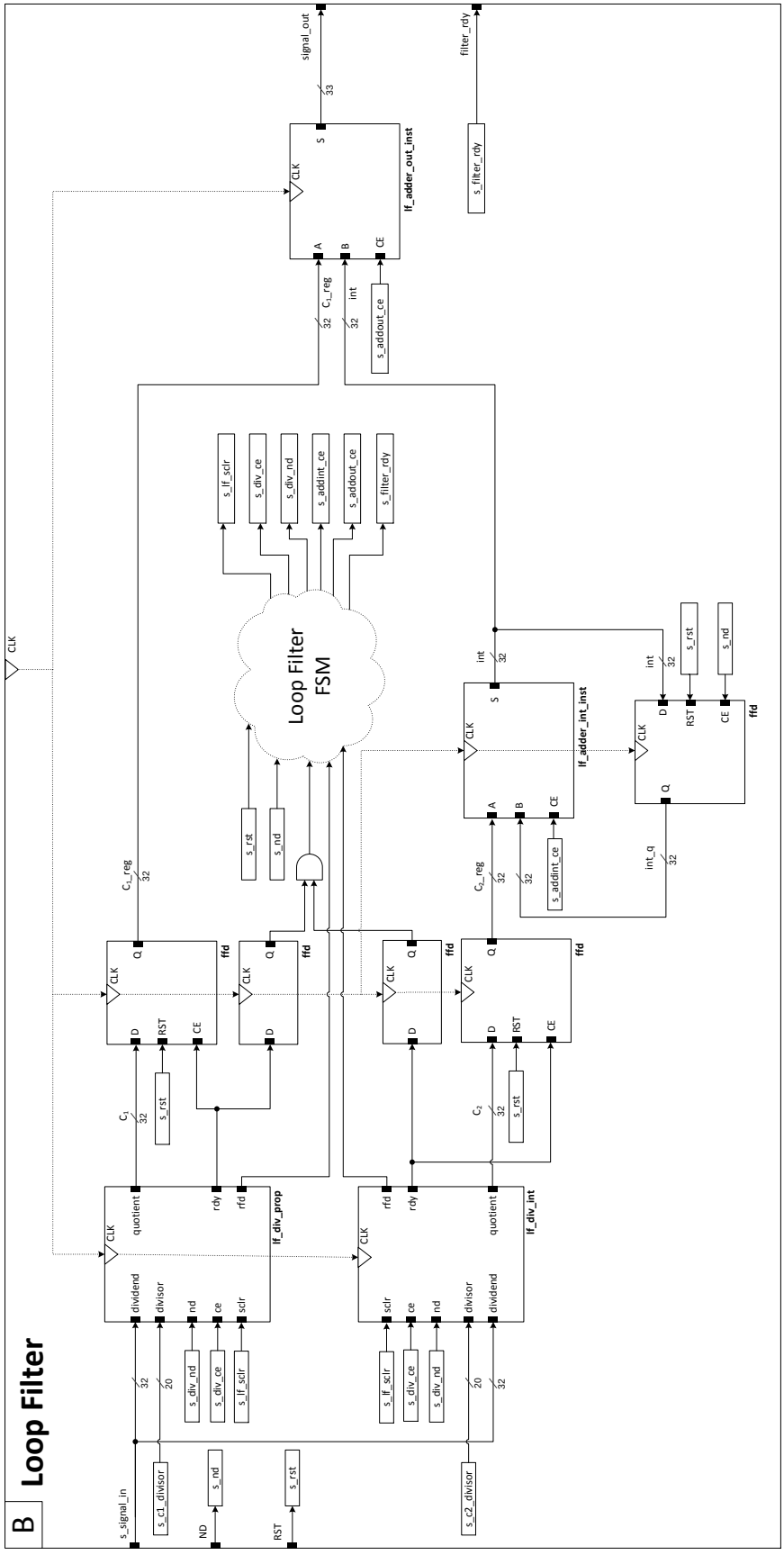


Figura 4.17: Filtro de Malha, diagrama de blocos e de ligações.

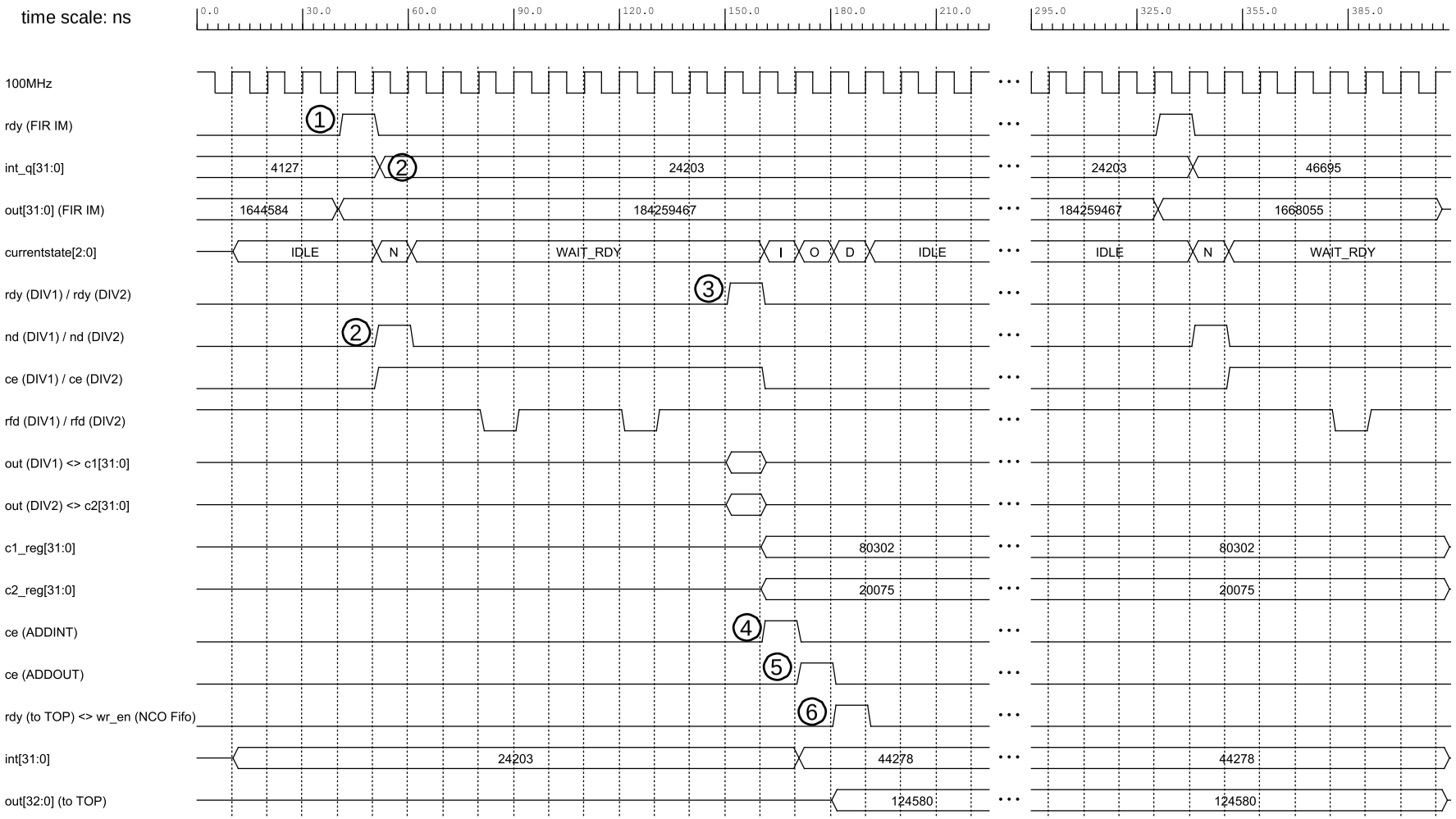


Figura 4.18: Diagrama temporal: Filtro de malha, processamento de uma amostra.

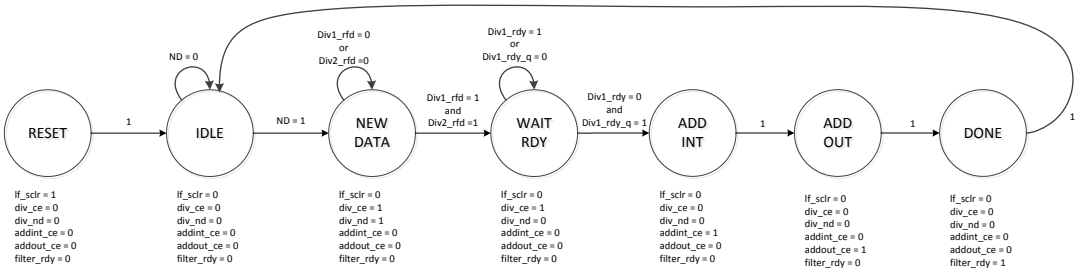


Figura 4.19: Filtro de Malha, diagrama de transições da máquina de estados.

4.6 Canal Crosspolar

A implementação de um segundo canal para a detecção do canal *crosspolar* é em tudo idêntica ao do canal *copolar*. O único aspecto a destacar é a utilização do mesmo NCO para a detecção do segundo canal tal como se pode observar na figura 4.20.

A sincronização do NCO é feita pelo canal *copolar* uma vez que tem maior amplitude. O sincronismo que seria obtido através do sinal *crosspolar* (cerca de 30 dB abaixo) seria muito mais precário dada a reduzida CNR.

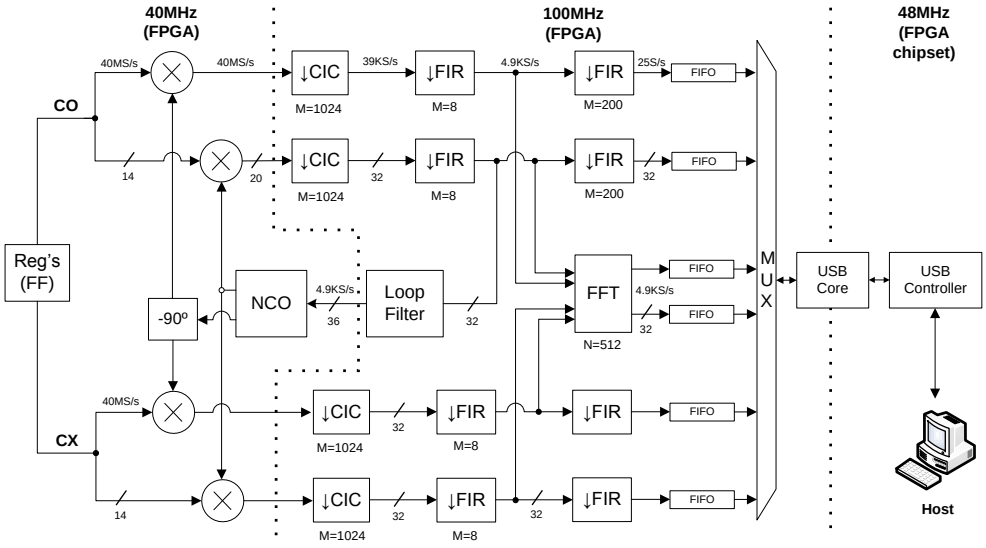


Figura 4.20: Diagrama de blocos geral do sistema, inclusão do canal crosspolar.

Os resultados da detecção: as componentes I/Q no domínio do tempo, quer a detecção no domínio da frequência são armazenados temporariamente em FIFOs antes de serem transferidos para o PC, o próximo capítulo será dedicado à implementação das comunicações.

Capítulo 5

Comunicação em tempo real com o Host via USB

5.1 Introdução

O esquema da comunicação entre FPGA e computador é apresentado na figura 5.1. As componentes detetadas apresentam diferentes taxas de transmissão (I/Q: 25 S/s, FFT 4.9 kS/s) síncronizadas pelo relógio de 100 MHz, o controlador USB opera síncronamente pelo sinal de relógio `IFCLK` (48 MHz). De modo a efetuar a mudança de frequência foram implementados vários FIFOs com *clocks* de leitura e escrita independentes. Ao mesmo tempo o armazenamento de dados permite minimizar o número de transferências a efetuar o que é benéfico para o sistema uma vez todas as operações USB são necessariamente iniciadas pelo computador hospedeiro e a latência das operações pode ser bastante elevada (ordem dos segundos) dependendo da carga do sistema computacional.

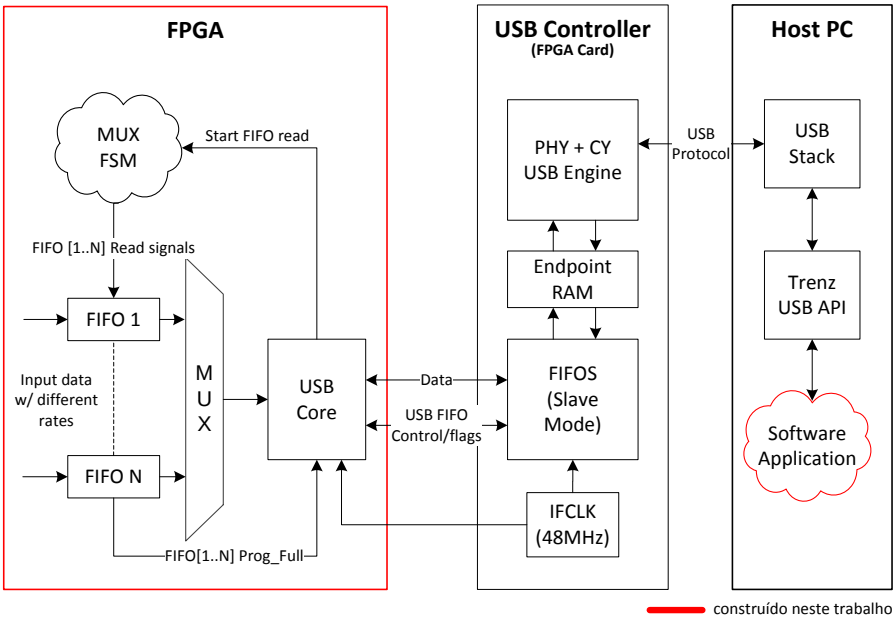


Figura 5.1: Esquema geral da comunicação FPGA ↔ computador hospedeiro.

Do lado do computador a interação com o controlador é feita com base na *Application Pro-*

grammable Interface (API) disponibilizada pela *Trenz* [Ele12a], fornecendo esta um conjunto de funções para as operações de leitura, escrita, operações de acesso ao barramento I2C, etc. As instruções para o uso da API assim como a instalação do *firmware* respectivo para o controlador podem ser encontradas no anexo C.

