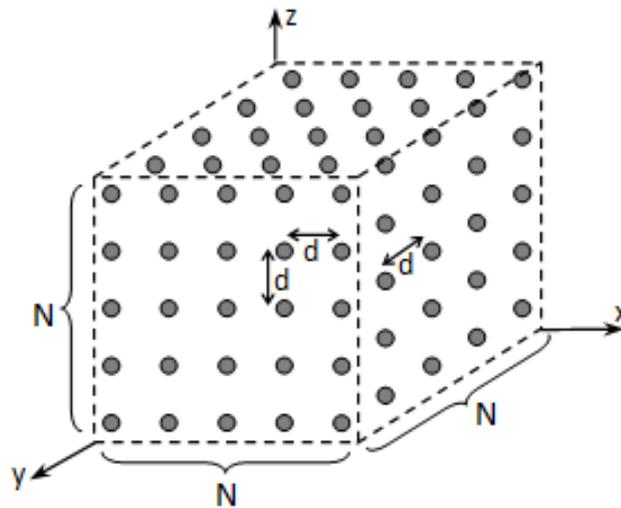




Carlos Manuel
Bernardes Romeiro

Hardware para paralelização de malhas de guias-de- onda digitais





**Carlos Manuel
Bernardes Romeiro**

**Hardware para paralelização de malhas de guias-de-
onda digitais**

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor António Guilherme Rocha Campos e do Doutor Arnaldo Silva Rodrigues de Oliveira, Professores Auxiliares do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus avós por todo o carinho que me deram; aos meus pais e irmã por todo o amor e por todo o incentivo que sempre me deram; a uma pessoa muito especial, a Mafalda, por todo o apoio que me deu e aos meus amigos, em especial ao Flávio Jorge e ao Diogo Almeida, pelos bons momentos que me proporcionaram.

o júri

Presidente

Prof. Doutor Tomás António Mendes Oliveira e Silva

professor associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutor João Paulo de Castro Canas Ferreira

professor auxiliar da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor António Guilherme Rocha Campos

professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira

professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores, Prof. Doutor António Campos e Prof. Doutor Arnaldo Oliveira por me terem proporcionado a realização desta dissertação de mestrado e aberto o mundo das áreas de sistemas reconfiguráveis e processamento digital de sinal, contribuindo deste modo para o enriquecimento dos meus conhecimentos nesta área de engenharia e por toda a sua disponibilidade e acompanhamento ao longo de todo o projecto. Também gostaria de lhes agradecer a confiança em mim depositada, quando me incentivaram para a elaboração de um *paper* de investigação para o SAAHPC'11 (*Symposium on Application Accelerators in High-Performance Computing 2011*), acreditando sempre na qualidade do trabalho desenvolvido, e pelo seu esforço e paciência manifestados na explicação e resolução dos problemas que surgiram no desenvolvimento da dissertação.

Em segundo lugar, gostaria de agradecer a toda a minha família, especialmente aos meus avós, aos meus pais e à minha irmã, sem os quais não poderia estar hoje aqui a desenvolver esta dissertação de mestrado. Souberam sempre motivar-me e fazer-me acreditar no meu potencial, enquanto engenheiro e no potencial da dissertação. Além disso, foram incansáveis em dar-me sempre uma palavra de alento durante os momentos mais difíceis.

Por último, gostaria de agradecer a todos os meus amigos, em especial ao Flávio Jorge e ao Diogo Almeida, que souberam sempre apoiar-me e oferecer-me uma gargalhada nos momentos mais difíceis e à Mafalda Castelo-Branco cujo sorriso me deu sempre a força necessária para desenvolver este projecto de forma tão profissional quanto possível.

palavras-chave

Modelação acústica, Resposta impulsional, Guias-de-onda digitais, Paralelização de aplicações, Hardware para aplicação específica, VHDL, FPGA.

resumo

O Meshotron tem como objectivo a paralelização em larga escala de modelos acústicos baseados na técnica de modelação *Digital Waveguide Mesh-3D*. Para tal desenvolveram-se unidades especializadas (*ASH*). Dependendo da topologia adoptada para o modelo acústico a complexidade da nossa unidade vai variar. Neste caso, optou-se por uma topologia rectangular.

De forma geral a dissertação pode dividir-se em quatro secções: uma secção introdutória, que explica em que consiste o princípio de modelação acústica, quais as técnicas de modelação existentes, em que consiste e qual a contribuição do Meshotron para esta temática; uma secção que explica a arquitectura do Meshotron; uma secção relativa ao fluxo de projecto, onde são descritos as simulações e testes efectuados que contribuíram para a validação do Meshotron e uma secção final de análise dos resultados finais e trabalho futuro.

Na secção da arquitectura do Meshotron faz-se alusão aos seus principais elementos, como os bancos de memória e as *FIFOs*; a forma como estes estão interligados e como é efectuado o seu controlo, de modo a respeitar as etapas de funcionamento do Meshotron.

A secção relativa ao fluxo de projecto tem como objectivo validar o funcionamento do Meshotron. A validação passa pela comparação das respostas impulsionalis (*RIRs* – *room impulse responses*) de cada modelo obtidas pelas descrições em *VHDL*, com as obtidas pelo programa de modelação *DWM-3D* criado em *Matlab* em condições similares de funcionamento. Esta secção divide-se em duas fases: uma primeira fase de simulação e uma segunda fase de depuração e implementação.

Durante a fase de simulação foi feito o estudo da influência do número de *bits* do barramento de dados na precisão numérica, a fim de determinar o número óptimo de *bits* a utilizar, que minimizasse o erro relativo associado às operações de cálculo. Com este parâmetro definido procedeu-se à simulação comportamental de modelos com 1, 2, 4 e 8 unidades e à extracção das *RIRs* de cada modelo.

A fase de depuração e implementação implicou um estudo dos recursos consumidos e da frequência máxima de funcionamento do Meshotron, de modo a optimizá-lo em termos de desempenho. Após este estudo, procedeu-se à implementação de modelos com 1 e 2 unidades na *FPGA* seleccionada. Nesta fase foi necessário especificar alguns parâmetros, como, por exemplo, o sinal de relógio aplicado a cada unidade de cada modelo e o sinal de *trigger*. Após esta especificação, utilizou-se a ferramenta de depuração - *Chipscope* para extrair as *RIRs* de cada um dos modelos descritos em *VHDL*.

Os resultados obtidos por simulação comportamental dos modelos descritos em *VHDL* estiveram de acordo com os obtidos pelo programa de modelação *DWM-3D*. Em termos de implementação e depuração, quando sujeitas às mesmas condições de precisão numérica (inteiros de 32 *bits*) os resultados eram iguais. No entanto, quando a precisão numérica era distinta (inteiro de 32 *bits* e *double*), existia uma diferença entre a *RIR* obtida pelo *Chipscope* e pelo programa de modelação *DWM-3D*, diferença essa que não era significativa, devido à sua ordem de grandeza. Logo concluir-se-á que o funcionamento do Meshotron foi validado.

keywords

Acoustic modelling, Impulse response, Digital waveguides, Parallelisation of applications, Application-specific hardware, VHDL, FPGA.

abstract

The main aim of the Meshotron is a large scale parallelisation of three-dimensional (3D) digital waveguide-mesh (*DWM*) room acoustic models. To perform this task application-specific-hardware (*ASH*) were developed. The complexity of the Meshotron depends on the topology adopted. The rectangular mesh topology was elected due to its simplicity.

