

# Implementação de Algoritmos em FPGA para Estimação de Sinal em Sistemas Ópticos Coerentes

Nuno M. Pinto      Henrique M. Salgado  
DEEC, FEUP      INESC Porto, FEUP  
ee03171@fe.up.pt      hsalgado@inescporto.pt

João C. Ferreira  
INESC Porto, FEUP  
jcf@fe.up.pt

Luís M. Pessoa  
INESC Porto, FEUP  
luis.pessoa@ieee.org

## Resumo

Neste artigo descreve-se a implementação em FPGA de algoritmos para estimação de sinal em sistemas ópticos coerentes. A ferramenta de desenvolvimento usada para a implementação destes em hardware foi o *System Generator*. Foram criadas duas implementações dos algoritmos estudados, uma sequencial e outra paralela. Obtiveram-se resultados em termos de taxas de transmissão e do desempenho de cada uma das implementações, que permitiram avaliar a utilização de recursos na FPGA.

Foram, ainda, implementados compensadores para 200 km e 500 km de dispersão para a fibra óptica SSMF. Cada equalizador, juntamente com o compensador, foi testado para sistemas de transmissão com modulação 4QAM e 16QAM. Os resultados são bastante promissores justificando a aposta nesta tecnologia.

## 1. Introdução

Os sistemas ópticos coerentes têm vindo a ganhar importância nas comunicações por fibra óptica. Uma grande vantagem destes sistemas, comparativamente aos sistemas de Modulação em Intensidade e Detecção Directa (IM/DD), consiste na possibilidade de usar na transmissão vários tipos de modulação, nomeadamente a modelação em fase (M-PSK) e constelações multi-nível (M-QAM), havendo a preservação da fase do campo eléctrico do domínio óptico para o domínio eléctrico, sendo o sinal amostrado à taxa de *Nyquist*. Adicionalmente, é possível compensar as penalidades lineares do canal de transmissão, bem como a dispersão cromática (CD) e a dispersão do modo de polarização (PMD) através de um filtro linear [7], que pode operar de forma adaptativa para superar as distorções do sinal ao longo do tempo. Estes sistemas ganharam um renovado interesse devido à disponibilidade de Processamento Digital de Sinal (PDS) de alta velocidade, o que permite que operações complexas sejam realizadas no domínio digital, dando origem a um receptor óptico reconfigurável.

Os conversores analógico-digitais serão capazes de satisfazer brevemente as elevadas taxas de amostragem requeridas em sistemas de transmissão ópticos de alta velocidade. A evolução é no sentido de se usar conversores de elevado desempenho (>40 GSample/s) que aliados a FPGAs (*Field Programmable Gate arrays*) tornam estes sistemas realizáveis em tempo real. Existem já no mercado

conversores de 30 GSample/s, com capacidades especiais para interligação com FPGA [5] [4].

A implementação em FPGA é uma boa opção para estes sistemas, uma vez que é uma plataforma de aplicação muito flexível e moldável à situação pretendida. De facto, foram realizadas várias experiências de transmissão a alta velocidade recorrendo a FPGAs e conversores rápidos [2] [9] [17] [8].

No receptor, a fase do Oscilador Local (OL) deve ser sincronizada com a fase do sinal, para evitar as dificuldades associadas ao uso de uma OPLL (*Optical Phase-Locked Loop*). Essa sincronização pode ser feita em DSP através de algoritmos de estimação de fase digital, onde o OL fica em funcionamento livre. Os algoritmos apropriados para a estimação de fase e compensação de dispersão foram estudado em [13] [14]. A implementação destes algoritmos usando FPGAs com capacidade de processamento paralelo [10] são discutidas neste artigo. Para este trabalho foram implementados algoritmos com compensação de dispersão adaptativa usando a ferramenta *System Generator* da Xilinx.

Após a introdução, segue-se na secção 2 uma descrição da equalização adaptativa, que retrata os algoritmos usados no trabalho, bem como o funcionamento do módulo para a compensação da dispersão. A secção 3 apresenta a implementação em FPGA e as ferramentas de desenvolvimento. Os resultados são apresentados na secção 4 e, por último, na secção 5 são dadas as conclusões.