Os principais obstáculos no estabelecimento de comunicações com base neste controlador podem-se resumir nos seguintes pontos;

1. Todas as operações USB são iniciadas pelo computador hospedeiro, o que impede que a FPGA tome a iniciativa de enviar dados;
2. As operações através da API, na realidade, apenas permitem a leitura/escrita de dados nos FIFOs internos do controlador;
3. Os FIFOs internos são configurados em *slave mode* (via *firmware*), o que significa que a FPGA é responsável pela geração dos sinais de controlo para os mesmos.
4. Uma operação de leitura só é bem sucedida se o número de *bytes* a ler corresponder ao número de *bytes* escritos pela FPGA, isto obriga de antemão a saber a quantidade de dados a transmitir;

As soluções respectivas para cada um dos problemas descritos, e que acabam por justificar o esquema adotado na figura 5.1.

1. Antes de a uma operação de leitura dos FIFOs (FPGA) é enviado primeiro um pedido de `request` dos FIFOs (FPGA) a serem lidos;
2. Após início da operação de leitura, a FPGA tem um tempo `timeout` (definido na API) para a escrita dos dados nos FIFOs internos do controlador. A informação é depois automaticamente encapsulada com o protocolo USB pelo controlador e lida continuamente pelo PC;
3. O *Core* USB adaptado da *Trenz* permite a gestão da interface;
4. Os FIFOs da FPGA foram implementados com *thresholds* que indicam uma ocupação superior a metade, estes *thresholds* são periodicamente lidos pelo PC, sempre que um *threshold* seja atingido é efetuado a leitura do número de *bytes* correspondente ao mesmo.

As próximas secções serão dedicadas à arquitectura e modo de funcionamento do controlador USB, estrutura do *USB core* na FPGA e *software* implementado no computador.

5.2 Estrutura geral do controlador USB (Fx2)

A figura 5.2 representa a arquitectura geral do controlador *Cypress EZ-USB* Fx2.

- **USB PHY** - Responsável pela adaptação dos sinais físicos do USB, aspectos como a modulação, *pulse shaping* (NRZI), multiplexagem (3-*state buffers*) são implementados neste bloco;
- **PHY Interface (SIE)** - Um dos blocos mais importantes do controlador, efetua a descodificação dos pacotes (PID), correção de erros através dos bits de CRC e encaminha o payload de dados para os *endpoints* respectivos;
- **Micro controlador 8051** - O objectivo principal do microcontrolador não é interferir na transferência direta do *payload* de dados. Embora o possa fazer as velocidades de transferência são grandemente afetadas. A principal tarefa consiste no atendimento de *requests* de controlo no *endpoint* 0 que não são críticos em termos de largura de banda. Operações de controlo típicas do USB: Get Descriptor (identificador do dispositivo USB), Set Address,

etc. A maioria destas transferências de controlo ocorrem no processo de enumeração do dispositivo. Todas as opções de configuração do controlador são controladas através de registos especiais acessíveis a partir do 8051. O controlador também pode desempenhar funções de uso geral, dispondo de portos de leitura/escrita e interface I2C.

- **4KB RAM** - Memória dedicada ao armazenamento do *payload*, referente a cada um dos *endpoints* (EP0, EP1, EP2, EP4, EP6, EP8);
- **0.5KB RAM** - Memória para armazenamento das *waveforms* para a *General Programmable InterFace (GPIF)*;
- **USB Regs** - Registos de configuração do controlador USB;
- **16KB RAM** - Memória de programa (*firmware*), em caso de presença de uma EEPROM no barramento I2C (caso do módulo TE0320), e após *power-on* ou *reset*, o *bootloader* do 8051 transfere o conteúdo da EEPROM para a memória interna do microcontrolador, passando a executar o respectivo código a partir daí;
- **FIFOS** - Memória para transferências de alta velocidade (480 Mb/s, *High Speed* USB) que permitem a entrada/saída de dados sem a intervenção do microcontrolador. Estes FIFOs funcionam em modo *slave*, podendo o *master* ser uma entidade exterior ou o módulo GPIF interno ao controlador.

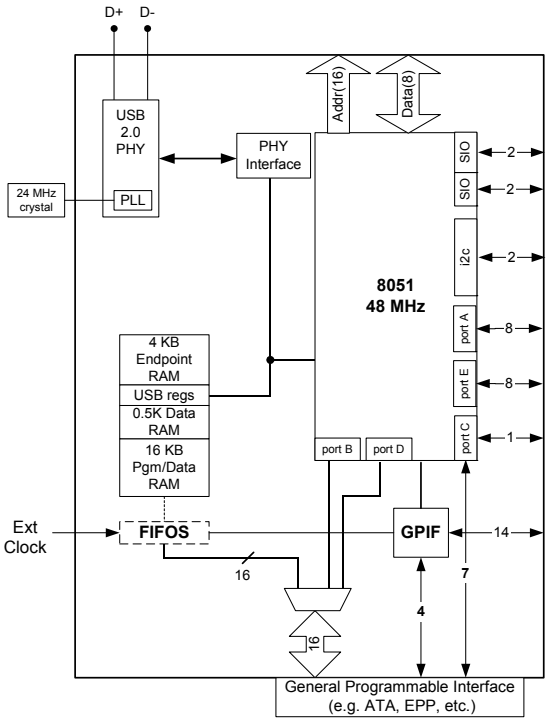


Figura 5.2: Diagrama interno do controlador USB Cypress Ez-USB Fx2. [Sem12]

5.2.1 Configuração via Firmware

O *firmware* (TE-USB-FX2 v3.xx) utilizado foi desenvolvido pela *Trenz*, as funções mais importantes implementadas pelo mesmo são descritas a seguir:

1. Informação de enumeração do dispositivo;

2. Configuração dos *endpoints* conforme a tabela 5.1, os *endpoints* EP0, EP1OUT e EP1IN não têm FIFOs associados;
3. FIFOs relativos aos *endpoints* EP2, EP4, EP6 e EP8 encaminham diretamente (AUTOIN/AUTOOUT) a informação para o SIE sem intervenção do CPU;
4. FIFOs configurados em *slave mode* através de *master* exterior (neste caso FPGA), síncronos com o relógio interno de 48 MHz;
5. *Polling* do *endpoint* 1 e execução de funções específicas; todos os comandos enviados a partir de `TE_USB_FX2_SendCommand` (API) são enviados para este *endpoint* e atendidos pelo 8051.

Endpoint	**RAM**				**FIFO**	
	Size	**Dir.**	**Type**	**Buf.**	**Mode**	**E. Flag**
EP0	64	I/O	CTRL	1x	N.A.	N.A.
EP1IN	64	I	BULK	1x	N.A.	N.A.
EP1OUT	64	O	BULK	1x	N.A.	N.A.
EP2	512	I	BULK	2x	AUTOIN	FULL
EP4	512	I	BULK	2x	AUTOIN	FULL
EP6	512	I	BULK	2x	AUTOIN	FULL
EP8	512	O	BULK	2x	AUTOOUT	EMPTY

Tabela 5.1: Configuração dos endpoints do controlador USB Fx2 via firmware.

5.2.2 Interface Slave FIFOs

Os *slave* FIFOs encontram-se ligados fisicamente à FPGA na placa `TE0320`, para a configuração específica do *firmware* indicado, têm o seguinte significado:

- `FD[7:0]` - Barramento de dados bidireccional;
- `FIFOADR[1:0]` - Selecção do *endpoint*;
- `SLRD,SLRW` - *Strobes* de leitura e escrita (*Active High*);
- `SLOE` - *SLave Output Enable*, configuração dos 3-*state buffers* para permitir a leitura ou escrita de dados;
- `FLAGB` - *Flag* para indicação de FIFO cheio (correspondente ao indicado por `FIFOADR`);
- `FLAGC` - *Flag* para indicação de FIFO vazio (correspondente ao indicado por `FIFOADR`);
- `FLAGD` - *Flag* de indicação de FIFO vazio (correspondente ao EP8 (OUT));
- `PKTEND` - Sinal para o envio forçado de um pacote `IN`[1];
- `ICLK` - Sinal de relógio por quais todos estes sinais são síncronos.

As operações do tipo `OUT` são bastante fáceis de realizar do ponto de vista da FPGA, é apenas necessário observar quando o *slave* FIFO contêm dados (FLAGD) e proceder à leitura do mesmo pela activação dos sinais `SLRD` e `SLOE`.

[1]Uma vez as operações de USB são relativas sempre ao *master* (*Host PC*) um pacote IN é um pacote no sentido FPGA → PC.

Operações do tipo `IN` são um pouco mais complicadas uma vez que os *slave* FIFOs só são lidos para a RAM do *endpoint* desde que seja atingido o valor de *bytes* máximo de pacote[2]. Pacotes inferiores ao valor máximo só serão enviados com a activação de `PKTEND`. Adicionalmente, e sob o ponto de vista da FPGA, os *slave* FIFOs apresentam uma capacidade máxima igual ao valor máximo de pacote referente ao *endpoint* em causa.

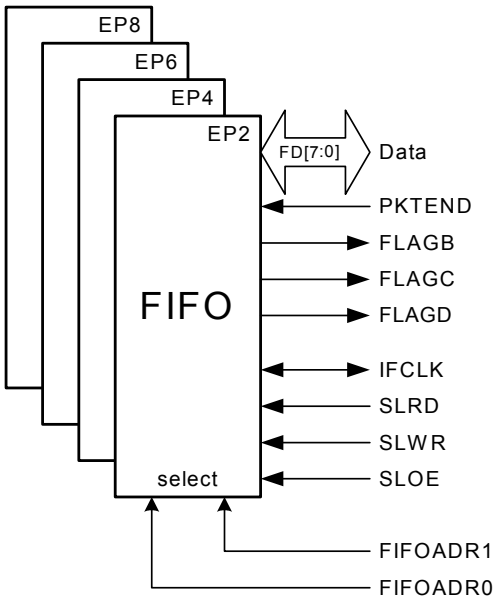


Figura 5.3: FIFOs Internos do controlador Fx2 + sinais de interface para configuração em Slave Mode.

5.2.3 Interface I2C

Como já foi referido a interface I2C é utilizada juntamente com o *endpoint* 1 para o envio e recepção de dados entre FPGA e PC. Esta interface foi utilizada para realizar operações de controlo como vai ser explicado na secção seguinte. Os sinais de interface são os típicos do protocolo I2C havendo uma linha de dados série `SDA` e um sinal de *clock* `SCL` de 100 kHz. Ambos os sinais funcionam numa configuração *open drain* pelo que do lado da FPGA é necessário recorrer a primitivas `IOBUF` para gerar o estado de alta impedância na linha.

Uma vez que controlador interno 8051 é o *master* do barramento, as operações de leitura e escrita são iniciadas por este, a API permite a realização remota destas operações através do *endpoint* 1 e a partir do computador hospedeiro.

5.3 Implementação

Em termos de *software* foi construída uma pequena aplicação gráfica em QT que permite algumas funções de controlo básicas sobre a FPGA, e receber/guardar os dados resultantes da mesma. A figura 5.4 representa o fluxo da comunicação:

1. Do lado da FPGA, é escrito num registo o estado dos *thresholds* relativos aos FIFOs internos;

[2]512 bytes relativamente aos *endpoints* IN-BULK (EP2,EP4,EP6) e para o *firmware* referido

2. O computador hospedeiro, através de um *timer* de *software*, efetua periodicamente o *polling* desse registo;

3. Se existirem FIFOs com *thresholds* atingidos é enviado um pedido de leitura dos mesmos seguida de uma operação de leitura;

4. A máquina de estados `MUX_FSM` efetua a leitura dos FIFOs internos e a escrita do seu conteúdo nos FIFOs do controlador USB de modo a completar a operação de leitura.

A razão da utilização do barramento I2C prende-se com a necessidade de separar o caminho da informação propriamente dito da informação de controlo, caso contrário seria necessário implementar algum tipo de protocolo muito simplificado para fazer a distinção. Adicionalmente, um *core* VHDL da *Trenz* foi utilizado para responder aos pedidos de leitura/escrita via I2C.

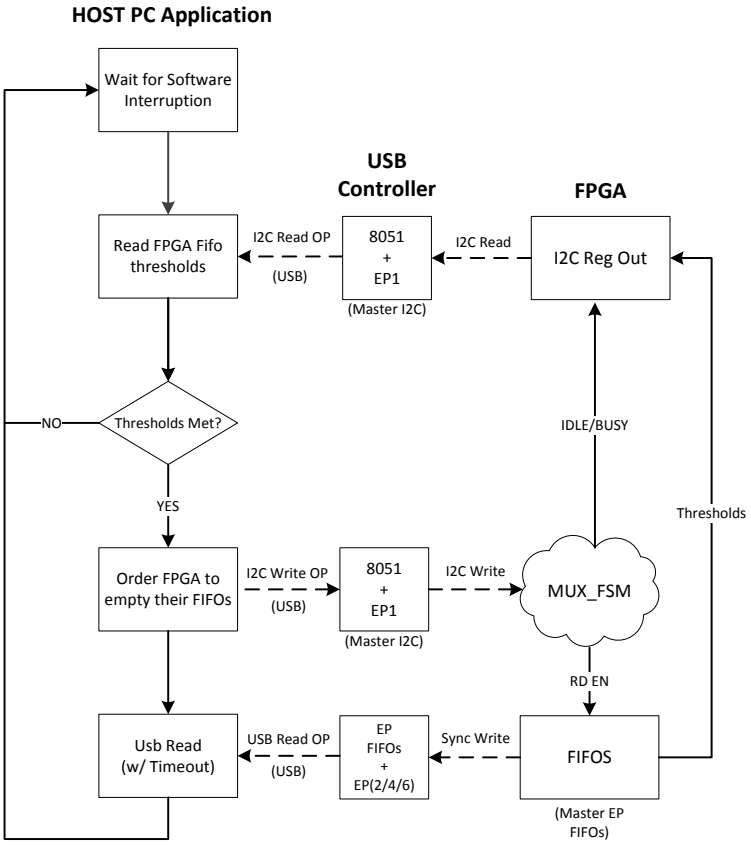


Figura 5.4: Diagrama de comunicações: Computador hospedeiro ↔ FPGA

5.4 Multiplexagem dos FIFOS da FPGA

De seguida é apresentado o diagrama temporal da multiplexagem dos FIFOs da FPGA que ocorre a partir do momento em que chega uma operação de *request* via I2C. A multiplexagem é efetuada com recurso a uma máquina de estados `MUX_FSM` que utiliza um contador para controlar o número de *bytes* transmitidos. A evolução temporal dos sinais encontra-se na figura 5.5.