This dissertation can be divided into four sections: a first section that describes what acoustic modelling is, its main techniques and what the contribution of the Meshotron to this subject is; a second section that explains the architecture of the Meshotron; a third section of the flow of the project, where the different tests and simulations performed are presented to validate the behaviour of the Meshotron and finally a fourth section that summarizes the main conclusions.

In the architecture section some of the elements that compose the Meshotron are described. For example: memory banks and *FIFOs*, how these elements are connected and how its control is performed according to each operating stage of the Meshotron.

The section that describes the flow of the project is aimed at validating the operation of the Meshotron. The validation is based on the comparison of the room impulse responses (*RIRs*) of each model obtained from the *VHDL*'s descriptions and the *3D-DWM* modeling program created in *Matlab* in the same operating conditions. This section is divided into two phases: a first phase of simulation and a second phase of debugging and implementation.

During the simulation phase a study was carried out to find the optimum number of *bits* to be used in the data path that would minimize the error. After defining this parameter, the behaviour of a set of models with 1, 2, 4 and 8 units was simulated. The room impulse response of each model created by the *VHDL*'s descriptions was extracted.

The implementation and debugging phase implied a study of the Meshotron in terms of resources and maximum operating frequency to optimize its performance. This study led to the implementation of a set of models in the selected *FPGA*. Two models, with 1 and 2 units were tested. In this phase it was necessary to specify some signals, such as the clock signal and the trigger signal. After this specification the signal analysis tool (*Chipscope*) was used to extract the room impulse responses of each model created in *VHDL*.

In terms of simulation the results proved to be exactly identical to those obtained using *3D-DWM* modelling program for the same models and operating conditions. In terms of implementation and debugging the room impulse responses that were obtained by the *Chipscope* were identical to the room impulse responses obtained to the same models by the *3D-DWM* modelling program under the same numeric accuracy (*32-bits* integer). However, when the numeric accuracy is different (*double* and *32-bits* integer), the difference between the room impulse responses from the same model was negligible. So we can conclude that the design was validated.

Conteúdo

Capítulo 1	Introdução	1
1.1.	Enquadramento.....	1
1.2.	Motivação e objectivos.....	1
1.3.	Organização da dissertação	2
Capítulo 2	Modelação acústica.....	3
2.1.	Técnicas de modelação acústica.....	3
2.2.	<i>Digital Waveguide Mesh-3D</i>	5
2.3.	Peso computacional dos modelos acústicos	7
2.4.	Paralelização de modelos acústicos <i>DWM-3D</i> : Meshotron.....	8
Capítulo 3	Arquitectura da unidade-base do Meshotron	13
3.1.	Introdução.....	13
3.2.	Abordagem inicial	13
3.2.1.	<i>Data path</i>	13
3.2.1.1.	Estrutura geral	13
3.2.1.2.	Bancos de memória.....	14
3.2.1.3.	<i>Buffers</i> de comunicação	15
3.2.1.4.	Módulo de <i>scattering</i>	15
3.2.1.5.	Formato dos dados	17
3.2.1.6.	Unidade de <i>scattering</i>	17
3.2.1.7.	Barramento de dados.....	20
3.2.2.	Bloco de controlo	21
3.2.2.1.	Sequência geral de computação.....	21
3.2.2.2.	Endereçamento.....	22
3.2.3.	<i>Drivers</i> de endereçamento.....	26
3.2.4.	Sinais de controlo.....	27
3.2.5.	Tempo de computação.....	29
3.3.	Segunda abordagem.....	30
3.3.1.	Módulo de <i>scattering</i>	30
3.3.2.	Endereçamento do <i>scattering pass</i>	31
3.3.3.	Sinais de controlo.....	32
3.3.4.	Tempo de computação.....	32

3.4.	Abordagem final.....	33
3.4.1.	<i>Data path</i>	34
3.4.1.1.	Estrutura geral	34
3.4.1.2.	Barramento de dados.....	36
3.4.2.	Bloco de controlo	37
3.4.2.1.	Sequência de controlo geral	37
3.4.2.2.	Endereçamento do <i>scattering delay pass 12</i> – Etapa <i>SD12</i>	38
3.4.3.	<i>Drivers</i> de endereçamento.....	40
3.4.4.	Sinais de controlo.....	41
3.4.5.	Tempo de computação.....	43
Capítulo 4	Modelação, simulação, síntese e depuração.....	45
4.1.	Introdução.....	45
4.2.	Modelação e simulação.....	45
4.2.1.	Estudo do impacto do número de <i>bits</i> na precisão numérica.....	47
4.2.2.	Modelo com uma unidade do Meshotron.....	48
4.2.2.1.	Resposta impulsional	49
4.2.2.2.	Tempo máximo de cálculo da unidade de <i>scattering</i>	49
4.2.3.	Modelo com duas unidades do Meshotron.....	50
4.2.4.	Modelo acústico com quatro unidades do Meshotron.....	51
4.2.5.	Modelo com oito unidades do Meshotron.....	51
4.2.6.	Análise dos resultados obtidos.....	53
4.3.	Síntese.....	53
4.3.1.	Recursos e frequência máxima de funcionamento da unidade de <i>scattering</i> ...	53
4.3.2.	Variação da frequência máxima de funcionamento em função do número de unidades de <i>scattering</i>	54
4.3.3.	Caminho crítico.....	54
4.3.4.	Recursos e frequência máxima de funcionamento da unidade do Meshotron	57
4.4.	Implementação e depuração.....	58
4.4.1.	Configurações iniciais.....	60
4.4.2.	Modelo com uma unidade do Meshotron.....	62
4.4.3.	Modelo com duas unidades do Meshotron.....	64
4.4.3.1.	Teste 1 - Variação do sinal de relógio	64
4.4.3.2.	Teste 2 - Variação da coordenada do nó emissor	69

4.4.4.	Análise dos resultados obtidos.....	71
Capítulo 5	Conclusão e trabalho futuro	73
5.1.	Análise final dos resultados e conclusões.....	73
5.2.	Trabalho futuro	73