## 2. Equalização adaptativa

Um equalizador adaptativo é um filtro localizado no receptor que recebe os dados provenientes de um canal de transmissão, por exemplo a fibra óptica. A fibra óptica, como qualquer canal, introduz distorções no sinal transmitido. Ora, um equalizador adaptativo vai, através de algoritmos, adaptar-se às características adversas do canal para, deste modo, as poder compensar no sinal transmitido, ou seja, o equalizador vai retirar as distorções que o sinal sofre ao longo do canal dispersivo, ficando este o mais semelhante possível ao sinal originalmente transmitido.

Os algoritmos podem ser auto-adaptativos, como é o caso do algoritmo CMA (*Constant Modulus Algorithm*) cuja característica é convergir para um módulo constante, ou serem supervisionados e necessitando de uma sequência de treino, como é o caso do algoritmo LMS (*Least Mean*

*Square*) cuja característica é possuir um decisor.

## 2.1. O Algoritmo CMA

O algoritmo CMA realiza uma equalização cega e foi estudado por Sato em 1975 [15] e mais tarde por Godard em 1980 [6]. O modo de funcionamento deste algoritmo é atingir a convergência perante aplicações com envolvente constante, ou módulo constante [1]. O CMA atinge a convergência adaptando automaticamente os seus coeficientes às características do canal.

Este algoritmo é o mais usado em equalizadores adaptativos, essencialmente por causa da sua robustez e baixa complexidade, efectuando a estimação dos dados através da minimização da função custo pelo método do gradiente descendente [11].

O processo do equalizador CMA passa por três etapas, onde a primeira é dada pela equação seguinte:

$$y(n) = \mathbf{w}^H(n) \cdot \mathbf{u}(n) \quad (1)$$

onde  $y(n)$  representa o sinal à saída do equalizador, obtido através da convolução dos coeficientes do equalizador ( $\mathbf{w}^H(n)$ ) com o sinal que se pretende equalizar ( $\mathbf{u}(n)$ ). Em seguida o erro é calculado da seguinte forma:

$$e(n) = y(n) \cdot (R_2 - |y(n)|^2) \quad (2)$$

onde  $R_2$  é uma constante que depende da constelação seleccionada. Para QPSK, o valor de  $R_2$  é unitário.

Por último, a actualização dos coeficientes é dada por:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \cdot \mathbf{u}(n)e^*(n) \quad (3)$$

onde  $\mu$  é o passo de adaptação do algoritmo.

Uma característica importante deste algoritmo é a inicialização dos coeficientes, já que dela depende o sucesso da convergência do CMA. A inicialização mais usada (e também utilizada neste trabalho) é a denominada de *center spike*, e consiste em colocar o coeficiente central igual à unidade, enquanto todos os outros coeficientes são colocados a zero.

Apesar de não tomar em consideração a fase do sinal, o que origina a rotação da constelação, este algoritmo apresenta um bom desempenho, permitindo convergir mesmo se houver ruído de fase elevado. Porém, uma vez atingida a convergência há benefícios em comutar para uma arquitectura baseada em decisões dos símbolos estimados — o LMS (*Least Mean Squares*) — onde a fase do sinal é considerada.

## 2.2. Algoritmo LMS

O algoritmo LMS, o mais popular de todas as estimativas, foi sugerido por Widrow e Hoff em 1959 [18] e consiste simplesmente em substituir os valores médios das variáveis pelos seus valores instantâneos [1].

É um método estocástico de gradiente descendente em que os coeficientes do filtro adaptativo são obtidos por forma a minimizar o erro quadrático médio da diferença entre o valor decidido e o valor estimado do sinal.

Este algoritmo é em tudo semelhante ao CMA excepto no cálculo do erro. Para tal é necessário um módulo extra que realiza a decisão do símbolo à saída do equalizador. O erro pode ser calculado da seguinte forma:

$$e(n) = d(n) - y(n) \quad (4)$$

onde  $d(n)$  é o símbolo dado pelo decisor.

Uma outra característica, que representa uma dificuldade deste algoritmo, é a necessidade de inicialização dos coeficientes. Uma forma de efectuar a inicialização é aplicar uma sequência de treino conhecida até atingir a convergência, sendo depois comutado para o modo Dedicado à Decisão, onde fica a funcionar sem qualquer apoio de treino. Contudo há uma maneira mais eficaz de obter a inicialização dos coeficientes do LMS, evitando sequências de treino: usar o CMA numa fase inicial até se atingir a convergência e depois utilizar esses coeficientes para inicializar o algoritmo LMS. Esta abordagem é a usada neste trabalho.