1. Os FIFOs inicialmente têm as *flags* de *threshold* `full` activas, e a máquina de estados encontra-se inactiva `IDLE`;
2. Após polling do PC (não representado na figura) é recebido o pedido de *request* pelo mesmo, a subida de `interrupt (I2C_slave)` indica a presença de novos *bytes* recebidos;
3. É efetuado o início da máquina de estados `MUX_FSM` através do sinal `start_fx2_tx`;
4. `MUX_FSM` efetua a leitura dos FIFOs através do sinal de `rd_en`, ao mesmo tempo efetua o *setup* e o *reset* do contador de modo a ler o número de *bytes* correspondentes ao *threshold*;
5. O sinal de `valid (FIFO)` indica que a leitura foi efetuada e que os dados estão presentes no barramento de saída do FIFO, ao mesmo tempo este sinal funciona como *strobe* de escrita para o *core* USB abordado mais adiante `(fx2_write_enable)`.

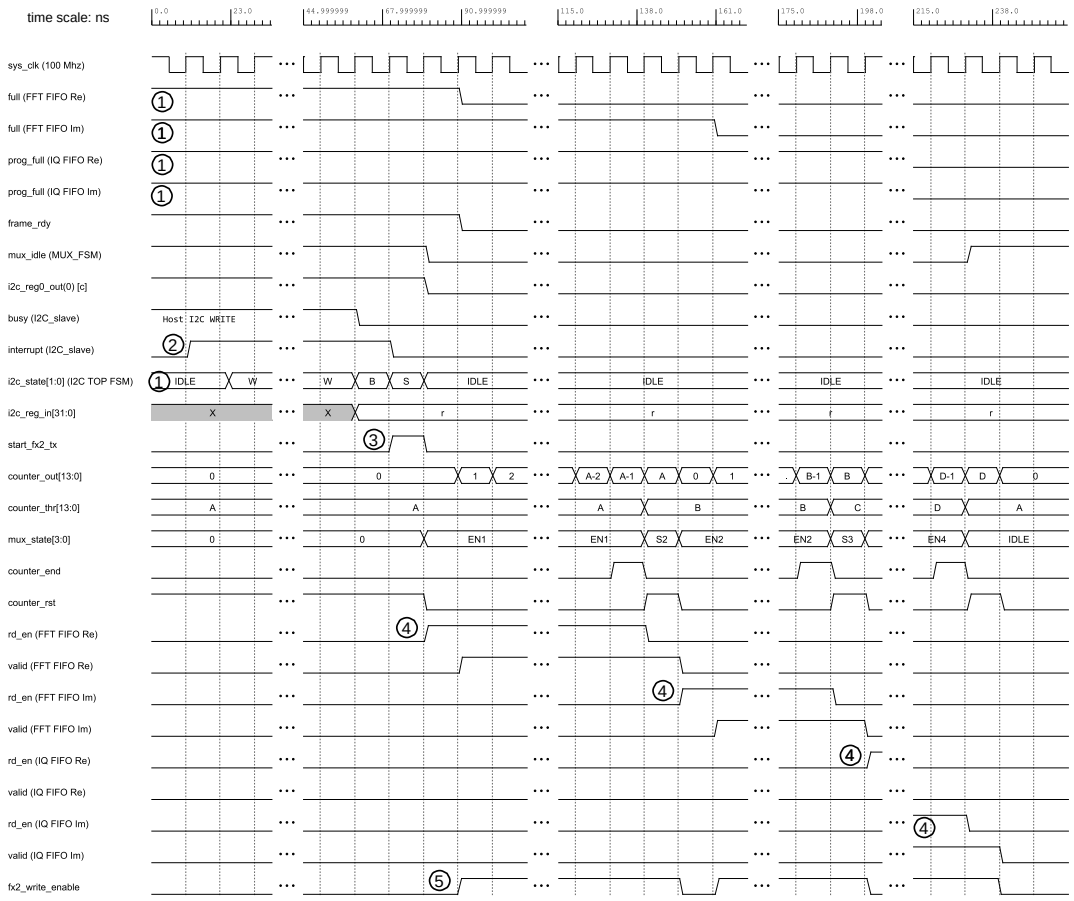


Figura 5.5: Diagrama temporal: multiplexagem dos FIFOs de armazenamento.

5.5 Core USB

Este *core* (aberto) à semelhança do *core I2C* foi adaptado do *reference design* disponibilizado pela *Trenz*. Na sua essência é constituído por dois FIFOs que efetuam a conversão de 32 bits para 8 bits (ou vice-versa) e de 100 MHz para 48 MHz (ou vice-versa). Para além disso gera todos os sinais de interface para o controlador USB com base em duas máquina de estados `FIFO_to_FX2_FSM` e `FX2_to_FIFO_FSM` que funcionam alternadamente para transmissão e recepção de dados respec-

tivamente. O *core* apresenta uma outra máquina de estados responsável pela geração do sinal `PKTEND` caso o número de *bytes* a transmitir não seja múltiplo do tamanho do *slave* FIFO do respectivo *endpoint* IN.

Não sendo objectivo deste trabalho descrever detalhadamente o funcionamento do *core* são apenas descritos os aspectos essenciais do ponto de vista do utilizador, de qualquer das formas é apresentado o diagrama de blocos no anexo B.

- **Transmissão** as entradas `tx_fifo_din` e `tx_fifo_wr_en` são respectivamente o porto de entrada e o sinal de escrita para o FIFO do *core*. A partir do momento em que o Tx FIFO contêm dados, a máquina de estados respectiva efetua o processo de escrita no controlador.
- **Recepção** a máquina de estados de recepção monitoriza a presença de dados nos *slaves* FIFOs do controlador através da *flag* `FLAGD`, se o Rx FIFO não se encontrar cheio efetua a leitura de dados. A recepção tem prioridade sobre a transmissão, sendo a última temporariamente suspensa em caso de conflito.

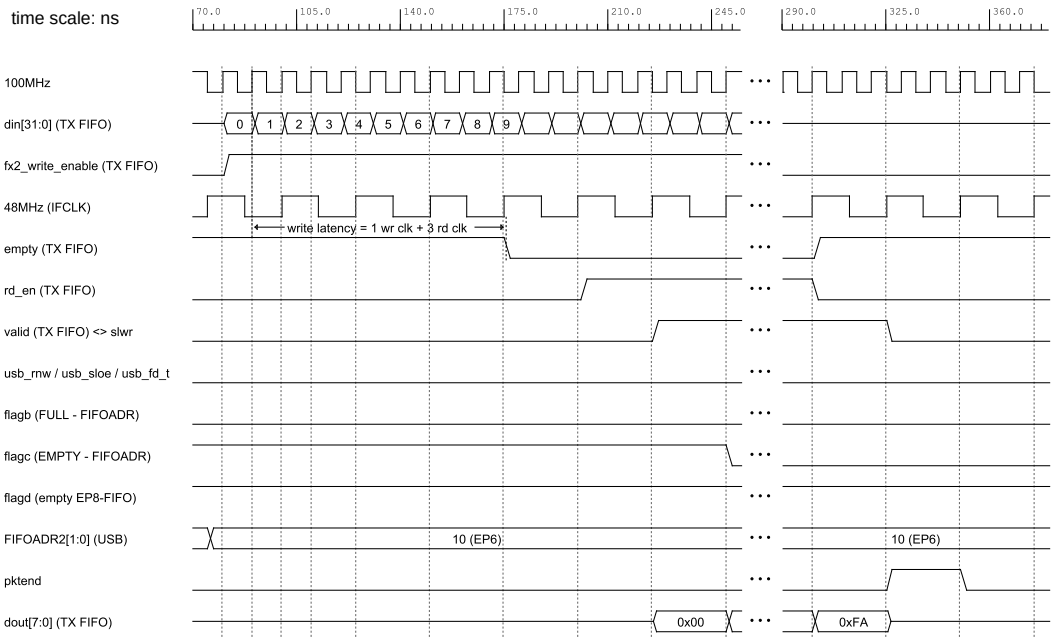


Figura 5.6: Diagrama temporal: Transmissão de dados para os Slave Fifos do controlador USB.

Capítulo 6

Testes e Resultados

6.1 Introdução

Os testes aqui apresentados contemplam o hardware desenvolvido e o software para a FPGA. A PLL foi dimensionada para uma potência de sinal de 10 dBm considerando uma ADC de 14 bits com representação em complemento para dois, ou seja as constantes dos filtros foram calculados para estes valores do sinal de entrada.

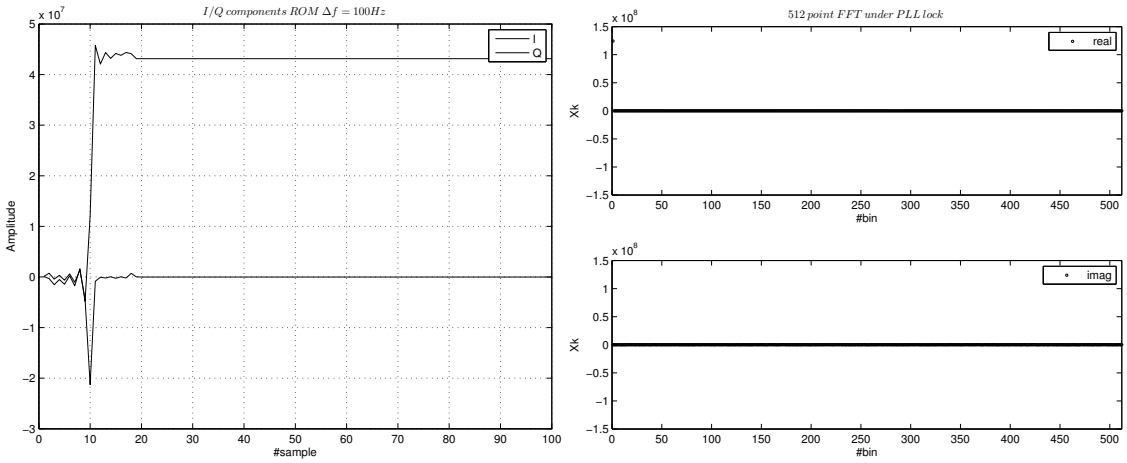
A cadeia de sinal foi inicialmente testada, para evitar frequentemente a montagens dos recursos laboratoriais que eram morosos, com uma simulação do sinal de 10.7 MHz guardada em memória na própria FPGA (sinal sintético). Numa fase posterior foi utilizado um gerador de formas de onda para fornecer o sinal CW ao detetor desenvolvido, os dados das componentes I/Q e os dados da FFT foram analisados para vários valores de potência de entrada ou equivalentemente para vários valores de CNR.

Para facilitar os testes foi desenvolvida uma interface gráfica feita para observar os dados em tempo real, as componentes em fase e quadratura recolhidas no PC anfitrião, as quais foram posteriormente analisadas com o MatLab.

6.2 Resultados usando sinal sintético

A figura 6.1 seguinte mostra os resultados respeitantes à aquisição de sincronismo com o sinal sintético. Adquiridos os dados com uma resolução temporal de 40 ms (taxa das componentes I/Q é de 25 S/s) é possível observar a resposta transitória (deverá ter um amortecimento próximo do crítico como se constata da componente I) a qual decorre sensivelmente em 4 amostras ou seja 100 ms o que é típico nestas circunstâncias de elevada relação CNR. A componente Q atinge um valor praticamente nulo devido ao elevado ganho DC do filtro de malha e portanto, após as primeiras amostras, em que a fase relativa entre o sinal Read Only Memory (ROM) e NCO rola devido às distintas frequências, o NCO rapidamente alcança a frequência do sinal terminando o processo de sincronização.

O segundo método de detecção usando FFT na FPGA é implementado em paralelo (embora não seja necessariamente obrigatório) realizado com 512 bins e consta da figura 6.1. Naturalmente apresenta-se apenas uma risca dada a qualidade espectral do sinal sintético. Dada a resolução espectral da FFT (cerca de 9 Hz) seria necessário haver uma significativa dispersão espectral do sinal para observar conteúdos de potência significativa nas outras riscas.



(a) Processo de aquisicao, I/Q.

(b) Espectro estimado na FPGA.

Figura 6.1: Testes com sinal sintetico armazenado na FPGA.

6.3 Testes com gerador RF

Foi usado de seguida um gerador de sinal de Radio Frequency (RF) para testar o sistema com um sinal CW mais realista contudo ainda sem ruído gaussiano adicionado. O gerador não é contudo a melhor solução para estes testes pois ao mudarmos a amplitude do sinal o gerador suprime-o por momentos. Contudo no presente estado de desenvolvimento o método foi suficiente para avaliarmos a capacidade do sistema sincronizar após aparecimento do sinal, re-adquirir o sincronismo de cada vez que se alterava a amplitude e de testar a linearidade do sistema.

A gama de potências de entrada para o teste foi escolhida de modo a varrer a gama de entrada da ADC, o valor de pico à entrada da ADC é dado por:

$$V_{ADC} = \sqrt{P_S \cdot Z} \tag{6.1}$$

em que Z representa a impedância de entrada do circuito da ADC, neste caso 50 Ω e P_S a potência fornecida pelo gerador. O *span* de entrada da ADC foi configurado para 2 V_{pp} em todos os testes, o que significa que ocorre overflow da ADC sempre que a potência do gerador seja superior a 13 dBm.

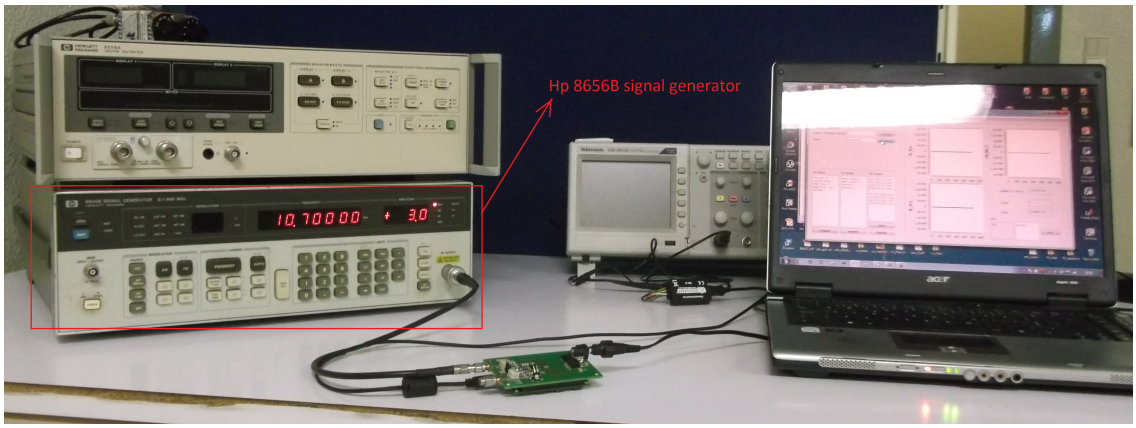


Figura 6.2: Equipamento envolvido nos testes de laboratório.