Lista de Figuras

Fig. 1 – Obtenção da <i>RIR</i> de uma sala.	3
Fig. 2 – Resposta acústica de uma sala; o símbolo ‘*’ representa ‘convolução’.....	3
Fig. 3 – Obtenção de <i>RIR</i> por <i>Ray-tracing</i> : E – Emissor; R – Receptor; r – Raio da zona de recepção. [26].....	4
Fig. 4 – Algumas topologias de <i>Digital Waveguide Mesh</i> : (a) 2-D rectangular; (b) 2-D triangular; (c) 3-D rectangular.	4
Fig. 5 – Matriz de nós (<i>Digital Waveguide Mesh-2D</i>).	5
Fig. 6 - Exemplo de um bloco cúbico <i>DWM-3D</i> com $N=5$ e uma distância entre nós de d	6
Fig. 7 – Diagrama de blocos do algoritmo de modelação <i>DWM-3D</i> [16].....	7
Fig. 8 – Estrutura de um nó genérico na topologia rectangular 3-D: (a) Modelo <i>DWM-3D</i> em topologia rectangular; (b) Nó genérico [16].	9
Fig. 9 – Comunicação do bloco rectangular de um modelo <i>DWM-3D</i> com os blocos adjacentes.	10
Fig. 10 – Sequência e articulação das etapas do <i>delay pass</i>	11
Fig. 11 – Encaminhamento de dados (<i>data path</i>) numa unidade do Meshotron na abordagem inicial.	14
Fig. 12 – Interface externa de um banco de memória.	14
Fig. 13 – Interface externa de um <i>buffer</i> de comunicação.....	15
Fig. 14 – Interface externa do módulo de <i>scattering</i>	16
Fig. 15 – Estrutura interna do módulo de <i>scattering</i>	16
Fig. 16 – Abordagem inicial para a gestão da etapa de <i>scattering</i>	17
Fig. 17 – Estrutura de uma unidade de <i>scattering</i> para <i>nós de ar</i>	18
Fig. 18 - Estrutura de uma unidade de <i>scattering</i> para <i>nós fronteira</i>	18
Fig. 19 – Salvaguarda da situação de <i>overflow</i> por extensão de 3 <i>bits</i>	19
Fig. 20 – Estrutura da unidade de <i>scattering</i> para <i>nós de ar</i> com detecção de <i>overflow</i>	19
Fig. 21 – Barramento de dados para o sentido positivo do eixo x . As setas representam o fluxo de dados em cada etapa (S , $D1$, $D2$ e $D3$).	20
Fig. 22 – Diagrama de estados do bloco de controlo da unidade do Meshotron.	21
Fig. 23 – Obtenção das condições <i>albe</i> (<i>all local buffers empty</i>) e <i>aabf</i> (<i>all adjacent buffers full</i>) a partir dos semáforos dos <i>buffers</i>	22
Fig. 24 – Gerador de endereços da etapa <i>INI</i>	23

Fig. 25 – Gerador de endereços da etapa <i>S</i> na abordagem inicial.	24
Fig. 26 – Gerador de endereços das etapas <i>D1</i> e <i>D3</i>	25
Fig. 27 – Gerador de endereços da etapa <i>D2</i>	26
Fig. 28 – Sinais de <i>clock enable</i> para as operações de endereçamento, escrita e leitura.....	28
Fig. 29 – Segunda abordagem para a gestão da etapa de <i>scattering</i>	30
Fig. 30 – Gerador de endereços da etapa <i>S</i> (segunda abordagem).....	31
Fig. 31 – Transferências efectuadas nas etapas <i>S</i> e <i>D2</i> da segunda abordagem.	33
Fig. 32 – Transferências de informação na da etapa <i>S</i> e <i>D2</i> efectuadas na etapa <i>SD12</i> da abordagem final.....	34
Fig. 33 - Estrutura de dados (<i>data path</i>) de uma unidade do Meshotron na abordagem final para a direcção positiva do eixo do <i>x</i>	35
Fig. 34 - Encaminhamento de dados (<i>data path</i>) numa unidade do Meshotron na abordagem final.....	35
Fig. 35 – Barramento de dados para a direcção positiva do eixo do <i>x</i> na abordagem final. As setas representam o fluxo de dados em cada etapa (<i>SD12</i> e <i>D3</i>).	36
Fig. 36 – Máquina de estados de definição dos bancos de memória de entrada e saída.	37
Fig. 37 - Diagrama de estados do bloco de controlo da abordagem final de uma unidade do Meshotron.....	38
Fig. 38 – Gerador de endereços da etapa <i>SD12</i>	39
Fig. 39 – Alternativa ao conjunto de somadores e substractores.	40
Fig. 40 – Procedimento de simulação.	46
Fig. 41 – Erro existente entre a <i>RIR</i> em formato <i>double</i> e em formato inteiro de 32 e 16 <i>bits</i> ...47	
Fig. 42 - Evolução do sinal de erro e da <i>RIR</i> com o número de amostras para grandezas de 32 <i>bits</i>	48
Fig. 43 - Evolução do sinal de erro e da <i>RIR</i> com o número de amostras para grandezas de 16 <i>bits</i>	48
Fig. 44 – Modelo com uma unidade do Meshotron.	49
Fig. 45 – Modelo com duas unidades do Meshotron.	50
Fig. 46 – Modelo com quatro unidades do Meshotron.....	51
Fig. 47 – Modelo com oito unidades do Meshotron.	52
Fig. 48 – Comparação entre o modelo criado em <i>VHDL</i> com o modelo criado em <i>Matlab</i> . ..	52
Fig. 49 - Gráfico de variação do atraso total do principal caminho crítico em função do número de unidades de <i>scattering</i>	55

Fig. 50 – Principal caminho crítico da unidade do Meshotron.	56
Fig. 51 – Circuito combinatório do detector de <i>overflow</i> com as entradas registadas.	57
Fig. 52 – Procedimento de implementação.	59
Fig. 53 – Interface gráfico de configuração do número de amostras a recolher pelo <i>Chipscope</i> no modelo com uma unidade do Meshotron.	61
Fig. 54 – Interface gráfico de configuração do número de amostras a recolher pelo <i>Chipscope</i> no modelo com duas unidades do Meshotron.	61
Fig. 55 – Diagrama de blocos do programa de validação do modelo com uma unidade do Meshotron.	62
Fig. 56 – <i>RIR</i> obtida pelo <i>Chipscope</i> , pelo <i>Matlab</i> e o respectivo sinal de erro na coordenada (0,0,0).	63
Fig. 57 – <i>RIR</i> obtida pelo <i>Chipscope</i> , pelo <i>Matlab</i> e respectivo sinal de erro para a coordenada (1,2,3).	63
Fig. 58 – <i>RIR</i> obtida pelo <i>Chipscope</i> , pelo <i>Matlab</i> e respectivo sinal de erro para a coordenada (5,1,0).	64
Fig. 59 – Diagrama de blocos do programa de validação do modelo com duas unidades do Meshotron.	65
Fig. 60 – <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 33 MHz.	66
Fig. 61 - <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 27 MHz.	66
Fig. 62 - <i>RIR</i> obtida no <i>Chipscope</i> para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 27 MHz e de 33 MHz.	67
Fig. 63 - <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 33 MHz.	67
Fig. 64 - <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 27 MHz.	68
Fig. 65 - <i>RIR</i> obtida no <i>Chipscope</i> para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 27 MHz e de 33 MHz.	68
Fig. 66 - <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas com o nó emissor na coordenada (1,2,3) e o nó receptor na coordenada (0,0,0).	69
Fig. 67 – <i>RIR</i> obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas com o nó emissor na coordenada (3,1,0) e o nó receptor na coordenada (0,0,0).	70

Fig. 68 - RIR obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas com o nó emissor na coordenada (7,2,1) e o nó receptor na coordenada (0,0,0).	70
Fig. 69 - RIR obtida no <i>Matlab</i> e no <i>Chipscope</i> em diferentes precisões numéricas com o nó emissor na coordenada (1,6,7) e o nó receptor na coordenada (1,4,5).	71
Fig. 70 – Janela de configuração do número de sinais de relógio da <i>FIFO</i>	77
Fig. 71 – Janela de configuração do número de bits dos parâmetros de dados da <i>FIFO</i>	77
Fig. 72 – Janela de configuração dos sinais de controlo da <i>FIFO</i>	78
Fig. 73 – Demonstração da operação de multiplicação.	79
Fig. 74 – Interface gráfico de configuração do bloco de divisão da <i>Xilinx</i>	80
Fig. 75 – Bloco elementar de um divisor.	80
Fig. 76 – Tabela de verdade.	81
Fig. 77 – Blocos elementares em cascata.	81
Fig. 78 – Organização da estrutura hierárquica dos ficheiros de código constituintes da unidade do <i>Meshotron</i>	86
Fig. 79 – Organização da estrutura hierárquica dos ficheiros de código constituintes da <i>driver</i> de endereços.	87
Fig. 80 - Organização da estrutura hierárquica dos ficheiros de código constituintes da unidade de <i>scattering</i>	87