## 2.3. Compensação da dispersão

A compensação da dispersão pode ser realizada no domínio óptico ou no domínio eléctrico. No domínio óptico é possível realizar a compensação através de fibras com dispersão cromática contrária à do sistema de transmissão. No domínio eléctrico é possível compensar a dispersão cromática com a ajuda de filtros de resposta impulsional finita. Este método diminui a complexidade associada à compensação no nível óptico.

Recorrendo ao processamento de sinal este módulo pode ser implementado em conjunto com os equalizadores adaptativos discutidos anteriormente. Assim, colocando um módulo deste tipo antes dos algoritmos de equalização, LMS ou CMA, é possível compensar a maior parte da dispersão, sendo o equalizador adaptativo responsável pela compensação da dispersão residual e efeitos variantes no tempo como a PMD.

A implementação do módulo de compensação é realizada em FPGA através de um filtro de coeficientes fixos, cujos valores são calculados de acordo com a transformada inversa de *Fourier* da função de transferência da fibra representada pela equação:

$$G(z, \omega) = \exp\left(-j \frac{D\lambda^2 z}{4\pi c} \omega^2\right) \quad (5)$$

onde  $D$  representa o coeficiente de dispersão da fibra,  $\lambda$  o comprimento de onda,  $z$  a distância de transmissão,  $c$  a velocidade da luz e  $\omega$  a frequência angular.

O filtro de compensação da dispersão é dado por um filtro-passa tudo com a característica  $1/G(z, \omega)$  e pode ser construído usando tanto filtros digitais recursivos como não-recursivos [16].

### 3. Implementação em FPGA

#### 3.1. Metodologia de desenvolvimento

Relativamente à programação da FPGA, existe um conjunto de ferramentas de software associado a um fluxo de projecto que proporciona um alto nível de abstracção ao programador, permitindo que este se foque no algoritmo que deseja implementar em vez de se preocupar com os circuitos que serão implementados. Desta forma, a programação do dispositivo pode ser feita ou através de uma linguagem de descrição de hardware (VHDL ou Verilog) ou recorrendo à ferramenta de modelização de sistemas — *System Generator*.

O *System Generator* é uma ferramenta de projecto integrado com FPGAs, que utiliza como suporte de desenvolvimento o *Simulink*, a ferramenta de modelização, simulação e análise de sistemas dinâmicos do MATLAB.

Além do *Simulink*, o *System Generator* utiliza um conjunto de ferramentas para especificar os detalhes de implementação de hardware em dispositivos da Xilinx. Mas é no *Simulink* que o *System Generator* é apresentado sob a forma de uma biblioteca adicional (*Xilinx Blockset*).

A ferramenta de desenvolvimento *System Generator* possibilita ao utilizador desenvolver algoritmos sofisticados e sistemas de processamento de sinal, abstraindo-se de funções complexas de matemática, lógica, memória ou PDS. A biblioteca da Xilinx no *Simulink* possui também blocos que proporcionam interfaces com outras ferramentas, bem como outros que geram automaticamente o código VHDL ou Verilog [20].

No *Simulink* podem ser usados os blocos das bibliotecas *Simulink* em conjunto com os do *System Generator*. No entanto, há que ter em linha de conta alguns aspectos importantes. O subsistema a implementar em hardware deve ser constituído apenas por elementos do *Xilinx Blockset*. As entradas e saídas deste subsistema são obrigatoriamente constituídas por blocos *Gateway In* e *Gateway Out*, respectivamente. Estes blocos definem a fronteira da FPGA no ambiente de simulação e realizam, portanto, a conversão dos dados entre os formatos internos de MATLAB (números de vírgula flutuante) e os formatos usados no processamento PDS em FPGA (vírgula fixa).

A definição da conversão dos dados de entrada deve encontrar um compromisso entre a precisão requerida pelo algoritmo a ser implementado e a utilização de recursos de hardware. Embora o sistema admita operandos com 4096 bits [19], tais dimensões são claramente superiores ao que é utilizável na prática. A utilização de representação em vírgula fixa leva a que seja necessário descartar alguma informação, levando a desvios, seja por *overflow* (nos bits mais significativos) seja por quantização (nos bits menos significativos).