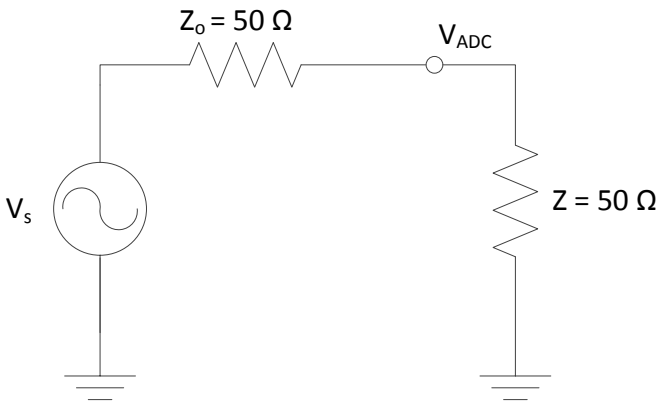


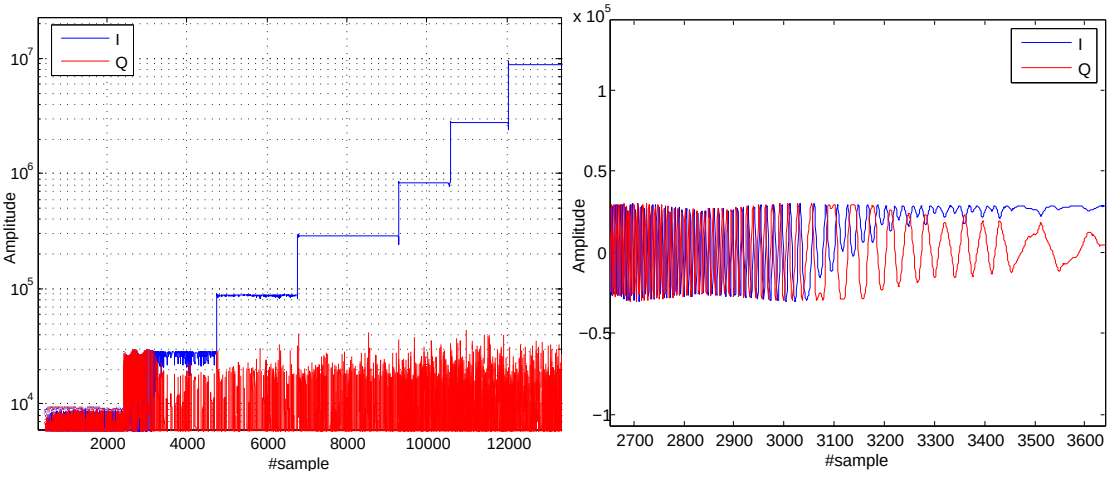
Figura 6.3: Circuito ilustrativo para cálculo da potência entregue à ADC.

A figura 6.4a mostra os resultados do teste introduzindo uma atenuação cada vez menor até alcançar os 3 dB começando em -57 dBm. Repare-se que para 60 dB de atenuação, sem AGC, o ganho do detetor de fase se reduziu de 1000 e a largura de banda da PLL e da constante de amortecimento são 33 vezes menores que o projectado: a malha funciona em condições muito distintas do projectado.

Evidenciam-se os pontos mais relevantes:

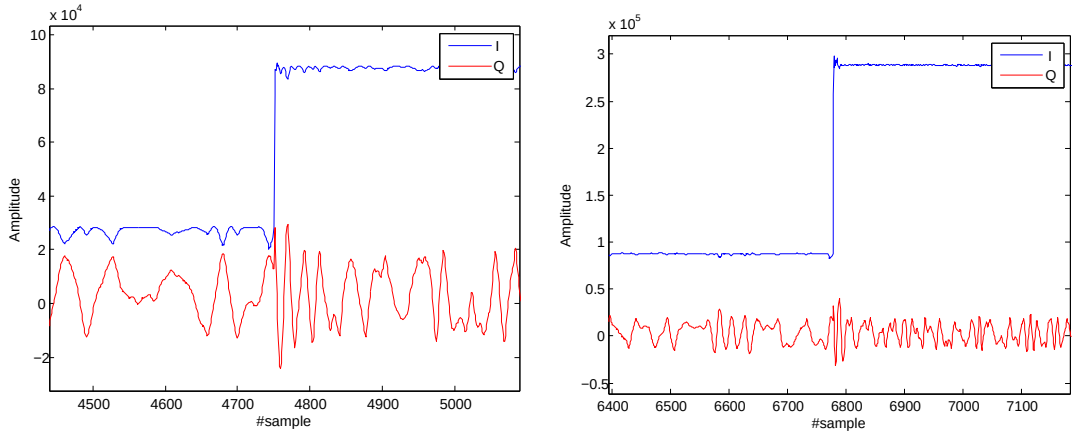
- Primeiro patamar (até cerca da amostra 2400): -57 dBm. O sincronismo não é adquirido pelo menos no tempo até subir a amplitude do sinal. A ampliação mostra duas sinusóides em quadratura indicando que a fase entre o sinal e o NCO está a variar sistematicamente e é difícil, sem uma análise mais detalhada, averiguar se há tendência à aquisição de sincronismo. Esta situação é esperada: o ganho do detetor de fase é muito baixo e a integração da sua saída exige um longo tempo para alcançar o valor suficiente para corrigir a frequência do NCO.
- Segundo patamar (até cerca da amostra 4700): -47 dBm. O sinal adquire sincronismo num processo bastante lento o qual se deve a uma enorme redução da largura de banda do loop em relação ao valor nominal de projecto (3 dBm) e eventualmente ao fundo de ruído térmico do sinal. De facto a frequência natural do loop é ainda inferior a 3 Hz. Os detalhes do processo de aquisição estão na figura 6.4b onde é visível o batimento pré-aquisição. O sincronismo é ainda precário: por vezes a fase parece "fugir" e o loop tende a corrigir: nenhum ciclo é perdido (a componente I não inverte polaridade).
- Terceiro patamar (depois de 4700 e até 6800). A supressão de sinal e o reaparecimento leva ao ajuste da sincronização. O sincronismo parece bastante mais estável contudo o loop reage de forma muito oscilatória num processo que aparenta sub-amortecido (de facto a constante de amortecimento é de cerca de 0.05).
- Quarto patamar e seguintes (depois da amostra 6800). A resposta à mudança de amplitude é ágil, não tem sinais de sub-amortecimento, o sincronismo é robusto e a componente em quadratura tem flutuações relativas muito baixas. No penúltimo patamar a fase tem um pico de apenas 0.7º.

A variância de fase nas componentes I/Q (taxa de 25 S/s) foi avaliada para os patamares de teste nos períodos após o sincronismo estar definitivamente concluído. Os resultados constam da figura 6.5 e naturalmente reflectem o comportamento esperado: uma redução com o aumento da amplitude que tem como óbvia consequência a melhoria da CNR e SNR na malha da PLL.



(a) Componentes I/Q, incrementos de 10 dB.

(b) Detalhe do processo de lock para -47dBm.



(c) Detalhe do processo de lock para -37dBm.

(d) Detalhe do processo de lock para -27dBm.

Figura 6.4: Resultados obtidos com o gerador de RF.

A variância foi calculada através da seguinte expressão em que co_i e co_q representam as amostras das componentes em fase e quadratatura respectivamente.

$$\overline{\theta_{rms}}^2 = VAR\left(\tan^{-1}\left[\frac{co_Q(n)}{co_I(n)}\right] . \frac{180}{\pi}\right) \tag{6.2}$$

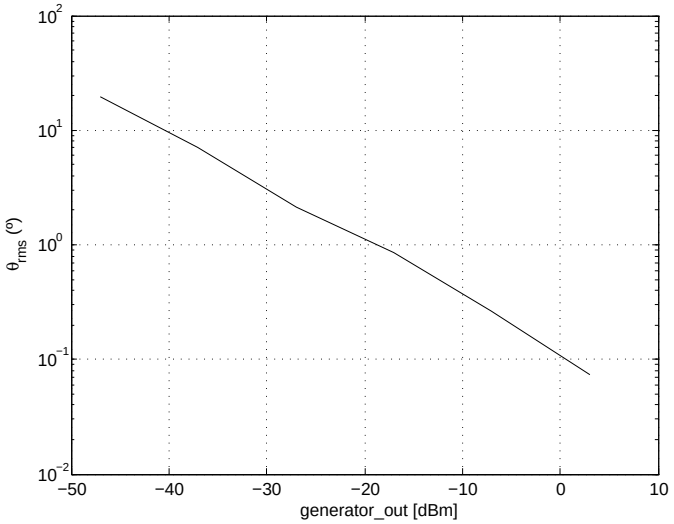


Figura 6.5: Valor quadrático médio da variância de fase à saída do detector síncrono

6.3.1 Análise do período de pré aquisição

O período de menor amplitude do sinal (-57 dBm até pouco mais da amostra 2000) foi igualmente analisado por suscitar alguma curiosidade quanto ao comportamento da malha. Estará a malha nestes cerca de 90 segundos a convergir para sincronismo? A frequência foi então avaliada recorrendo a FFTs complexas de 128 pontos (resolução espectral de cerca de 0.2 Hz) calculadas sobre as componentes I/Q multiplicadas por uma janela de Hamming: uma FFT no início da gama temporal (First), outra a meio (Middle) e outra quase no fim (End) do período. Os resultados apresentam-se na figura 6.6.

É óbvio que a frequência (negativa) das componentes I/Q está a evoluir de forma a conseguir o sincronismo. Os indícios são dois:

- O pico do espectro aproxima-se dos 25 Hz (ou 0 Hz);
- O sinal está a aumentar de amplitude ou seja a frequência evolui para DC e deve estar no sentido ascendente do flanco da resposta do filtro passa-baixo.

Portanto a malha parece estar, embora lentamente, a tentar adquirir o sincronismo. Interessante a existência de um offset (DC) que é no entanto cerca de 32 dB inferior à amplitude média do sinal e portanto o erro de fase após sincronismo é muito reduzido (1.43°). A existência deste offset tem duas desvantagens:

- No processo de aquisição o valor médio (DC) do batimento tem que cancelar e sobrepor-se ao offset para conseguir o sincronismo. Neste caso, devido às constantes não óptimas da malha, a aquisição está ainda mais dificultada.
- Na ausência de sinal a integração deste offset conduz mais rapidamente ao desvio da frequência dos oscilador pois é integrado.

É possível também verificar que a potência do sinal está praticamente toda contida numa largura de banda de 5 Hz.

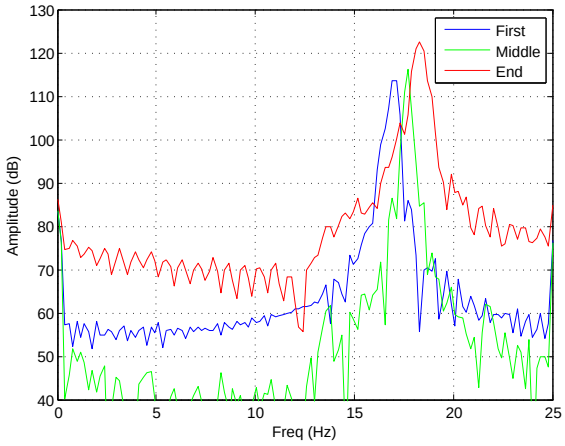


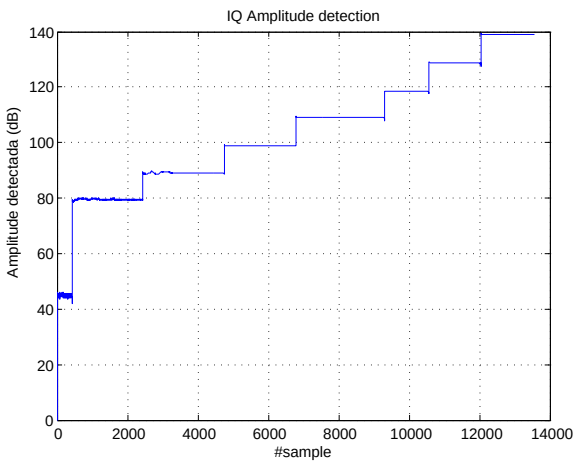
Figura 6.6: Evolução do espectro na aquisição de sincronismo.

6.3.2 Deteção por componentes I/Q

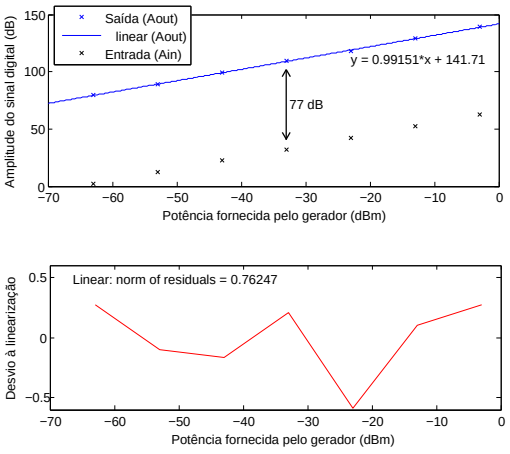
Uma estimativa da amplitude do sinal de entrada utilizando as componentes I/Q à saída pode ser obtida através de:

$$A_{0\_detect} = 20\log_{10}\left(\sqrt{{co_i}^2 + {co_q}^2}\right) \tag{6.3}$$

O resultado obtido com o gerador RF para a gama de -57 dBm até 3 dBm encontra-se na figura 6.7a. A figura 6.7b compara a amplitude do sinal à entrada do detetor (saída da ADC) e à saída (calculado através de 6.3). O ganho de processamento é de 77 dB o que está muito próximo do valor calculado para o loop (80 dB). A linearidade do método é bastante satisfatória (m=0.9915) tendo em conta que depende de factores como a própria linearidade da ADC e da linearidade do gerador de sinais, contudo convêm relembrar que o teste foi efetuado com um sinal espectralmente muito limpo e que apresenta pouco ruído de fase em comparação a uma situação real de deteção.



(a) Amplitude detectada [-57 dBm → 3 dBm]



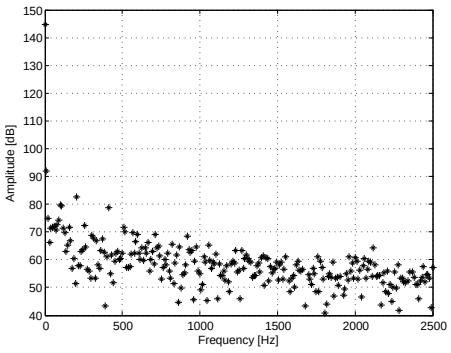
(b) Linearidade da detecção por componentes I/Q.