Lista de Tabelas

Tabela I – Esquema do endereçamento da etapa <i>INI</i>	22
Tabela II – Esquema de endereçamento da etapa <i>S</i> (abordagem inicial).....	23
Tabela III – Esquemas de endereçamento das etapas <i>D1</i> e <i>D3</i>	25
Tabela IV – Esquemas de endereçamento da etapa <i>D2</i>	26
Tabela V – Endereçamento de cada banco de memória nas diferentes etapas (abordagem inicial).....	27
Tabela VI – Sinais de controlo de escrita e leitura em cada etapa (abordagem inicial).....	28
Tabela VII – Expressões booleanas dos sinais de controlo (abordagem inicial).....	29
Tabela VIII - Duração de cada etapa na abordagem inicial.....	29
Tabela IX – Esquema de endereçamento da etapa <i>S</i> (segunda abordagem).....	31
Tabela X – Sinais de controlo de escrita e leitura em cada etapa (segunda abordagem).....	32
Tabela XI – Expressões booleanas para os diferentes sinais de controlo na segunda abordagem.....	32
Tabela XII – Duração de cada etapa na segunda abordagem.....	32
Tabela XIII – Esquemas de endereçamento da etapa <i>SD12</i>	38
Tabela XIV – Endereçamento de cada banco de memória em cada etapa na abordagem final.....	41
Tabela XV – Sinais de controlo dos bancos de memória de entrada/saída.....	42
Tabela XVI – Sinais de controlo dos <i>buffers</i> locais.....	42
Tabela XVII – Sinais de controlo dos <i>buffers</i> adjacentes e do módulo de <i>scattering</i>	42
Tabela XVIII – Expressões booleanas dos sinais de controlo da unidade do Meshotron na abordagem final.....	43
Tabela XIX - Duração de cada etapa na abordagem final.....	43
Tabela XX – Relação sinal ruído para grandezas de 16 e 32 <i>bits</i>	48
Tabela XXI – Duração total admissível da operação de <i>scattering</i>	50
Tabela XXII – Recursos consumidos e frequência máxima de funcionamento da unidade de <i>scattering</i>	54
Tabela XXIII – Frequência máxima de funcionamento em função do número de unidades de <i>scattering</i>	54
Tabela XXIV – Frequência máxima de funcionamento da unidade do Meshotron com o número óptimo de unidades de <i>scattering</i> com e sem detecção de <i>overflow</i>	57

Tabela XXV – Recursos consumidos e frequência máxima de funcionamento de uma unidade do Meshotron.	58
Tabela XXVI – Frequência máxima de funcionamento e recursos consumidos.	82
Tabela XXVII – Erro relativo.....	83
Tabela XXVIII – Ficheiros de código presentes no topo da hierarquia.	85
Tabela XXIX – Ficheiros de código presentes na unidade do Meshotron.	85
Tabela XXX – Ficheiros de código presentes no <i>driver</i> de endereçamento (<i>DriverAddress.vhd</i>).	85
Tabela XXXI – Ficheiros de código presentes no módulo de <i>scattering</i> (<i>Scattering Module.vhd</i>).	86
Tabela XXXII – Ficheiros de código não utilizados.	86
Tabela XXXIII – Lista de programas criados em <i>Matlab</i>	89

Capítulo 1 Introdução

1.1. Enquadramento

A presente dissertação de mestrado insere-se no projecto Meshotron, que visa desenvolver unidades especializadas (*application-specific hardware -ASH*) capazes de formar uma rede dedicada de computação paralela para modelação acústica por malhas de guias de onda digitais (*Digital Waveguide Meshes - DWM*) tridimensionais (3-D) [15].

A estratégia de paralelização consiste no particionamento de dados - os modelos *DWM-3D* são decompostos em blocos cúbicos. Sendo o *overhead* de comunicação entre blocos independente do número de blocos em utilização [15], é, em teoria, possível conseguir uma paralelização em larga escala e, conseqüentemente, uma enorme diminuição do tempo de processamento, abrindo a porta à aplicação prática de modelos físicos para simulação acústica de salas.

1.2. Motivação e objectivos

A resposta impulsional é um elemento fundamental para caracterizar um espaço acústico. Para a sua obtenção existe um conjunto de técnicas de modelação acústica que podem ser utilizadas, cada uma com as suas limitações em termos de aplicabilidade, o que torna esta temática algo ainda em desenvolvimento. Alguns dos métodos de modelação, como por exemplo o *Ray-tracing* e o *Image-Source*, são utilizados para variadas aplicações, mas não possuem o rigor necessário. A solução para este problema passa pela utilização de modelos físicos, uma vez que estes produzem soluções mais rigorosas. No entanto estes modelos apresentam uma grave limitação: serem computacionalmente muito exigentes. Para eliminar esta limitação é imperativa a paralelização dos modelos físicos.

Digital Waveguide Mesh é um dos métodos de modelação física cuja paralelização por particionamento de dados já foi experimentada e oferece uma escalabilidade ilimitada [15]. Por outro lado como o algoritmo desta técnica de modelação é extremamente simples [6] torna-se possível desenvolver *ASH* (*application-specific hardware*) para efectuar a paralelização do modelo.

O Meshotron consiste numa rede de unidades *ASH* (*application-specific hardware*) para *Digital Waveguide Mesh-3D*.

Neste contexto, os objectivos definidos para esta dissertação foram:

- Desenvolver um protótipo virtual, realizado com ferramentas de simulação de *hardware* (nomeadamente linguagem de descrição *VHDL*), de uma unidade completa do Meshotron para a topologia de *DWM* rectangular;
- Avaliar o comportamento global do sistema, simular e escolher alternativas de implementação e estabelecer uma plataforma de integração e teste dos blocos de *hardware* a desenvolver.
- Desenvolver um bloco de *scattering* em *FPGA*; investigar as possibilidades de implementação combinacional e sequencial.

1.3. Organização da dissertação

Esta dissertação desenvolve-se em 4 capítulos. No capítulo que se segue (Capítulo 2 – Modelação acústica) descreve-se em que consiste modelação acústica; quais as principais técnicas de modelação existentes (entre estas destaca-se a *Digital Waveguide Mesh-3D*) e as suas principais limitações. Ainda neste capítulo se explica o princípio de funcionamento do Meshotron e que vantagens advêm da sua utilização.

No capítulo 3 (Projecto da unidade-base do Meshotron) explica-se a arquitectura da unidade do Meshotron e os seus principais elementos (bancos de memória, *buffers* de interface, unidade de controlo, ...).