Neste trabalho o número de bits usados para a representação dos dados é 18 bits, sendo que 1 bit é reservado para o sinal, 5 bits para a parte inteira e os restantes 12 bits para a parte fraccionária. O que pesou nesta escolha foi o facto de haver FPGAs que disponibilizam multiplicadores  $18 \times 18$  dedicados.

A maioria dos blocos do *Xilinx Blockset* permite ao utilizador escolher a precisão que melhor se ajusta ao projecto, mas também é possível deixar o sistema deduzir o número de bits necessário para representar os resultados de cada bloco a partir das características dos respectivos dados de entrada. A propagação automática das dimensões dos dados leva geralmente ao seu crescimento ao longo da cadeia de cálculo (por exemplo, os produtos requerem uma representação com o dobro dos bits dos operandos). Quando existe realimentação de dados (como é o caso para os algoritmos em consideração, cf. figura 1), o mecanismo de dedução de dimensões falha, pelo que é imprescindível especificar pelo menos algumas dessas dimensões.

O *System Generator* permite realizar uma co-simulação software/hardware, com a parte do sistema especificada com elementos do *Xilinx Blockset* a ser executada em FPGA, enquanto os outros elementos são simulados em MATLAB. Trata-se de uma forma cómoda de validar parcialmente o hardware desenvolvido sem prescindir das ferramentas e da comodidade associadas ao ambiente MATLAB.

Além do *Simulink*, o *System Generator* utiliza um conjunto de ferramentas para especificar os detalhes de implementação de hardware em dispositivos da família Xilinx. Para tal, o *System Generator* utiliza a biblioteca *Xilinx DSP Blockset*, também instalada no *Simulink* e para gerar a *Netlist* otimizada dos módulos PDS invoca automaticamente o *Xilinx Core Generator*. Opcionalmente, pode-se gerar um *testbench* para usar no *ModelSim* ou no *Xilinx ISE Simulator* para aprofundar o nível de detalhe do projecto a implementar.

#### 3.2. Estrutura da Implementação em Hardware

Os algoritmos em estudo foram desenvolvidos em *System Generator*. Foram implementadas duas configurações diferentes para cada algoritmo, uma sequencial e outra paralela. Conforme a própria designação indica, na versão sequencial os dados são tratados em cadeia sequencial, enquanto que a versão paralela realiza o tratamento dos dados em paralelo.

Apresenta-se aqui o diagrama de blocos de cada um dos algoritmos, começando pelo diagrama do CMA ilustrado na figura 1.

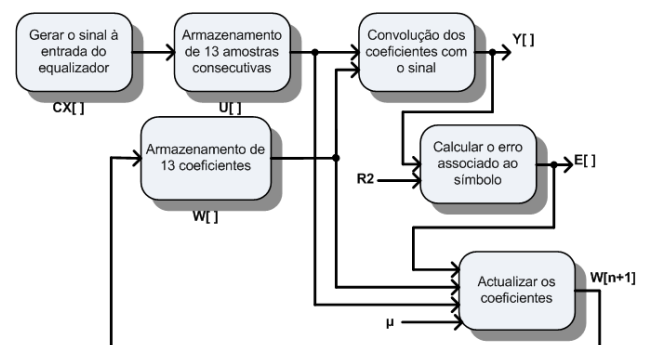


Figura 1. Diagrama do algoritmo CMA.

O CMA armazena quer os dados do sinal, quer os coeficientes do algoritmo. O armazenamento dos dados difere de acordo com a versão a implementar. Na versão sequencial foi utilizado o *Addressable Shift Register (ASR)*, pois o seu simples funcionamento permite apresentar a cada ciclo de relógio uma amostra do sinal, e ainda fazer um deslocamento das amostras descartando as mais antigas. Para a versão paralela foram usados registos individuais interligados, a fim de criar um ASR mas em paralelo. A vantagem deste segundo caso é que as amostras estão disponíveis todas em simultâneo.

Para os coeficientes também foram adoptadas duas abordagens diferentes para cada versão. Na versão sequencial foi usada uma *Dual Port RAM*, cuja vantagem é poder ler e escrever no mesmo ciclo de relógio. Esta vantagem é importante, pois é necessário ler os coeficientes actuais e escrever os novos coeficientes actualizados pelo algoritmo. Já para a configuração paralela o processo usado foi simplesmente um registo controlado por um relógio para garantir que a escrita dos novos coeficientes só é realizável quando os coeficientes antigos já não são necessários.