6.3.3 Deteção por FFT

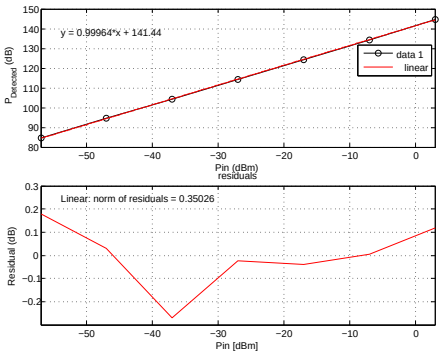
Um conjunto de testes, mas agora com uma resolução de 3 dB, foi efectuado usando com o detetor o método das FFTs. O espectro calculado com 512 pontos sobre o sinal sincronizado e com amplitude de 3 dBm está representado na figura seguinte.

Levando em conta que a resolução de frequência é próxima de 10 Hz o espectro confirma que quase toda a potência está contida em DC: o segundo bin já tem uma amplitude pelo menos 50 dB inferior. Existe contudo à volta de DC uma densidade espectral de ruído superior àquela existente acima de 1 kHz. Levando em conta a resolução da FFT a CNR será da ordem de 95 dB para um sinal de 3 dBm que denota pelo menos 10 dB de ganho de processamento. Aliás isto confirma os resultados obtidos nos espectros anteriores (figura 6.6) onde o sinal, com -57 dBm (60 dB inferior) é perfeitamente identificado mesmo que adicionemos $10\log_{10}(10/0.02) = 27\ dB$ ao patamar de ruído exibido nestas figuras.

A linearidade é analisada na figura seguinte em que se tomou para amplitude do sinal apenas a risca mais potente. O desvio da linearidade é muito reduzido não alcançando os 0.3 dB nos pontos de observação.



(a) Espectro do sinal a 5 kS/s. FFT de 512 pontos (sem janela), resolução 10 Hz.



(b) Linearidade do método de deteção por FFT.

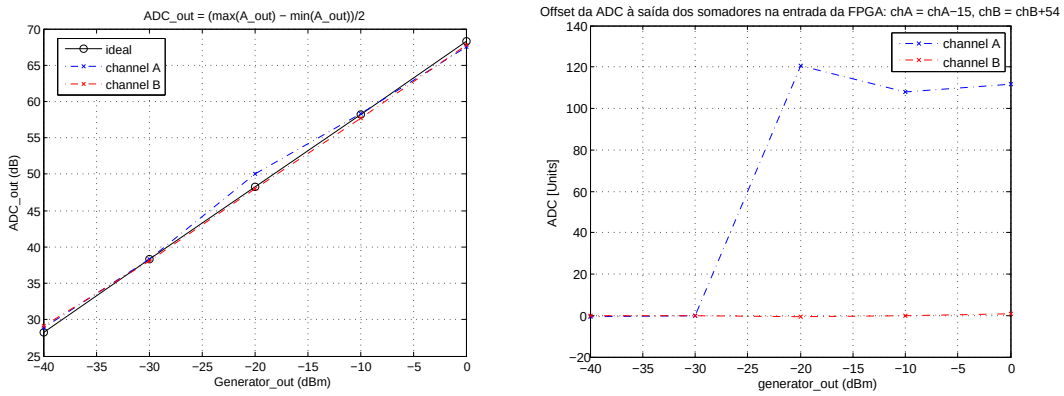
6.3.4 ADC

Com os dados acima explorou-se a linearidade dos dois canais: o gerador de sinal foi passado por um divisor de 3 dB e metido em cada um dos divisores. Os resultados foram avaliados a partir das componentes cartesianas e constam da figura abaixo.

Enquanto um dos canais é bastante linear o outro parece exibir um desvio significativo que se revelou inesperado.

O software foi escrutinado com detalhe bem como a alocação dos pinos da ADC sem ser descoberta nenhuma razão aparente para justificar o desvio. Optou-se então pela recolha das samples da ADC e constatou-se que a partir de um certo nível de sinal de entrada, que deverá estar algures por volta dos -20 dBm aparece um significativo offset no canal A que não aparece para amplitudes muito baixas. O facto é visível na a figura 6.9.

O problema parece ser de Hardware. Até ao momento não foi possível rever o hardware mas não é de excluir a possibilidade de um mau contacto num dos bits da ADC que faz a interface com a FPGA.



(a) Linearidade de amplitude

(b) Offset

Figura 6.9: Teste efetuados à saída dos dois canais do AD9248.

6.4 Sumário

O detetor implementado adquire rapidamente o sincronismo para CNRs iguais ou superior a 55 dB (ordem dos 100 mS), para uma CNR de 45 dB o sincronismo já demora bastante tempo a adquirir demorando cerca de 40 segundos. Para valores inferiores o sincronismo ainda se torna mais lento, muito embora se tenha observado pela análise espectral das componentes I/Q que a frequência do NCO se aproxima lentamente da frequência de entrada. Estes valores poderiam ser melhorados se para o efeito se tivesse dimensionado a PLL para o valor mínimo de amplitude de entrada ao invés do máximo, ou alternativamente com a inclusão de um AGC.

Recordando que numa situação real o valor de CNR esperado é de cerca de 56 dB isto abre boas perspectivas à utilização do detetor desenvolvido, contudo não é invulgar que o sinal de satélite sofra atenuações da ordem dos 10-20 dB em alguns períodos do dia pelo que é importante 'esticar' ao máximo a gama dinâmica da PLL.

Capítulo 7

Conclusões e trabalho futuro

7.1 Conclusões

Existem placas de desenvolvimento que poderiam ser usadas para atingir os objectivos aqui descritos contudo, como os custos do hardware e software de desenvolvimento é proibitivo neste tipo de aplicações, os resultados conseguidos foram bastante interessantes.

Foi desenvolvido o embrião para um detetor digital baseado em FPGA de dois canais que se potencia completamente versátil tanto no respeitante ao método de detecção do sinal como à frequência de entrada que poderá tomar um valor pelo menos até metade da frequência de *sampling* das ADCs utilizadas e eventualmente até mais elevada com a perda de alguma gama dinâmica.

O sistema foi baseado num simples kit FPGA (custo de apenas 200 Euros) e uma placa desenvolvida para o efeito contendo ADCs. O preço desta última é essencialmente o preço da realização do PCB.

Os testes realizados mostram que globalmente os objectivos foram alcançados tendo-se implementado um sistema de dois canais e dois métodos de detecção do sinal: o método síncrono (usando PLL) e o método de realização de FFT complexas realizadas sobre sinais reais. A utilização da FFT complexa foi premeditada para a posterior implementação de uma FLL. A ocupação da FPGA está a cerca de 60% o que poderá ser crítico se quisermos realizar um sistema com todos os métodos de detecção.

Chama-se a particular atenção para o método da PLL em que os dados adquiridos mostram precisamente o comportamento esperado pelas simulações realizadas com o modelo *Simulink* especialmente no processo de aquisição. Este desenvolvimento foi interessante pois mistura neste trabalho um assunto relativamente complexo que é o funcionamento da PLL, se calhar mais simples de interpretar quando observado no mundo analógico, com o mundo digital e muito facilitou o entendimento do sistema implementando em FPGA.

7.2 Trabalho futuro

A ideia global do projecto seria abrir caminho para o desenvolvimento de um detetor quase universal e que teria integrada na mesma placa a FPGA e a placa de ADCs constituindo um sistema autónomo e se possível auto-reconfigurável.

Mostrado o conceito existem ainda alguns pontos que necessitam de mais alguns desenvolvimentos, nomeadamente:

- A implementação do AGC. Como vimos das equações da PLL a redução da amplitude do sinal provoca uma redução tanto da constante de amortecimento como da largura de banda da PLL. Ambos contribuem a PLL a perder o sincronismo com maior facilidade devido a qualquer desvio de frequência breve (falta de sinal por atenuação por exemplo) ou perturbação devido a ruído térmico. A proposta em estudo consiste na realização de um AGC coerente. A potência do sinal seria estimada pelas componentes cartesianas devidamente filtradas passa-baixo que dividiriam o sinal da componente que alimenta o filtro de loop. Este pode ser um ponto interessante que deveria ser simulado no modelo Simulink.

- Realizar uma estimativa da CNR. Esta estimativa é importante para estabelecer a qualidade do sincronismo da PLL e evitar a perda irreversível deste. A ideia que proponho é a realização de um filtro passa-alto no último estágio para rejeitar a potência de sinal. A potência do sinal seria então sujeita a uma média de longa duração (alguns segundos) e a potência do sinal, calculada no ponto anterior, usada para obter a relação.

- Implementar um mecanismo que, mediante a indicação de uma CNR crítica, permita congelar o cálculo do filtro de loop para evitar que a eventual perda total de sinal possa conduzir a um afastamento demasiado elevado do frequência do NCO e portanto a reaquisição plena seja mais imediata logo que o CNR recupere.

Uma ideia interessante para o futuro seria um método FLL: frequency locked loop ou mais correntemente Automatic Frequency Control (AFC). Este método de tracking é visto como mais robusto pois depende da variância de Allan da frequência do sinal [Roh12] a qual é tipicamente 10^{-12} para as fontes de sinal nesta aplicação. Isto permite a realização de malhas com largura de banda bastante inferior às PLLs que seguem a fase. O trabalho referente ao AGC e CNR seria partilhado nesta implementação. O discriminador de frequência (que substitui o detetor de fase) deveria ser precedido de um filtro passa-baixo com cerca de 50 Hz de largura de banda e a polaridade do ganho total da malha levada em consideração pois ao contrário da PLL só há um ponto com realimentação negativa nos detetores de frequência.

Um ponto interessante seria ter implementado todos os detetores e avaliar o desempenho comparativo de cada um. De facto nesta FPGA ou outra com maiores capacidades poderemos ter vários NCOs independentes e eventualmente tirar sinergias de cada um dos sistemas estendendo a gama dinâmica dos receptores além do que é possível com sistemas puramente analógicos.

Apêndices

Apêndice A

Placa de conversão analógico-digital

O esquemático da placa de ADC's realizado em OrCad e o respectivo diagrama de blocos encontram-se nas figuras A.2 e A.1 respectivamente.

A placa é alimentada com 5 Volts a partir de um socket DC, os 5 V alimentam o chipset da FPGA através do conector `JM5` e ao mesmo tempo o `LM1086` que fornece os 3.3 V para o chip da ADC `(AD9248)`. O sinal RF é convertido para um sinal diferencial através dos baluns T1-1T, as saídas dos canais de conversão são ligados a pinos gerais de Input/Output (I/O) da FPGA assim como os sinais de OTR que indicam overflow da saída da ADC. A configuração hardwired da ADC encontra-se na tabela A.3, nas tabelas A.1 e A.2 encontram-se as configurações possíveis com a mudança dos jumpers J1 e J2.

Status	**Mode**	**VREF(V)**	**Span (Vpp)**
Sense = Vref (J1)	Internal Fixed Reference	0.5	1.0
Sense = Gnd (J1)	Internal Fixed Reference	1.0	2.0

Tabela A.1: ADC configuração (J1).

Status	**Data format**
DFS High (J2)	Twos Complement
DFS Low (J2)	Offset Binary

Tabela A.2: Configuração ADC (J2).

Pin	Description	Mode
DCS	Duty cycle Stabilizer	Enabled (High)
MUX_SELECT	Multiplexed data mode	Multiplex Disabled (High)
SHARED_REF	Shared Reference	Enabled (High)
PDWN_A	Power Down CH-A	CH-A Enabled (Low)
PDWN_B	Power Down CH-B	CH-B Enabled (Low)

Tabela A.3: Configuração ADC (Hardwired).

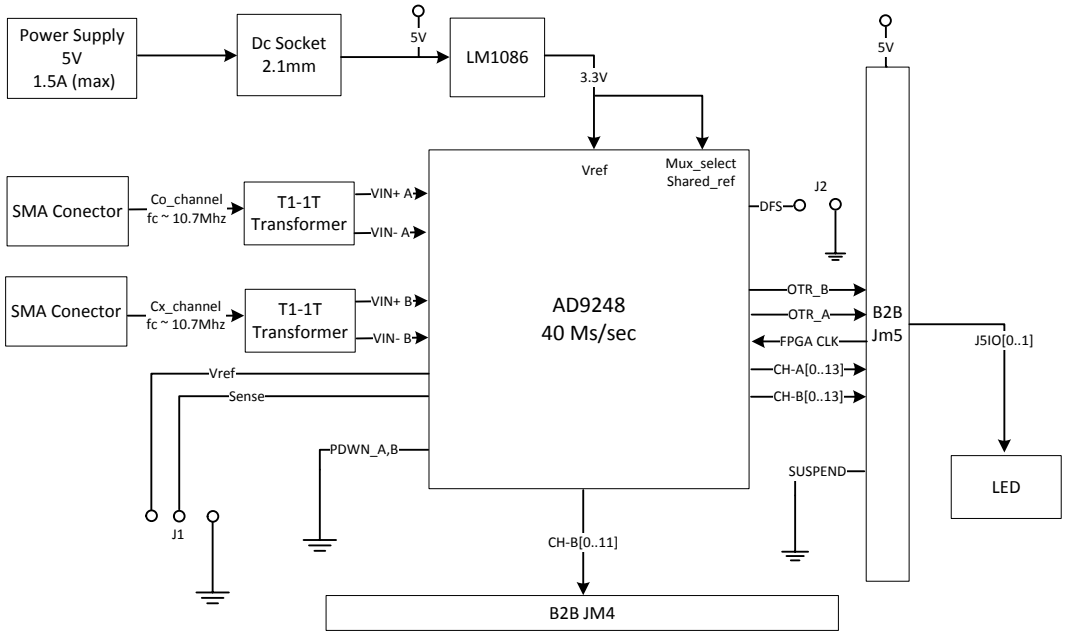


Figura A.1: Diagrama de blocos da placa de conversão analógica.