No capítulo 4 descrevem-se o conjunto de simulações e testes de síntese e implementação que conduziram à validação e concretização da unidade do Meshotron. Entre as simulações e testes executados há que destacar a:

- Determinação de qual o número óptimo de *bits* do barramento de dados, de modo a minimizar o erro relativo existente nas operações de cálculo da unidade de *scattering*;
- Simulação comportamental de pequenos modelos com 1, 2, 4 e 8 unidades do Meshotron (análise comportamental);
- Determinação do número óptimo de unidades de *scattering* (*d*) a utilizar;
- Análise em termos de frequência máxima de funcionamento e recursos consumidos da unidade do Meshotron construída.
- Implementação da unidade do Meshotron na *FPGA* e extracção da *RIR* de um conjunto de modelos criados;

Finalmente, no capítulo 5 resumem-se as principais conclusões e apresentam-se as principais tarefas a efectuar em trabalho futuro com vista a desenvolver e optimizar o funcionamento do Meshotron.

Capítulo 2 Modelação acústica

2.1. Técnicas de modelação acústica

O objectivo da modelação acústica é prever o comportamento acústico de um espaço fechado, dados os pontos de emissão e recepção de som [18]. Na medida em que os espaços podem ser considerados, do ponto de vista acústico, como sistemas lineares e invariantes no tempo (*LTI*), esse comportamento é completamente descrito pela sua resposta impulsional [23] (*room impulse response* - *RIR*) – vide Fig. 1 e Fig. 2.

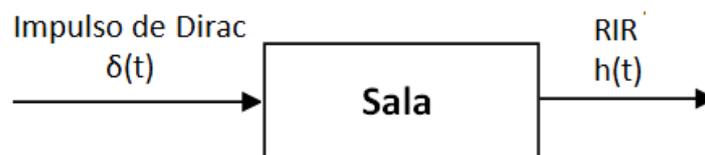


Fig. 1 – Obtenção da *RIR* de uma sala.

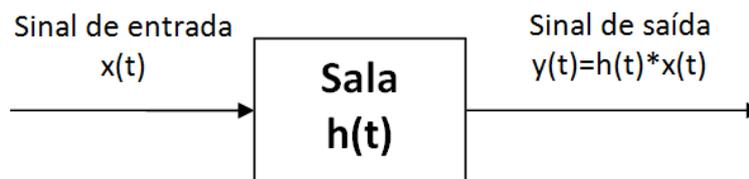


Fig. 2 – Resposta acústica de uma sala; o símbolo “*” representa ‘convolução’.

Os modelos acústicos podem assim ser vistos como métodos de calcular a *RIR*. A modelação acústica divide-se em duas categorias: digital e analógica. Em termos de modelação acústica digital, que consiste na base desta dissertação, existem essencialmente dois conjuntos de técnicas:

- técnicas *ray-based* [14]: estas consistem essencialmente nos métodos de *Ray-tracing* [26], *Image-source* [26] e suas variantes. As ondas sonoras são vistas como raios cuja propagação obedece às leis da Óptica geométrica, considerando-se apenas fenómenos de reflexão e absorção no impacto com as superfícies que delimitam o espaço. Estas técnicas, apesar de relativamente pouco exigentes do ponto de vista computacional, são pouco rigorosas e extremamente simplificadas (condicionadas a altas frequências). Para comprimentos de onda elevados em comparação com as dimensões das superfícies de reflexão, fenómenos como a difracção e a interferência têm que ser tidos em conta, o que torna estes modelos desadequados. A Fig. 3 ilustra a obtenção de uma *RIR* pelo método *Ray-tracing*;

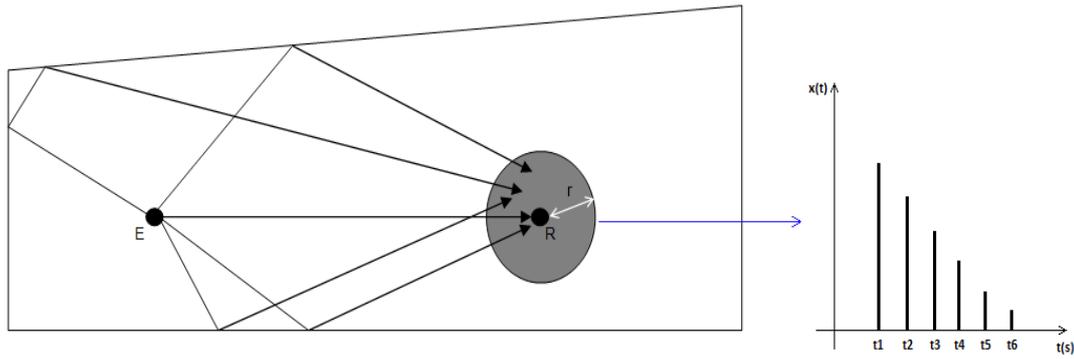


Fig. 3 – Obtenção de RIR por *Ray-tracing*.: **E** – Emissor; **R** – Receptor; r – Raio da zona de recepção. [26]

- técnicas *wave-based* [14]: são métodos de modelação física, na medida em que todos os fenómenos de propagação (difusão, reflexão, absorção, interferência, difracção e refracção) são considerados. Estas técnicas, apesar de mais rigorosas, são computacionalmente muito mais exigentes. Exemplos deste tipo de técnicas são o *FEM* [24] (*Finite Element Method*), o *BEM* [25] (*Boundary Element Method*), o *TLM* [28] (*Transmission Line Modelling*) e o *FDTD* [27] (*Finite Difference Time Domain*).

Este projecto explora um método numérico de diferenças finitas no domínio do tempo (*FDTD – Finite Difference Time Domain*) de resolução da equação de onda de *Helmholtz* (vide Eq. 1 - [1]). Trata-se da modelação por guias-de-onda digitais (*Digital Waveguides – DW*) introduzida por Julius Smith [3].

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} = \frac{1}{c^2} \frac{\partial^2 \psi}{\partial t^2}. \quad \text{Eq. 1}$$

Legenda:

ψ - função da posição e do tempo que descreve o comportamento da onda;

c - velocidade da onda (velocidade do som);

t - instante de tempo;

x - coordenada espacial x ;

y - coordenada espacial y ;

z - coordenada espacial z ;

A Fig. 4 ilustra algumas topologias da técnica *Digital Waveguide Mesh*.

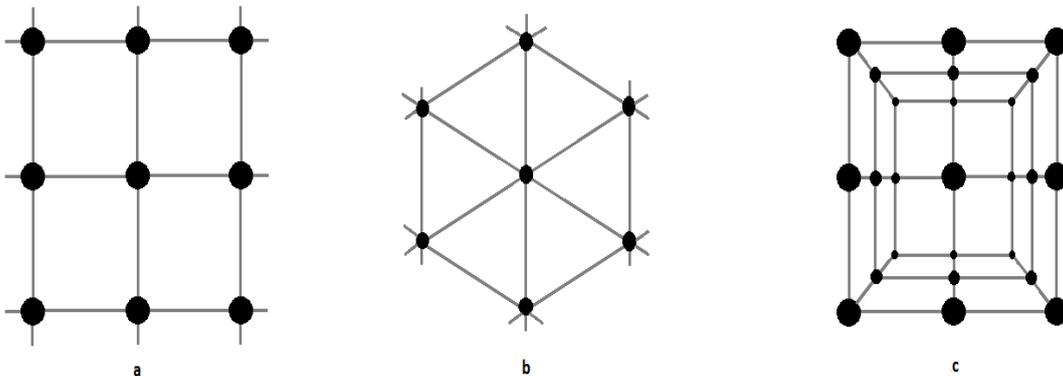


Fig. 4 – Algumas topologias de *Digital Waveguide Mesh*: **(a)** 2-D rectangular; **(b)** 2-D triangular; **(c)** 3-D rectangular.