O número de coeficientes usados pelo algoritmo, bem como o número consecutivo de amostras é de 13, conforme indicado em [16]. As amostras e os coeficientes são utilizados pelo bloco que realiza a convolução dos coeficientes com o sinal, de acordo com a equação 1.

O bloco que calcula o erro associado ao símbolo estimado (pelo bloco do filtro) executa uma operação correspondente à equação 2. Seguidamente são actualizados os coeficientes através da equação (3) e guardados para utilização no próximo ciclo do algoritmo. Também nestes três últimos casos foi necessário criar uma implementação diferente para cada uma das versões implementadas.

A diferença do algoritmo LMS para o anterior é o modo como calcula o erro, que foi implementado como indica o diagrama da figura 2.

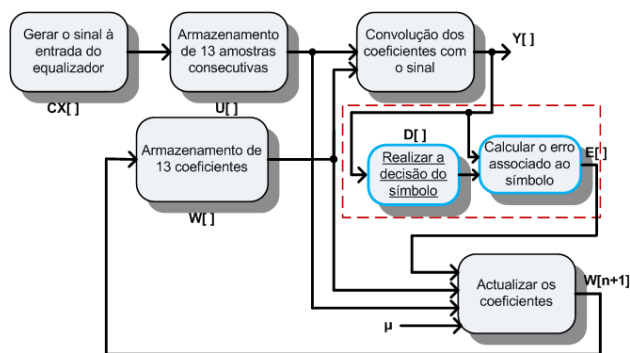


Figura 2. Diagrama do algoritmo LMS.

Importa referir que as amostras do sinal são valores complexos, pelo que é necessário tratar a parte real e imaginária separadamente, uma vez que a FPGA não realiza operações com números complexos. Para tal, quer as amostras, quer os coeficientes são separados em parte real e imaginária, e todas as operações são realizadas tendo em conta essa separação.

## 4. Resultados obtidos

### 4.1. Resultados de implementação

Para cada algoritmo (CMA e LMS) foram implementadas duas versões, como já referido. A realização destas duas versões permitiu obter indicações sobre os requisitos dos algoritmos em termos de ocupação da FPGA e sobre as taxas de transmissão que o sistema de transmissão em estudo poderá suportar.

A versão paralela permite naturalmente obter taxas de transmissão mais altas, à custa de uma maior ocupação de recursos na FPGA como se verifica na tabela seguinte.

Por outro lado, apesar de as taxas de transmissão serem mais baixas a implementação da configuração sequencial permite executar o algoritmo em FPGAs com menor número de recursos.

Tabela 1. Comparação entre as versões sequencial e paralela (Virtex-5 XC5VLX330T).

Componentes	Sequencial	Paralela
Slices	1070	6944
Flip-flops	939	5285
BRAMs	2	0
LUTs	1758	9933
IOBs	108	108
Multiplicadores	14	60
<b>Características</b>		
Max. Frequência	111,669 MHz	108,530 MHz
Min. Período	8,955 ns	9,212 ns
<b>Tempos mínimos</b>		
Latência (períodos)	26	10
Tempo de Símbolo (ns)	232,83	92,12
Taxa de Transmissão (MSímbolos/s)	4,29	10,86

A tabela mostra a alocação de recursos, tempos e taxas de transmissão para uma FPGA da família Virtex 5 (XC5VLX330T). É notório que a configuração em paralelo ocupa cerca de 6 vezes mais recursos que a configuração sequencial. No que se refere a multiplicadores dedicados a sua utilização é muito superior.

Um aspecto muito importante e que caracteriza a implementação de cada configuração é a latência, onde se observa que a versão paralela reduz a latência de 26 para 10 períodos em relação à sequencial. Dessa latência depende a taxa de transmissão onde a configuração paralela atinge perto de 11 MSímbolos/s enquanto a sequencial se fica pelos 4 MSímbolos/s.

Esta implementação, tendo sido realizada usando o System Generator, é válida para outros modelos de FPGA, bastando para o efeito gerar uma nova *Netlist*. Esta é uma das grandes vantagens do System Generator e foi o motivo porque se usou esta técnica. Com isso foi possível obter os limites do sistema apresentados na Tabela 2 para várias

plataformas FPGAs em termos de taxas de transmissão.

Tabela 2. Taxas de transmissão em MSímbolos/s para várias plataformas de FPGAs.