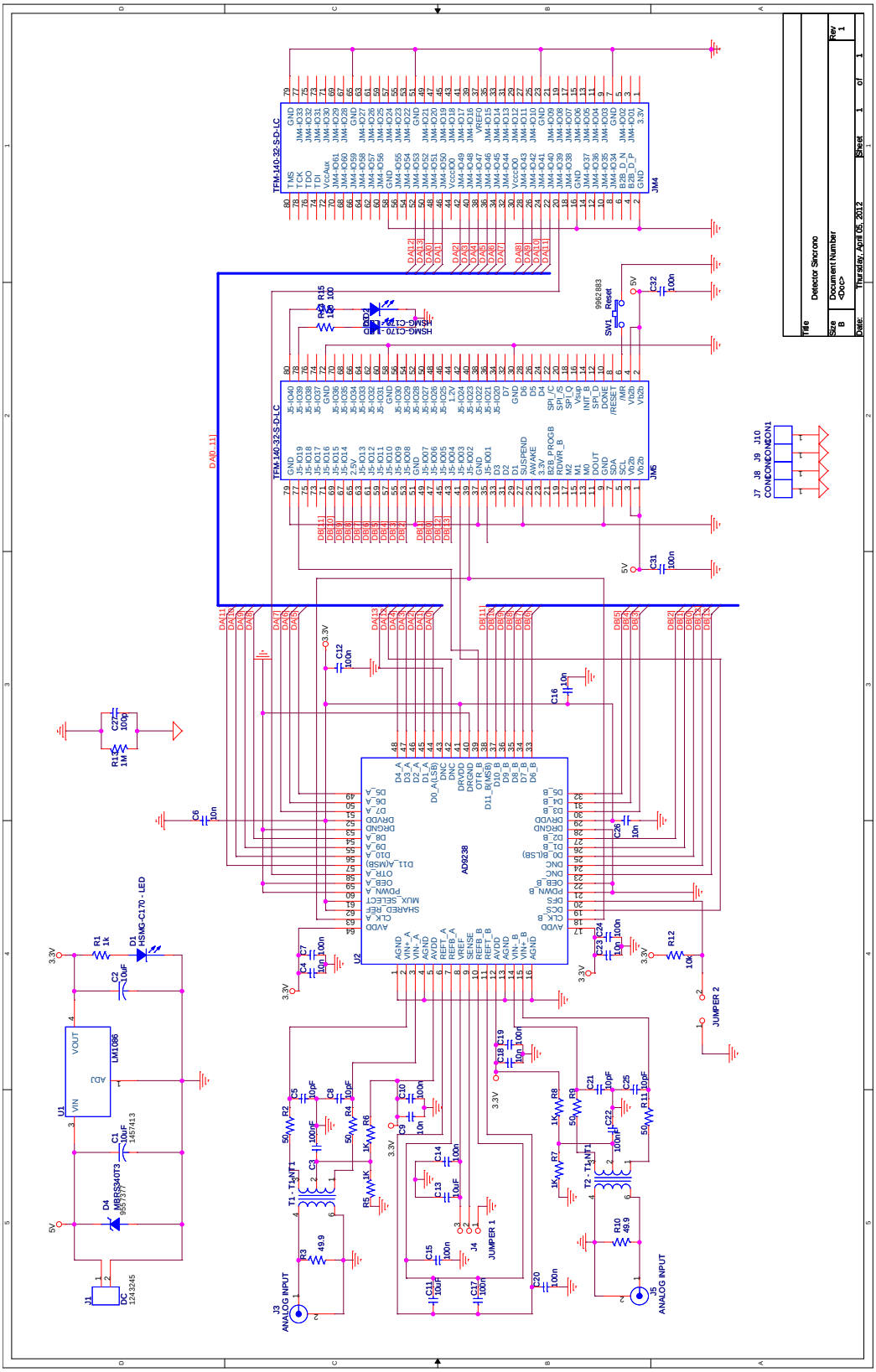


Figura A.2: Esquemático: ADC Printed Circuit Board.

Apêndice B

Sistema sintetizado em FPGA

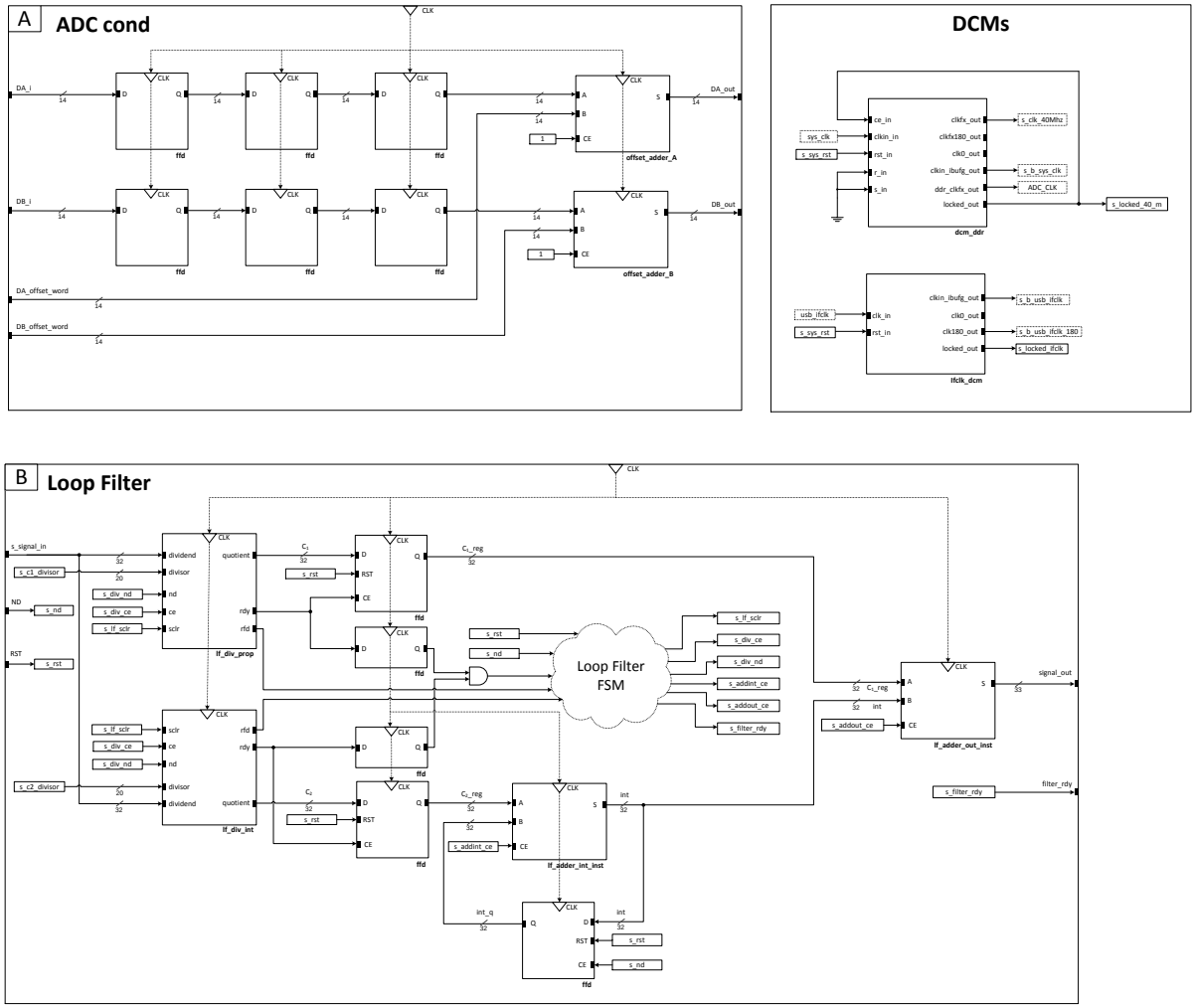


Figura B.1: FPGA: Bloco de entrada do sinal da ADC (`ADC_cond`), DCMs e filtro de malha.

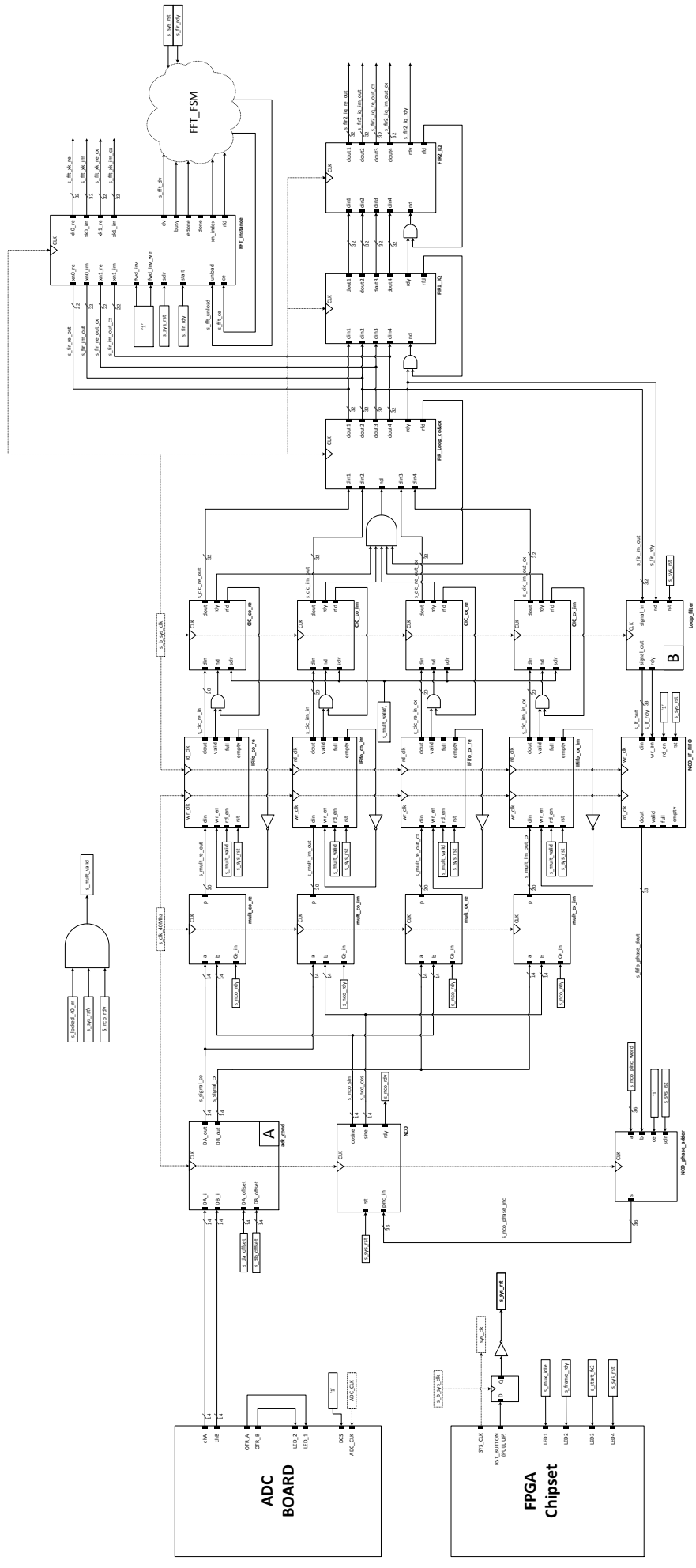


Figura B.2: FPGA: Interface com a placa de ADCs e o chipset, PLL digital, bloco FFT, filtragem das componentes I/Q

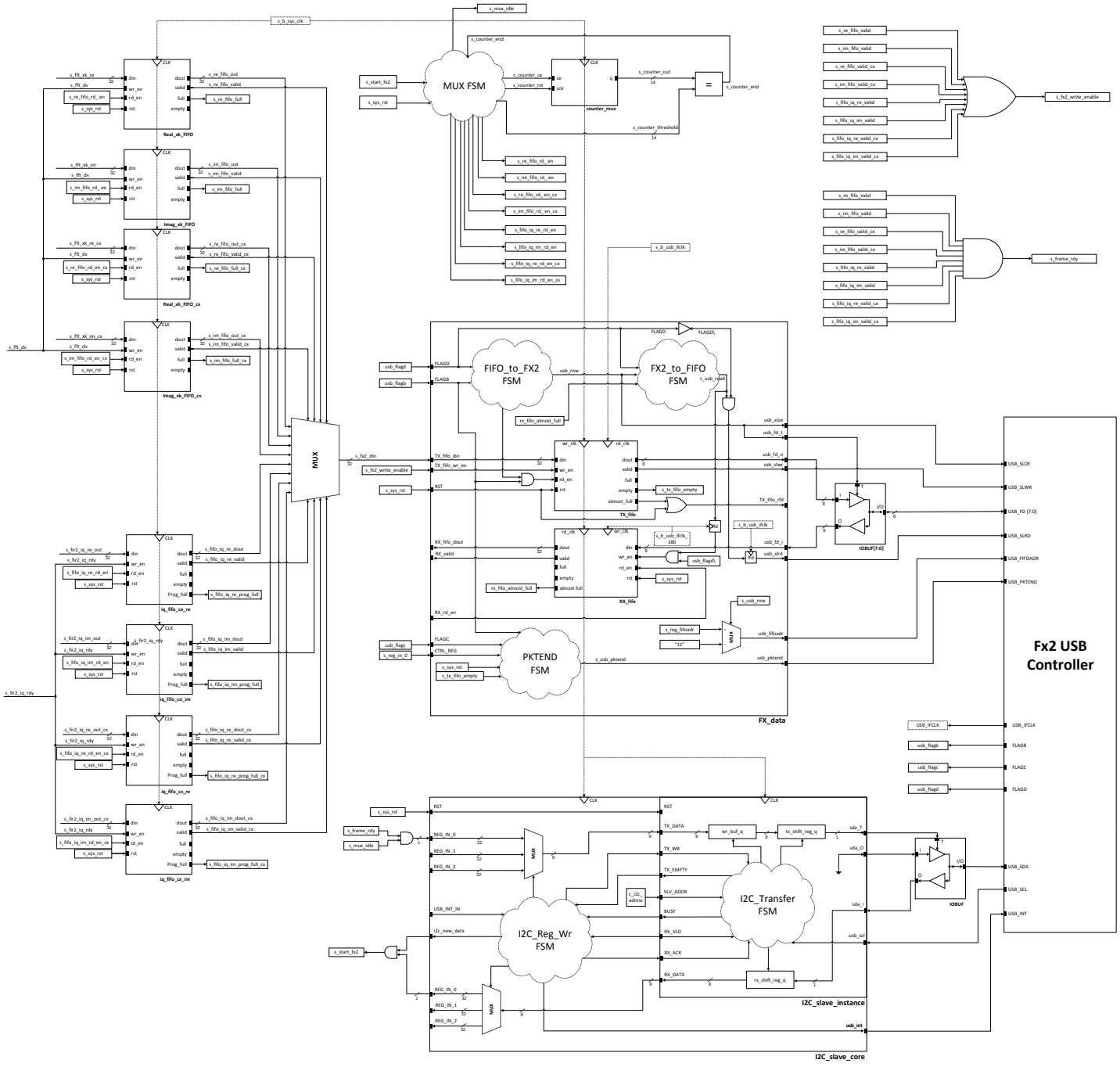


Figura B.3: FPGA: Fifos de armazenamento, Fx2 Core, I2C core, respectivas máquinas de estados e interface com o controlador USB.