Inicialmente, este método foi utilizado para modelos a uma dimensão (1-D), nomeadamente para síntese de sons de instrumentos de sopro e cordas, efeitos de *flanging* e reverberação. Estes modelos destacaram-se pelo seu realismo e eficiência computacional [2].

A aplicabilidade do método estende-se a espaços de n dimensões através da construção de malhas (*DWM*) formadas por estruturas regulares de nós, também chamados junções (*scattering junctions*), ligados por guias-de-onda unitárias (*delay units*) bidireccionais [3][12], como ilustra a Fig. 5.

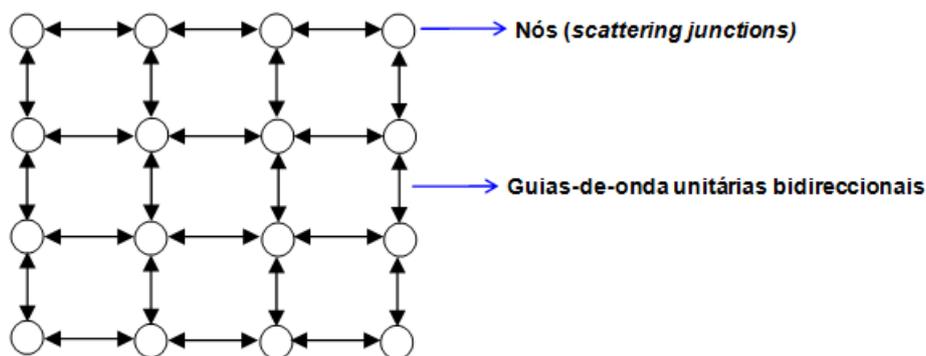


Fig. 5 – Matriz de nós (*Digital Waveguide Mesh-2D*).

Por exemplo, as malhas 2-D podem ser aplicadas à modelação física de tambores, pratos e gongos [4]. A frequência de amostragem, f_s , de qualquer sinal áudio injectado ou guardado no nó para esta técnica de modelação [3] é dada por:

$$f_s = \frac{c\sqrt{2}}{d}. \quad \text{Eq. 2}$$

Note-se que quanto maior a frequência de amostragem maior a densidade da matriz de nós. E como consequência maior o tempo de computação, que consiste numa das principais limitações desta técnica [3].

2.2. *Digital Waveguide Mesh-3D*

O *Digital Waveguide Mesh-3D* consiste numa técnica de modelação de espaços tridimensionais [6] [10]. Esta técnica tornou-se extremamente popular devido:

- ao seu algoritmo simples e intuitivo;
- à sua eficiência (demonstrado para o caso 1D-[2]);
- à flexibilidade dos modelos físicos criados;
- à sua precisão, que pode ser melhorada através do aumento da densidade de amostragem.

Um modelo *DWM* de um espaço acústico consiste numa rede tridimensional de nós interligados entre si por guias-de-onda unitários bidireccionais, onde o sinal de som é discretizado por *nós de ar* e as superfícies de reflexão são representadas por *nós fronteira*. O número de vizinhos que cada nó possui, n , depende da topologia do modelo adoptada [6]. A Fig. 6 ilustra um modelo *DWM* em topologia rectangular, $n=6$.

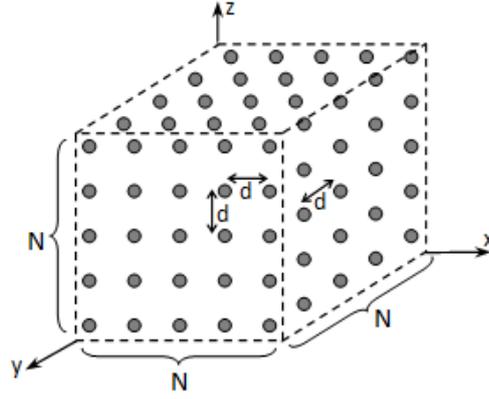


Fig. 6 - Exemplo de um bloco cúbico *DWM-3D* com $N=5$ e uma distância entre nós de d .

O algoritmo de modelação é iterativo [3]; cada ciclo corresponde a um intervalo de tempo T_s , valor determinado pela velocidade de propagação do som num espaço fechado, c (aproximadamente 344 m/s no ar em condições normais de pressão, temperatura e humidade), e pela distância entre os nós vizinhos, d ; o intervalo de tempo permite definir a frequência de amostragem, f_s , de qualquer sinal *audio* injectado ou guardado no nó [5] (vide Eq. 3 - [6]).

$$T_s = \frac{1}{f_s} = \frac{d}{c\sqrt{3}}. \quad \text{Eq. 3}$$

Cada ciclo divide-se duas etapas [3]: *scattering pass* e *delay pass* (vide Fig. 7). Na primeira etapa (*scattering pass* - *S*), o valor da variável da onda de propagação (pressão acústica - p) é calculado para cada nó em função das componentes de pressão acústica provenientes dos seus n vizinhos.

Para os *nós de ar* a pressão acústica é calculada através da equação seguinte [6]:

$$p = \left(\frac{2}{n} \right) \sum_{k=1}^n p_{kin}, \text{ onde } n \text{ é número de vizinhos.} \quad \text{Eq. 4}$$

As componentes de pressão acústica de saída para os n nós vizinhos são obtidas através da equação seguinte [6]:

$$p_{kout} = p - p_{kin}. \quad \text{Eq. 5}$$

Pela observação da Eq. 4 verifica-se que a complexidade da operação de divisão depende da topologia adoptada, isto é, no caso de adoptar-se uma topologia tetraédrica ($n=8$) ou octaédrica ($n=16$) a operação de divisão resume-se a operações de deslocação de *bits* para a direita, desde que as operações sejam efectuadas com números inteiros. No entanto, se for adoptada uma topologia rectangular ($n=6$), a simplicidade da operação de divisão é perdida.

Para os *nós fronteira*, caso seja adoptada uma terminação unidimensional, sendo R o coeficiente de reflexão do material constituinte da fronteira ($0 \leq R \leq 1$) [6],

$$p_{kout} = R p_{kin}. \quad \text{Eq. 6}$$

A segunda etapa (*delay pass* - *D*) baseia-se na transferência de informação entre nós adjacentes: as componentes de pressão acústica de saída de cada nó são transferidas para portos de entrada dos nós adjacentes. Isto simula uma iteração do algoritmo de propagação da onda de som. Uma vez completo o *delay pass*, inicia-se de imediato uma nova iteração.

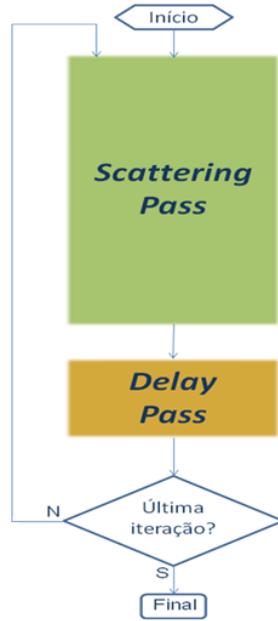


Fig. 7 – Diagrama de blocos do algoritmo de modelação DWM-3D [16].