Plataforma FPGA	Taxas de transmissão	
	Versão Sequencial	Versão Paralela
Spartan 3A XC3S700A	1,92	—
Virtex 4 XC4VLX60	2,23	9,71
Virtex 4 XC4VSX55	2,61	7,49
Virtex 5 XC5VLX330T	4,29	10,86
Virtex 5 XC5VSX95T	4,20	10,11

Em resumo, o sistema de transmissão em estudo pode atingir 10,86 MSímbolos/s na configuração paralela usando a FPGA Virtex 5 XC5VLX330T.

## 4.2. Resultados de simulação

Para os resultados apresentados em seguida, foram simulados o transmissor e o canal do sistema de transmissão. Para tal foi usada a codificação de impulsos NRZ, obtida através de um *trem* rectangular de impulsos ideais juntamente com um filtro de *Bessel* passa-baixo de 5ª ordem com 3 dB de largura de banda, a 80% da taxa de transmissão dos símbolos. É usado também um filtro *anti-alias*, constituído por um filtro de *Bessel* passa baixo de 3ª ordem.

Os resultados foram obtidos para a modulação 4QAM e posteriormente 16QAM, com 200 km de fibra óptica. Para cada tipo de constelação obtiveram-se as constelações à entrada do equalizador e à saída do compensador. Obtiveram-se ainda seguida as constelações que demonstram o desempenho quer do algoritmo CMA, quer do algoritmo LMS.

Na figura 3(a) estão representados os símbolos após viajarem através de fibra óptica com 200 km de comprimento para a constelação 4QAM, onde se observa que os dados aparecem bastante distorcidos. Para retirar a dispersão cromática referente a esses 200 km de fibra é necessário um compensador de dispersão constituído por coeficientes fixos cujo número de coeficientes varia consoante a distância de fibra óptica. Para o comprimento de 200 km são necessários 13 coeficientes para obter compensação, como é visível na figura 3(b).

Os algoritmos adaptativos descritos nas secções anteriores são capazes, por si só, de compensar a dispersão cromática da fibra até 200 km de comprimento, quando 13 coeficientes são utilizados; no entanto a rapidez de convergência dos algoritmos é afectada. Daí a necessidade de desenvolver este filtro compensador de coeficientes fixos para 200 km de fibra, a fim de melhorar os tempos de

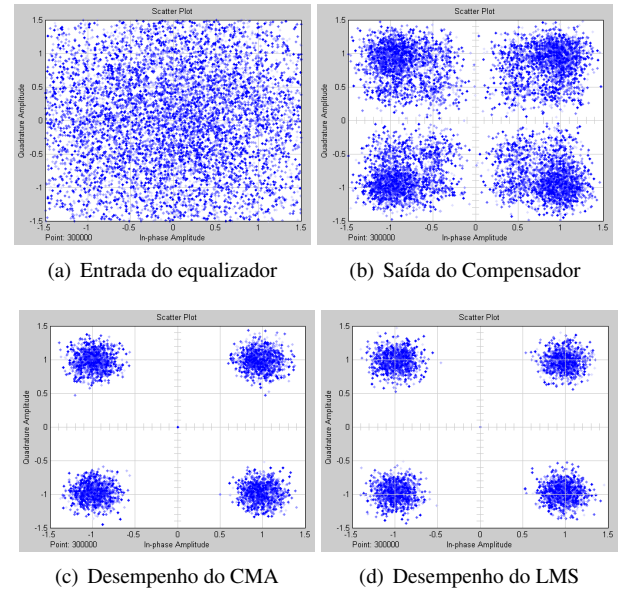


Figura 3. Constelação 4QAM para 200 km de fibra.

convergência. Foi ainda desenvolvido um outro compensador de dispersão cromática mas para 500 km, apresentando os dois compensadores óptimos resultados, sendo possível consultar este resultado em [12].

Como se observa na figura 3, o desempenho dos algoritmos adaptativos é bastante razoável pelo que a constelação 4QAM aparece bem definida, com uma nuvem à volta de cada ponto da constelação quer para o algoritmo CMA, quer para o algoritmo LMS. Verifica-se também que os compensadores têm um papel importante pois vão aumentar o desempenho dos algoritmos.