B.1 Ocupação de recursos

	Slices	Slice Reg (FFs)	LUTs	LUTRAM	BRAM	DSP48A	BUFG	DCM
ICON	56	28	72	0	0	0	1	0
ILA	311	324	293	113	31	0	0	0
Chipscope	367	352	365	113	31	0	1	0
offset_adder_a	14	14	14	0	0	0	0	0
offset_adder_b	14	14	14	0	0	0	0	0
logic/fsm's	42	84	0	0	0	0	0	0
ADC Cond	70	112	28	0	0	0	0	0
mult_co_re	0	0	0	0	0	1	0	0
mult_co_im	0	0	0	0	0	1	0	0
mult_cx_re	0	0	0	0	0	1	0	0
mult_cx_im	0	0	0	0	0	1	0	0
PD multipliers	0	0	0	0	0	4	0	0
NCO	87	141	90	2	1	2	0	0
cic_co_i	241	323	156	54	0	3	0	0
cic_co_q	243	323	156	54	0	3	0	0
cic_cx_i	243	323	156	54	0	3	0	0
cic_cx_q	241	322	156	54	0	3	0	0
CIC filters	968	1291	624	216	0	12	0	0
FIR (primary)	275	462	457	266	9	12	0	0
FFT	914	1314	880	436	3	10	0	0
fir1_iq	751	868	869	529	13	12	0	0
fir2_iq	278	470	471	267	9	12	0	0
FIR (secondary)	1029	1338	1340	796	22	24	0	0
divider_int	714	1026	715	69	1	10	0	0
divider_prop	721	1034	735	69	1	10	0	0
adder_int	32	32	32	0	0	0	0	0
adder_out	34	33	33	0	0	0	0	0
logic/fsm's	56	100	11	0	0	0	0	0
Loop Filter	1557	2225	1526	138	2	20	0	0
tx_fifo	126	178	113	0	4	0	0	0
rx_fifo	39	45	34	0	0	0	0	0
logic/fsm's	25	25	38	0	0	0	0	0
Fx2 Data	190	248	185	0	4	0	0	0
I2C	209	97	352	1	0	0	0	0
fifo_fft_co_re	0	0	0	0	0	0	0	0
fifo_fft_co_im	0	0	0	0	0	0	0	0
fifo_fft_cx_re	1321	72	1592	1024	0	0	0	0
fifo_fft_cx_im	1321	72	1592	1024	0	0	0	0
fifo_co_i	369	66	455	256	0	0	0	0
fifo_co_q	369	66	455	256	0	0	0	0
fifo_cx_i	369	66	455	256	0	0	0	0
fifo_cx_q	369	66	455	256	0	0	0	0
FIFOS (storage)	4118	408	5004	3072	0	0	0	0
nco_if	66	103	35	0	1	0	0	0
fifo_if_co_re	51	71	37	0	1	0	0	0
fifo_if_co_im	51	71	37	0	1	0	0	0
fifo_if_cx_re	52	71	37	0	1	0	0	0
fifo_if_cx_im	51	71	37	0	1	0	0	0
FIFOS (interface)	271	387	183	0	5	0	0	0
counter	14	14	14	0	0	0	0	0
nco_adder	36	36	36	0	0	0	0	0
toplevel	120	15	215	0	0	0	2	0
Toplevel Logic/FSMs	170	65	265	0	0	0	2	0
IFCLK DCM	0	0	0	0	0	0	2	1
SYSCLK DCM	0	0	0	0	0	0	3	1
Used	10225	8440	11299	5040	77	84	8	2
Available	23872	47744	47744		126	126	24	8
	Slices	Slice Reg (FFs)	LUTs	LUTRAM	BRAM	DSP48A	BUFG	DCM
% Utilização	42.9	17.7	23.7		61.2	66.7	33.4	25

Figura B.4: Distribuição dos recursos da FPGA.

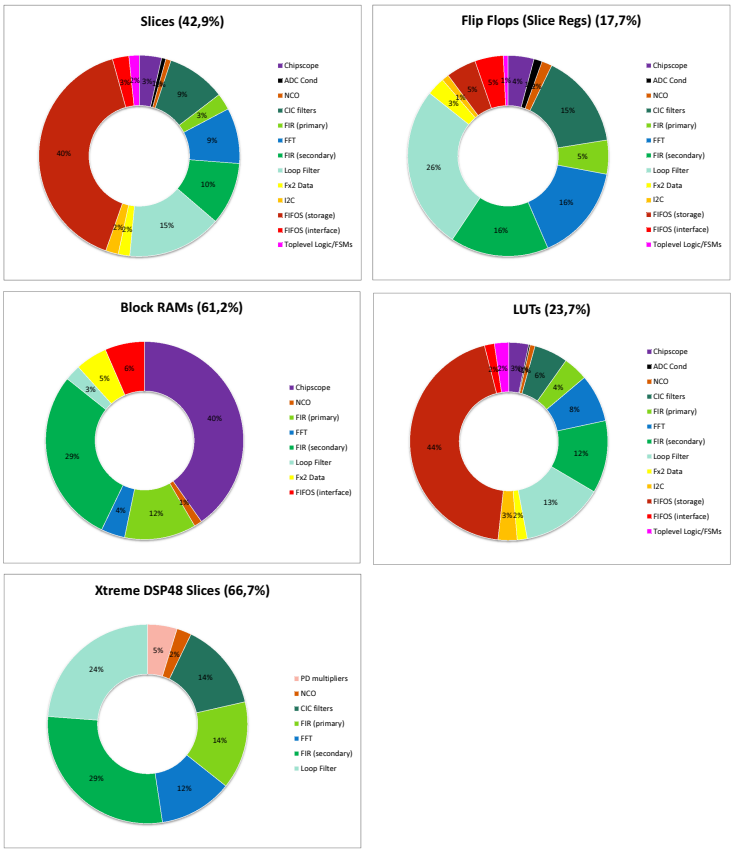


Figura B.5: Distribuição dos recursos da FPGA.

B.2 Xilinx IP Cores: Configuração

Component Name:	`dcm_ddr_40MHz`
Instance(s):	`sysclk_dcm`
DDR flip-flop selection:	`OFFDRRSE`
Select C0:	`CLKFX`
Select C1:	`CLKFX180`
Input Optional Ports:	`RST`
Output Optional Ports:	`LOCKED`
Input Clock Frequency:	`100 MHz`
PhaseShift:	`NONE`
Feedback:	`Internal 1x`
Use Duty Cycle Correction:	✓
Buffer Settings:	`Use Global buffers for all clock outputs`
Output Frequency:	`40 MHz (M=2 D=5)`

Tabela B.1: Xilinx Clocking Wizard Clock - Forwarding/Board Deskew.

Component Name:	ifclk_dcm
Instance(s):	ifclk_dcm_instance
Input Optional Ports:	RST
Ouput Optional Ports:	CLK180, LOCKED
Input Clock Frequency:	48 MHz
PhaseShift:	NONE
CLKIN **Source**:	External → Single
Feedback:	Internal 1x
Use Duty Cycle Correction:	✓
Buffer Settings:	Use Global buffers for all clock outputs

Tabela B.2: Xilinx Clocking Wizard - Single DCM_SP

Component Name:	nco
Instance(s):	nco_instance
Configuration Options:	Phase Generator and SIN COS LUT
System Clock:	40 MHz
Number of Channels:	1
Parameter Selection:	Hardware Parameters
Phase Width:	36
Output Width:	14
Noise Shaping :	Phase Dithering
Phase Increment Programmability :	Streaming
Phase Offset Programmability :	None
Output Selection :	Sine and Cosine
Amplitude Mode :	Unit Circle
Memory Type :	Auto
Optimization Goal:	Auto
DSP48 Use:	Maximal
Latency Options:	Auto
Optional Pins:	Synchronous Clear, RDY

Tabela B.3: Xilinx Core Generator - DDS Generator (NCO)

Core:	LogiCORE IP Multiplier v11.2
Component Name:	mult
Instance(s):	mult_re_co, mult_im_co, mult_re_cx, mult_im_cx
Multiplier Type:	Parallel Multiplier
Port A:	14 bit width - Signed
Port B:	14 bit width - Signed
Multiplier Construction:	Use Mults
Optimization Options:	Speed Optimized
Output Product Range:	Custom Width
Output MSB:	27
Output LSB:	8
Pipeline Stages:	1
Clock Enable:	✓

Tabela B.4: Xilinx Core Generator - PD Multiplier

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	if_fifo
Instance(s):	IFifo_co_re, IFifo_co_im, IFifo_cx_re, IFifo_cx_im
Read/Write Clocks:	Independent Clocks - Block Ram
Read mode:	Standard FIFO
Write Width:	32
Write Depth:	512
Read Width:	32
Read Depth:	512
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Asynchronous
Full Flags Reset Value:	1
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	No Threshold
Programmable Empty Type:	No Threshold

Tabela B.5: Xilinx Core Generator - Interface FIFOs

Core:	LogiCORE IP CIC compiler v2.0
Component Name:	cic_filter
Instances Name:	cic_co_re, cic_co_im, cic_cx_re, cic_cx_im
Filter Type:	Decimation
Number of Stages (N):	3
Differential Delay (M):	1
Number of Channels:	1
Rate Change:	Fixed
Rate (R):	1024
HW specification:	Frequency
Input Frequency:	40
Clock Frequency:	100
Input Data Width:	20
Quantization:	Truncation
Output Data Width:	32
Use Xtreme DSP Slices:	✓
Use Streaming Interface:	✓
ND pin:	✓
SCLR pin:	✓

Tabela B.6: Xilinx Core Generator - CIC Filter

Core:	LogiCORE IP FIR Compiler v5.0
Component Name:	fir_filter
Instances Name:	fir_instance
Coefficient Source:	COE File
Number of Coeff Sets:	1
Coefficients per set:	96
Filter Type:	Decimation
Rate Change Type:	Integer
Decimation Rate Value:	8
Number of channels:	1
HW specificatiom:	Frequency
Input Sampling Frequency:	0.0390625 MHz
Clock Frequency:	100 MHz
Filter Architecture:	Systolic Multiply Accumulate
Coefficient Structure:	Inferred
Coefficient Type:	Signed
Coefficient Width:	16
Number of Paths:	4
Input Data Width:	32
Input Fractional Bits:	0
Output Rounding Mode:	Truncate LSBs
Output Width:	32
Registered Output:	✓
Optimization Goal:	Area
ND pin:	✓
Memory Options:	Automatic

Tabela B.7: Xilinx Core Generator - Loop FIR

Core:	LogiCORE IP FFT v7.1
Component Name:	fft
Instance Name:	fft_instance
Channels:	1
Transform Length:	512
Clock Frequency:	100 MHz
Implementation:	Radix-2 Lite, Burst I/O
Data Format:	Fixed Point
Input Data Width:	22
Phase Factor Width:	24
Scaling Options:	Unscaled
Rounding Modes:	Convergent Rounding
Optional Pins:	CE, SCLR
Output Ordering:	Natural Order
Input Data timing:	No offset
Mem. Options (Data):	Block RAM
Mem. Options (Phase):	Block RAM
Complex Multipliers:	Use 4-multiplier structure
Butterfly Arithmetic:	Use XtremeDSP Slices

Tabela B.8: Xilinx Core Generator - FFT

Core:	LogiCORE Binary Counter v11.0
Component Name:	counter_mux
Instance(s):	counter
Implement using:	Fabric
Output Width:	14
Increment Value:	1
Count Mode:	Up
Clock Enable (CE):	✓
Synchronous Clear (SCLR):	✓
SCLR/CE priority:	SCLR overrides CE
Latency Configuration:	Manual: 1
Feedback Latency Conf.:	Manual: 0

Tabela B.9: Xilinx Core Generator - Binary Counter

Core:	LogiCORE IP Divider generator v3.0
Component Name:	lfdivider
Instance(s):	lf_div_prop, lf_div_int
Algorithm Type:	High Radix
Dividend and Quotient Width:	32
Divisor Width:	20
Fractional Width:	0
Latency Configuration:	Automatic
CE:	✓
SCLR:	✓
SCLR/CE priority:	SCLR overrides CE

Tabela B.10: Xilinx Core Generator - Loop Filter Dividers

Core:	LogiCORE IP Adder/Subtracter v11.0
Component Name:	lfadder_int
Instance(s) Name:	lfadder_int_inst
Implement Using:	Fabric
A Input Type:	Signed
B Input Type:	Signed
A Input Width:	32
B Input Width:	32
Add Mode:	Add
Output Width:	32
Latency Conf.:	Manual: 1
CE:	✓
SCLR:	✓
SCLR/CE priority:	SCLR overrides CE

Tabela B.11: Xilinx Core Generator - Loop Filter Adder Int

Core:	LogiCORE IP Adder/Subtracter v11.0
Component Name:	lfadder_out
Instance(s) Name:	lfadder_out_inst
Implement Using:	Fabric
A Input Type:	Signed
B Input Type:	Signed
A Input Width:	32
B Input Width:	32
Add Mode:	Add
Output Width:	33
Latency Conf.:	Manual: 1
CE:	✓
SCLR:	✓
SCLR/CE priority:	SCLR overrides CE

Tabela B.12: Xilinx Core Generator - Loop Filter Adder Out

Core:	LogiCORE IP Adder/Subtracter v11.0
Component Name:	nco_phase_adder
Instance(s) Name:	nco_adder
Implement Using:	Fabric
A Input Type:	Unsigned
B Input Type:	Signed
A Input Width:	36
B Input Width:	33
Add Mode:	Add
Output Width:	36
Latency Conf.:	Manual: 1
CE:	✓
SCLR:	✓
SCLR/CE priority:	SCLR overrides CE

Tabela B.13: Xilinx Core Generator - NCO phase Adder

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	nco_phase_fifo
Instance(s):	nco_fifo
Read/Write Clocks:	Independent Clocks - Block Ram
Read mode:	First Word Fall Through
Write Width:	33
Write Depth:	16
Read Width:	33
Read Depth:	16
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Asynchronous
Full Flags Reset Value:	1
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	No Threshold
Programmable Empty Type:	No Threshold

Tabela B.14: Xilinx Core Generator - NCO FIFO

Core:	LogiCORE IP FIR Compiler v5.0
Component Name:	fir_iq_1
Instances Name:	fir1_iq
Coefficient Source:	COE File
Number of Coeff Sets:	1
Coefficients per set:	86
Filter Type:	Decimation
Rate Change Type:	Integer
Decimation Rate Value:	40
Number of channels:	1
HW specificatiom:	Frequency
Input Sampling Frequency:	0.0048828125 MHz
Clock Frequency:	100 MHz
Filter Architecture:	Systolic Multiply Accumulate
Coefficient Structure:	Inferred
Coefficient Type:	Signed
Coefficient Width:	16
Number of Paths:	4
Input Data Width:	32
Input Fractional Bits:	0
Output Rounding Mode:	Truncate LSBs
Output Width:	32
Registered Output:	✓
Optimization Goal:	Area
ND pin:	✓
Memory Options:	Automatic