2.3. Peso computacional dos modelos acústicos

A principal limitação desta técnica consiste na dimensão computacional do problema. Seja t_{m1} o tempo de computação necessário por nó para completar uma iteração (etapas de *scattering* e *delay*), k o número total de iterações e N o número de nós existentes numa aresta do modelo cúbico em causa; o tempo necessário para calcular a *RIR* é dado por:

$$T_{RIR} = kN^3 t_{m1} . \quad \text{Eq. 7}$$

A relação entre o volume do espaço tridimensional, V , e a distância entre nós vizinhos, d , é dada por [6]:

$$v = N^3 d^3 . \quad \text{Eq. 8}$$

Como consequência o número de nós presente no modelo é dado por:

$$N^3 = \frac{V}{d^3} . \quad \text{Eq. 9}$$

Pelo recurso da relação entre a distância entre nós vizinhos, d , e a frequência de amostragem, f_s , efectua-se uma substituição na Eq. 9 e obtém-se:

$$N^3 = v \left(\frac{f_s}{c\sqrt{3}} \right)^3 . \quad \text{Eq. 10}$$

O número de iterações do algoritmo depende do período de amostragem, T_s , e do tempo de reverberação, RT_{60} , como está representado na Eq. 11 [6] :

$$k = \frac{RT_{60}}{T_s} = RT_{60} f_s . \quad \text{Eq. 11}$$

Por substituição das relações estabelecidas nas Eq. 8 e Eq. 11 obtém-se o tempo necessário para calcular a *RIR* [6]:

$$T_{RIR} = \frac{1}{(c\sqrt{3})^3} V RT_{60} f_s^4 t_{m1}. \quad \text{Eq. 12}$$

Para ilustrar a dimensão do problema, considere-se um *hall* de um concerto com $V=10000 \text{ m}^3$ e $RT_{60}=1.5 \text{ s}$ – relativamente pequeno e ‘seco’ acusticamente [7]. Seja $f_s = 44,1 \text{ KHz}$, a frequência de amostragem *standard* em áudio – valores significativamente mais elevados poderão ser necessários, de modo a evitar artefactos causados pelo erro de dispersão, a principal limitação do método [8]. Considerando $t_{m1}=50 \text{ ns}$ – valor relativamente optimista, considerando estudos anteriores de desempenho em computadores com um processador [8] – o tempo de computação seria de 155 dias! Torna-se claro por este exemplo que a aplicabilidade do método em utilização está ainda limitado a espaços pequenos e/ou baixas frequências. Para alargar a aplicabilidade do método, t_{m1} tem de diminuir drasticamente, logo a paralelização é imperativa [6] [8].

Como consequência criou-se uma unidade dedicada baseada no princípio de paralelização da técnica *DWM-3D* capaz de efectuar a descrição acústica de um espaço – O Meshotron.

2.4. Paralelização de modelos acústicos *DWM-3D*: Meshotron

O Meshotron consiste numa rede de unidades de *hardware* de aplicação específica (ASH), construídas com o objectivo de paralelizar em larga escala modelos tridimensionais *Digital Waveguide Mesh (DWM)* para a simulação acústica de espaços fechados. O conceito e a arquitectura da unidade do Meshotron para uma topologia rectangular *DWM-3D* estão descritas de forma geral em [15]. A unidade do Meshotron presta-se bem à abordagem clássica de separação do barramento de dados e do bloco de controlo [17], devido à regularidade e simplicidade do algoritmo em termos de sequência de cálculos e acessos à memória.

A complexidade do Meshotron depende do tipo de topologia escolhida para o modelo *DWM-3D* (rectangular [3][5], tetraédrica [10] e octaédrica [13]). Optou-se por utilizar uma topologia rectangular, de modo a minimizar a complexidade do endereçamento aos bancos de memória no *delay pass*. Contudo, existe consciência de que esta topologia não permite reduzir a divisão da unidade de *scattering* (vide Apêndice B) a uma simples deslocação de *bits*. Esta topologia implica a existência de seis pares de registos por nó, para efectuar as transferências de informação com os seus vizinhos nos seis sentidos espaciais (vide Fig. 8).

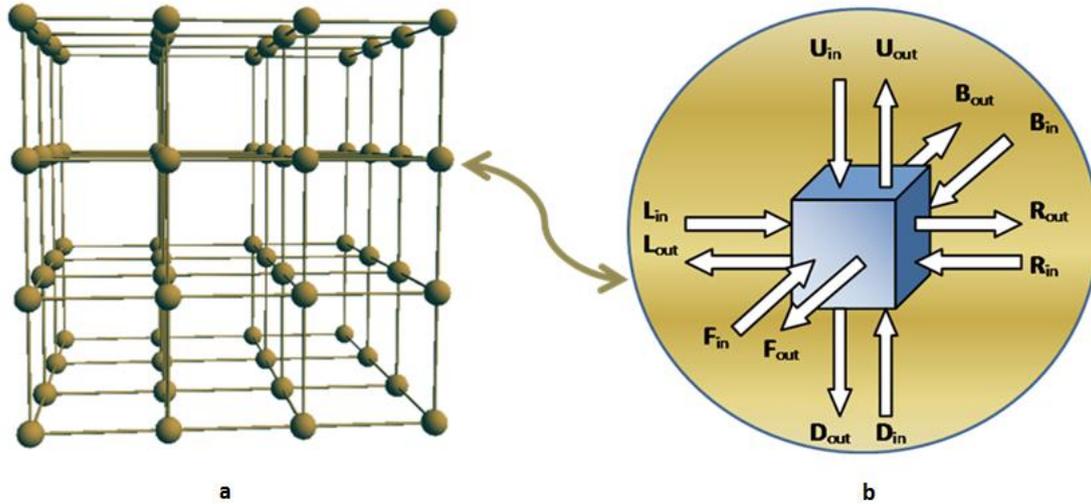


Fig. 8 – Estrutura de um nó genérico na topologia rectangular 3-D: **(a)** Modelo DWM-3D em topologia rectangular; **(b)** Nó genérico [16].

A unidade do Meshotron para uma topologia rectangular opera num *array* com N^3 nós discretizados segundo uma partição cúbica de um espaço fechado, como se encontra ilustrado na Fig. 6.

A topologia adoptada (topologia rectangular) para o modelo acústico *DWM-3D* implicou uma substituição do parâmetro n por 6 na Eq. 4 de cálculo da pressão acústica, p , descrita para o *scattering pass* (vide secção 2.2).

$$p = \frac{1}{3} \sum_{k=1}^6 p_{kin} = \frac{L_i + R_i + B_i + F_i + D_i + U_i}{3}, \text{ para } n=6 \text{ (topologia rectangular)}. \quad \text{Eq. 13}$$

As componentes de pressão de saída L_o , R_o , B_o , F_o , D_o e U_o são calculadas a partir das componentes de pressão de entrada L_i , R_i , B_i , F_i , D_i e U_i de acordo com a Eq. 5.

Note-se que, no *scattering pass*, não é necessária a existência de uma comunicação inter-blocos, uma vez que os cálculos efectuados são internos a cada nó.

As transferências de informação do *delay pass* efectuam-se da forma descrita na secção 2.2. Uma vez que cada bloco é constituído por N^3 nós, existem $6N^3$ transferências por iteração sendo que $6N^2$ correspondem às ligações com blocos adjacentes (N^2 por cada uma das seis faces do bloco), exigindo, portanto, comunicação inter-unidades.