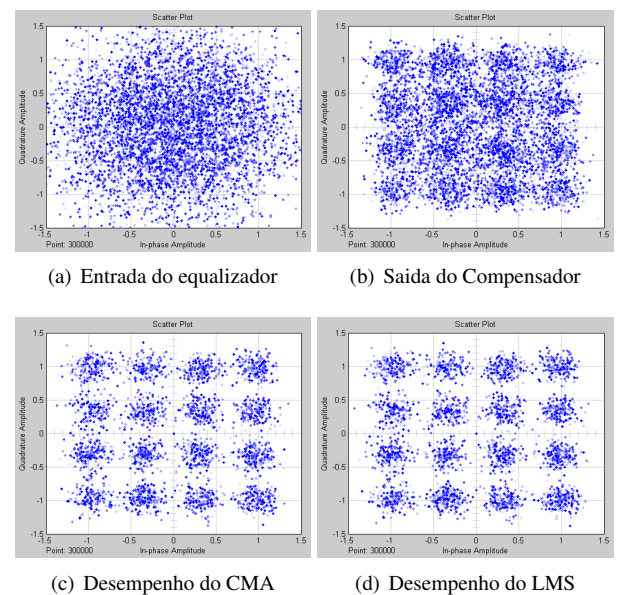


Figura 4. Constelação 16QAM para 200 km de fibra.

Para o caso da constelação 16QAM (figura 4) nota-se o bom desempenho dos algoritmos CMA e LMS, como no



caso anterior.

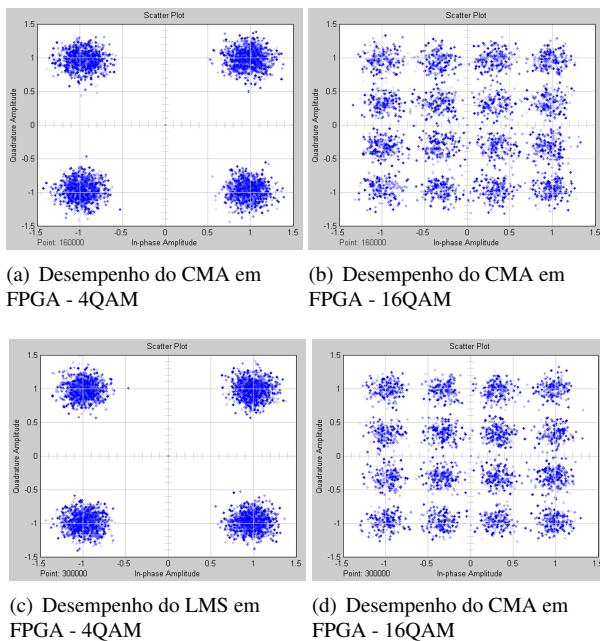


Figura 5. Resultados após co-simulação em hardware para 4QAM e 16QAM.

Posto isto, na figura 5 são apresentados os resultados da implementação dos algoritmos em hardware, usando a FPGA Virtex 4. Apresenta-se o desempenho do CMA para 4QAM (figura 5(a)) e 16QAM (figura 5(b)) onde é perceptível o bom funcionamento deste algoritmo como era de esperar, comparando com os resultados anteriores.

O mesmo se passa com o algoritmo LMS, onde a figura 5(c) mostra o resultado para a constelação 4QAM e para 16QAM o resultado é exibido pela figura 5(d).

Verifica-se, portanto, que os algoritmos CMA e LMS apresentam bom desempenho, uma vez que as respectivas constelações se apresentam bem definidas após o seu funcionamento.

## 5. Conclusões e Trabalho Futuro

Este trabalho envolveu o estudo e a implementação de algoritmos adaptativos para equalização de sinal em sistemas ópticos coerentes. A implementação foi realizada em *System Generator* e implementada em hardware através de co-simulação.

Esta metodologia é extremamente recente e é possível, graças ao poder de abstracção da ferramenta de desenvolvimento *System Generator*, obter resultados em simulação e/ou transportá-los para o ambiente MATLAB, a fim de serem tratados para boa interpretação dos mesmos.

Um tipo adicional de paralelismo pode ser implementado nestes algoritmos, em que o processamento dos símbolos é realizado em paralelo, isto é, utilizando vários módulos com implementações quer do CMA, quer do LMS, consegue-se reduzir a latência de cada algoritmo. Estes podem apresentar na saída um símbolo a cada dois

ciclos de relógio, reduzindo grandemente a latência de execução. Cada módulo teria os seus próprios coeficientes que seriam actualizados por cada instância colocado em paralelo.