Tabela B.15: Xilinx Core Generator - IQ FIR1

Core:	LogiCORE IP FIR Compiler v5.0
Component Name:	fir_iq_2
Instances Name:	fir2_iq
Coefficient Source:	COE File
Number of Coeff Sets:	1
Coefficients per set:	86
Filter Type:	Decimation
Rate Change Type:	Integer
Decimation Rate Value:	5
Number of channels:	1
HW specificatiom:	Frequency
Input Sampling Frequency:	0.0001220703125 MHz
Clock Frequency:	100 MHz
Filter Architecture:	Systolic Multiply Accumulate
Coefficient Structure:	Inferred
Coefficient Type:	Signed
Coefficient Width:	16
Number of Paths:	4
Input Data Width:	32
Input Fractional Bits:	0
Output Rounding Mode:	Truncate LSBs
Output Width:	32
Registered Output:	✓
Optimization Goal:	Area
ND pin:	✓
Memory Options:	Automatic

Tabela B.16: Xilinx Core Generator - IQ FIR2

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	IQ_FIFO
Instance(s):	iq_fifo_re, iq_fifo_im, iq_fifo_re_cx, iq_fifo_im_cx
Read/Write Clocks:	Common Clock - Shift Register
Read mode:	Standard FIFO
Write Width:	32
Write Depth:	128
Read Width:	32
Read Depth:	128
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Synchronous
Full Flags Reset Value:	0
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	Multiple Constants
Full Threshold Assert Value:	32
Full Threshold Negate Value:	31
Programmable Empty Type:	No Threshold

Tabela B.17: Xilinx Core Generator - IQ FIFOs

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	xk_fifo
Instance(s):	real_xk, imag_xk
Read/Write Clocks:	Common Clock - Shift Register
Read mode:	Standard FIFO
Write Width:	32
Write Depth:	512
Read Width:	32
Read Depth:	512
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Synchronous
Full Flags Reset Value:	0
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	No Threshold
Programmable Empty Type:	No Threshold

Tabela B.18: Xilinx Core Generator - FFT FIFOs

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	tx_fifo
Instance(s):	tx_fifi
Read/Write Clocks:	Independent Clocks - Block RAM
Read mode:	Standard FIFO
Write Width:	32
Write Depth:	2048
Read Width:	8
Read Depth:	8192
Almost Full Flag:	✓
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Asynchronous
Full Flags Reset Value:	1
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	No Threshold
Programmable Empty Type:	No Threshold

Tabela B.19: Xilinx Core Generator - FX2_Core TX FIFO

Core:	LogiCORE IP FIFO Generator v8.4
Component Name:	rx_fifo
Instance(s):	rx_fifi
Read/Write Clocks:	Independent Clocks - Block RAM
Read mode:	Standard FIFO
Write Width:	8
Write Depth:	2048
Read Width:	32
Read Depth:	512
Almost Full Flag:	✓
Read Port Handshaking:	Valid Flag (Active High)
Initialization:	Reset Pin, Enable Reset Sync
Reset Type:	Asynchronous
Full Flags Reset Value:	1
Use Dout Reset:	✓
Dout Reset Value:	0 (Hex)
Programmable Full Type:	No Threshold
Programmable Empty Type:	No Threshold

Tabela B.20: Xilinx Core Generator - FX2_Core RX FIFO

Apêndice C

TE0320: Configuração

C.1 Controlador USB Fx2 - Drivers/Firmware

O controlador USB quando conectado ao PC anfitrião pode ser reconhecido de duas maneiras diferentes dependente da posição do *switch* `S1A` da placa TE0320. A função deste *switch* é fazer com que a memória Erasable Programmable Read-Only Memory (EEPROM) que contêm o *firmware* do controlador possa ser acedida pelo mesmo (figura C.1b), adicionalmente o *boot up* do controlador depende da posição do *switch*, na posição ON executa o código do *firmware* e após o processo de arranque (enumeration) o dispositivo é reconhecido com base no *firmware*, quando se encontra OFF é reconhecido como um dispositivo genérico `(Generic Cypress USB Device)`.

O processo de configuração do controlador consiste nos seguintes pontos:

1. Instalação do driver genérico (secção 10.2.1 do UM);
2. Efectuar bootup do controlador com o switch `S1A` OFF e esperar que o dispositivo enumere como `Generic Cypress USB Device`;
3. Sem desligar a alimentação da placa mudar o switch `S1A` para a posição ON de modo ao controlador ter acesso à EEPROM;
4. Executar a aplicação `CyConsole` que permite escrever na EEPROM o ficheiro de firmware, instalar uma versão `3.x` do firmware [Ele12b];
5. Instalar o driver para a versão respectiva de Windows [Ele12a];
6. Fazer reset à placa mantendo `S1A` ON e desta vez, o controlador deverá enumerar de acordo com o conteúdo da EEPROM e ser reconhecido como `Trenz Electronic USB FX2`;

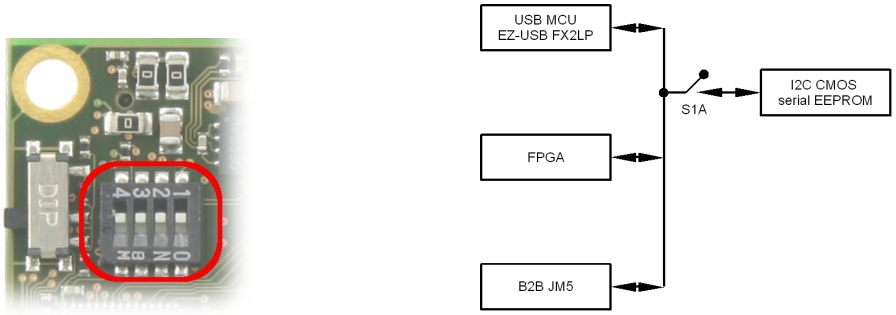
Após este procedimento o controlador está pronto para ser acedido através da API fornecida pela Trenz.

C.2 Programação da SPI PROM

A configuração da FPGA é executada no processo de boot up quando o módulo TE0320 é ligado ou quando é efectuado um master reset (MR) (push button branco da placa). A FPGA é configurada com o bitstream presente na memória PROM SPI.

O processo de escrita na memória PROM SPI pode ser efetuado de duas formas:

1. Através de cabo USB e da ferramenta USB `Firmware Upgrade Tool` (ver secção 10.2.5), apesar do nome esta ferramenta permite também fazer upload do bitstream para (FPGA). Este método é desaconselhado uma vez que é necessário reescrever o firmware cada vez que



(a) `TE0320`: DIP switches de configuração `S1[A:D]` (b) `TE0320`: Topologia do barramento I2C.

Figura C.1

se configura a FPGA, adicionalmente o processo demora mais tempo que o método descrito no ponto seguinte.

2. Configuração com cabo JTAG (ver secção 10.3 do UM) e da ferramenta iMPACT da Xilinx, este é o método aconselhado para configurar a FPGA. No processo de geração do ficheiro `.mcs` a opção `storage device` deverá ser selecionada com 32M (tamanho da memória SPI) e no menu `Select Attached SPI BPI` deverá ser selecionado `SPI PROM` → `M25P32`.

Apêndice D

Setup do ambiente de desenvolvimento de Software

D.1 Nokia Qt Framework

A aplicação gráfica desenvolvida neste trabalho foi realizada com base em Nokia QT (v4.8.2). Em termos de ambiente de desenvolvimento foi usado o Microsoft Visual Studio 2010 conjuntamente com o Add-in de QT.

D.2 Instruções para instalação da biblioteca *qwt*

Instalar a biblioteca de widgets gráficos `qwt` [Rat12]: fazer download do pacote zip e extrair para um diretório de instalação (sugerido: `C:/Qwt-6.0.1/`), realizar os passos descritos no ficheiro `INSTALL` descritos na secção `Building Qwt -> win32/MSVC` para compilar a biblioteca, a linha de comandos a utilizar deverá ser o `Visual Studio Command Prompt (2010)` executado em modo de administrador. Adicionar a seguinte variável de ambiente `QT_PLUGIN_PATH = /Qwt-6.0.1/plugins` às variáveis do sistema (System variables) de modo a que o QT Designer reconheça os widgets, adicionar também à variável de utilizador `PATH` os seguintes valores: `C:/QT/4.8.2/bin` e `C:/qwt-6.0.1/lib` para que o programa em runtime tenha os caminhos para os dlls necessários.

D.3 Parâmetros de compilação/linkagem para o uso de QT em ambiente MSVC10

Abrir o ficheiro `.sln` que contêm a árvore do projecto, na aba do Solution Explorer selecionar `TE0320->right click->properties`, na secção `configuration properties` → `C/C++` → `General` → `Addional Include Directories` certificar que os caminho para ($(qwt_dir)/include) se encontra correcto, na secção `Linker->general->additional library dependencies` certificar que os caminhos para `$(qt_dir)/lib` e `$(qwt_dir)/lib` se encotram correctos.

O projecto foi realizado com a versão de 32 bits da biblioteca da Cypress (TE_USB_FX2_CyAPIdll), para compilação em sistemas de 64 bits deverá ser utilizada a versão correta da biblioteca, com as alterações descritas acima para a utilização da biblioteca `qwt`.

Apêndice E

Aplicação de interface com a FPGA

A aplicação de *software* desenvolvida permite recolher e observar em tempo real as componentes I/Q e o resultado da FFT de ambos os canais, para além disso inclui também algumas utilidades de controlo sobre os FIFOs do controlador FX2, uma imagem do programa é apresentada na figura E.1.

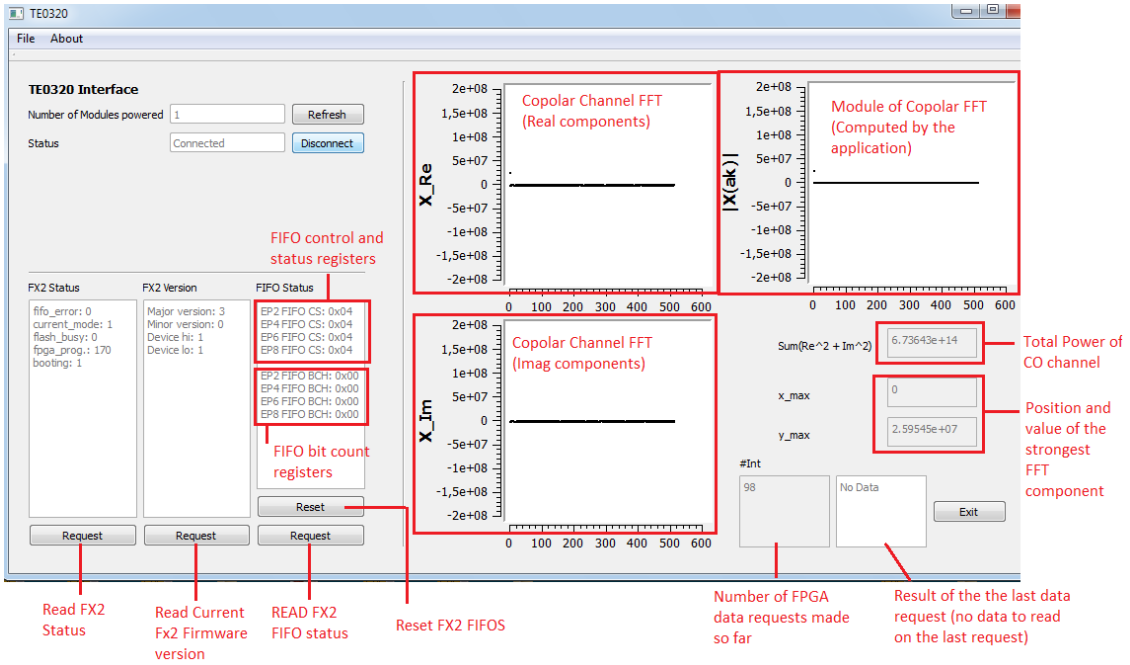


Figura E.1: Screenshot da aplicação desenvolvida.

Bibliografia

[BC07] V.S. Bagad and J.S. Chitode. *Communications Systems*. Ed. by Technical Publications. 2007.

[Bla76] Alain Blanchard. *Phase Locked Loops - Application to coherent receiver design*. Ed. by John Wiley & Sons. 1976.

[Ele12a] Trenz Electronic. *Trenz gen3 API*. Nov. 2012. URL: http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_3.html.

[Ele12b] Trenz Electronic. *Trenz git-hub repository*. Nov. 2012. URL: https://github.com/Trenz-Electronic/TE-USB-Suite.

[Ele13a] Trenz Electronic. *Figure 1/2*. June 2013. URL: http://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/TE0320_series/TE0320/documents/UM-TE0320.pdf.

[Ele13b] Trenz Electronic. *Figure 3*. June 2013. URL: http://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/TE0320_series/TE0320/documents/UM-TE0320.pdf.

[Ele13c] Trenz Electronic. *TE0320 board website*. June 2013. URL: http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/te0320-spartan-3a-dsp-series.html.

[Ele13d] Trenz Electronic. *TE0630 board website*. June 2013. URL: http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/te0630-spartan-6-series.html.

[Gar66] Floyd M. Gardner. *Phaselock Techniques*. Ed. by Inc. John Wiley & Sons. 1966.

[JCo98] O.Teong J.Cornelis B.Bowthorpe. "A DSP Based Satellite Beacon Receiver and Radiometer". MA thesis. 1998.

[Kes13] Walt Kester. *The Data Conversion Handbook. Figure 3.65.* June 2013. URL: http://www.analog.com/library/analogdialogue/archives/39-06/Chapter%203%20Data%20Converter%20Architectures%20F.pdf.

[Pir07] Jorge A.N.P. Pires. "Receptor de dois canais para balizas de satélite". MA thesis. Universidade de Aveiro, 2007. URL: http://hdl.handle.net/10773/1898.

[Rat12] Uwe Rathmann. *Qwt - Qt Widgets for Technical Applications*. Nov. 2012. URL: http://sourceforge.net/projects/qwt/files/qwt/6.0.1/qwt-6.0.1.zip/download.

[Roh12] Rohde&Schwarz. *Time Domain Oscillator Stability Measurement - Allan variance*. Application Note. Nov. 2012. URL: http://www2.rohde-schwarz.com/file_11752/1EF69_E3.pdf.

[Sem12] Cypress Semiconductor. *EZ-USB Technical Reference Manual*. Nov. 2012. URL: http://www.cypress.com/?docID=27095.

[Sou07] Ricardo L.M. Sousa. "Receptor digital para medição de balizas de satélite". MA thesis. Universidade de Aveiro, 2007. URL: http://hdl.handle.net/10773/1976.

[Sou11] Marco Sousa. "Receptor de Propagação para Satélite Alphasat". MA thesis. 2011.

[Xil12] Xilinx. *IP LogiCORE FIR Compiler v5.0*. DS534. Nov. 2012. URL: http://www.xilinx.com/support/documentation/ip_documentation/fir_compiler_ds534.pdf.