Uma outra funcionalidade interessante seria a implementação de algoritmos *FeedForward*. Estes algoritmos poderão desempenhar um papel fundamental na sincronização da portadora óptica, aquando da presença de ruído de fase que o transmissor provoca.

## Referências

- [1] Sílvia A. Abrantes, *Processamento Adaptativo de Sinais*, Fundação Calouste Gulbenkian, Lisboa, 2000.
- [2] S. Chen, Q. Yang, Y. Ma, and W. Shieh, *Multi-gigabit real-time coherent optical OFDM receiver*, Optical Fibre Communication/National Fibre Optic Engineers Conference (OFC/NFOEC), (OSA, 2009), Paper OTuO4, 2009.
- [3] J. Machado da Silva, J. Canas Ferreira, and J. Correia Lopes, *Modelo de escrita e formatação de dissertações/relatórios de projecto do MIEEC*, Maio 2008.
- [4] Micram Microelectronic GmbH, *25GS/s Digital-to-Analog Converter (DAC) Demonstrator*, 2009.
- [5] ———, *VEGA ADC30. 30GS/s / 6-bit High-Speed Analog to Digital Converter*, 2009.
- [6] D. Godard, *Self-recovering equalization and carrier tracking in two-dimensional data communication systems*, IEEE transactions on communications **28** (1980), no. 11, 1867–1875.
- [7] E. Ip, A. Lau, D. Barros, and J. M. Kahn, *Coherent Detection in Optical Fiber Systems*, Optics Express **16** (2008), no. 2, 753–791.
- [8] N. Kaneda, Q. Yang, X. Li, W. Shieh, and Y.K. Chen, *Realizing Real-Time Implementation of Coherent Optical OFDM Receiver with FPGAs*, Proceedings-ECOC 2009 (2009).
- [9] A. Leven, N. Kaneda, and Y.K. Chen, *A real-time CMA-based 10 Gb/s polarization demultiplexing coherent receiver implemented in an FPGA*, Proceedings of the Conference on Optical Fibre Communications (OFC 2008), San Diego, CA, USA, Paper OTuG3, 24th-28th February, 2008.
- [10] A. Leven, N. Kaneda, A. Klein, U.-V. Koc, and Y.-K. Chen, *Real-Time Implementation of 4.4 Gbit/s QPSK Intradyne Receiver Using Field Programmable Gate Array*, Electronics Letters **42** (2006), no. 24, 1421–1422.
- [11] Xi-Lin Li and Xian-Da Zhang, *A Family of Generalized Constant Modulus Algorithms for Blind Equalization*, IEEE Transactions on Communications **54** (2006), no. 11, 1913–1917.
- [12] J. C. Ferreira N.M. Pinto, L. M. Pessoa and H. M. Salgado, *FPGA Implementation of Signal Processing Algorithms in Coherent Optical Systems*, SEON 2009 (Amadora), June 2009.
- [13] L. M. Pessoa, H. M. Salgado, and I. Darwazeh, *Joint Mitigation of Optical Impairments and Phase Estimation in Coherent Optical Systems*, IEEE LEOS Summer Topical Meetings 2008 (Mexico), July 2008, Paper TuE4.3, pp. 169–170.
- [14] L. M. Pessoa, H. M. Salgado, and I. Darwazeh, *Performance evaluation of phase estimation algorithms in equalized coherent optical systems*, IEEE Photonics Technology Letters **17** (2009), 1181–1183.
- [15] Y. Sato, *A Method of Self-Recovering Equalization for Multi-level Amplitude Modulation*, IEEE transactions on communications **23** (1975), 679–682.

- [16] S. Savory, *Digital filters for coherent optical receivers*, Optics Express **16** (2008), no. 2, 804–817.
- [17] R. Waagemans, S. Herbst, L. Holbein, P. Watts, P. Bayvel, C. Fürst, and R.I. Killey, *10.7 Gb/s electronic predistortion transmitter using commercial FPGAs and D/A converters implementing real-time DSP for chromatic dispersion and SPM compensation*, Optics Express **17** (2009), no. 10, 8630–8640.
- [18] B. Widrow and M.E. Hoff, *Adaptive switching circuits*, (1960).
- [19] Xilinx, *System Generator for DSP - Reference Guide*, Release 10.1, March 2008.
- [20] ———, *System Generator for DSP - User Guide*, Release 10.1, March 2008.