Para lidar com este requisito de forma eficiente, a unidade foi equipada com interfaces de comunicação independentes para os seis blocos adjacentes (uma por cada sentido espacial: Direita, Esquerda, Frente, Atrás, Cima, Baixo), associados aos *buffers* de interface de N^2 elementos que armazenam a informação de saída (vide Fig. 9).

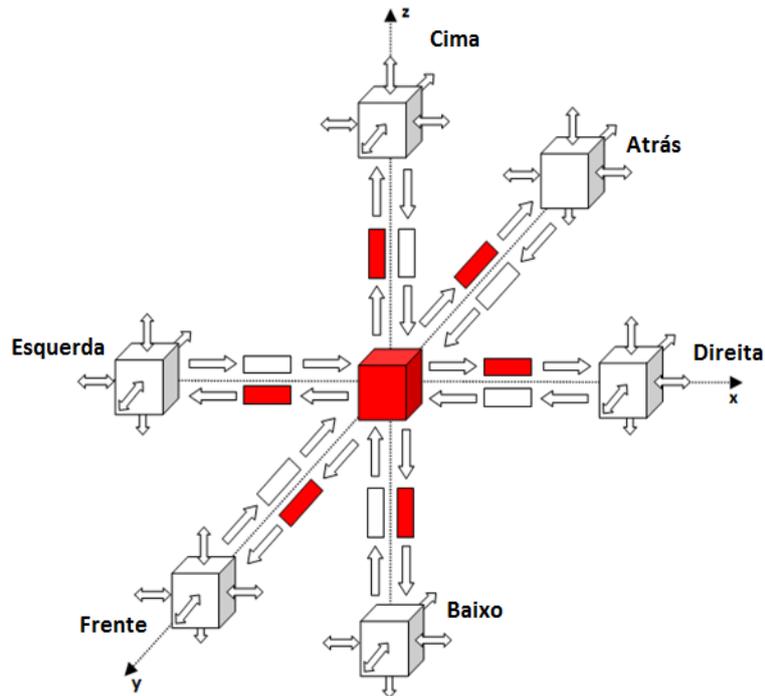


Fig. 9 – Comunicação do bloco rectangular de um modelo *DWM-3D* com os blocos adjacentes.

Legenda:

-  - Buffers de interface de saída do bloco central;
-  - Buffers de interface de saída dos blocos adjacentes;
-  - Bloco central.

O *delay pass* dividiu-se em três etapas (vide Fig. 10):

- *D1*: os seis conjuntos de N^2 valores destinados aos blocos adjacentes são transferidos dos registos de saída dos nós de superfície do bloco para os respectivos *buffers* de interface;
- *D2*: a informação de saída é copiada dos restantes nós directamente para os registos de entrada dos nós adjacentes, que pertencem ao mesmo bloco;
- *D3*: os seis conjuntos de N^2 valores, provenientes de blocos adjacentes necessários para completar a operação de transferência de informação, são transferidos dos respectivos *buffers* de interface pertencentes às unidades adjacentes para os registos de entrada dos nós de superfície.

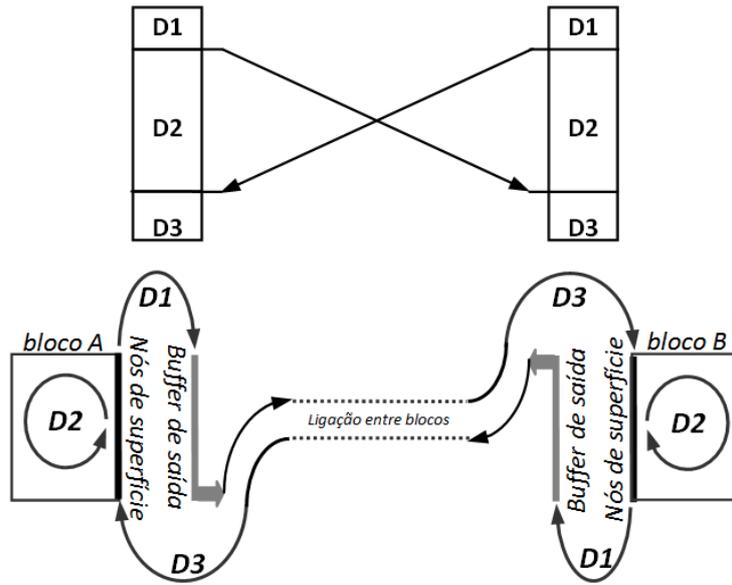


Fig. 10 – Sequência e articulação das etapas do *delay pass*.

Como cada nó tem seis registos de saída, o número total de registos por iteração é $6N^3$. Destes registos, $6N^2$ destinam-se a unidades adjacentes, através dos *buffers* de comunicação. O *overhead* de comunicação pode desta forma ser estimado como:

$$\frac{6N^2}{6N^3} = \frac{1}{N} \text{ - inversamente proporcional ao tamanho do bloco.} \quad \text{Eq. 14}$$

Por exemplo, para $N=64$ o *overhead* de comunicação é inferior a 2%. Uma vez que os interfaces de comunicação operam de forma independente, as transferências de informação podem ser iniciadas imediatamente após a fase *D1* e correr simultaneamente com a fase *D2*. As transferências de informação entre unidades adjacentes, que ocorrem em *D1* e *D3*, requerem um mecanismo de controlo que sinalize se a informação chegou correctamente ao seu destino. Este mecanismo implica a utilização de estados de espera (*wait states*). As condições de entrada/saída destas etapas encontram-se explicadas na secção 3.2.2.1.

No capítulo seguinte, descreve-se a arquitectura da unidade do Meshotron projectada neste trabalho e todos os detalhes de implementação no sentido de otimizar o seu desempenho. Em termos de sincronismo, o Meshotron pode ser classificado como um sistema globalmente assíncrono, localmente síncrono (*GALS*), uma vez que as suas unidades consistem em máquinas de estado finitas síncronas com os sinais de relógio independentes e a transferência de informação entre unidades do Meshotron adjacentes baseia-se na utilização de *FIFOs* com dois sinais de relógio, um sinal de leitura e outro de escrita (*dual-clock*).

Capítulo 3 Arquitectura da unidade-base do Meshotron

3.1. Introdução

O projecto da unidade do Meshotron (para a topologia rectangular) desenvolveu-se em três fases. Começou-se por implementá-la segundo as directrizes dos artigos [15] que previam, na etapa de *scattering*, fases alternadas de leitura e escrita de dados, com duplo endereçamento dos correspondentes bancos de memória, varridos em sucessivos segmentos.

Numa segunda abordagem, dispensou-se este varrimento segmentado através da paralelização das operações de escrita e leitura no módulo de *scattering* e endereçamento simultâneo dos bancos de memória de entrada e saída de dados. Isto permitiu otimizar recursos e diminuir o tempo de execução da etapa de *scattering* para cerca de metade.

Finalmente, através de uma gestão mais sofisticada dos bancos de memória, que exigiu também esquemas de endereçamento um pouco mais complexos, foi possível integrar o *scattering pass* e as primeiras etapas do *delay pass* (*D1* e *D2*) numa só sequência, poupando quase metade do tempo de execução (aceleração em relação à versão inicial por um factor próximo de 3).

Considerou-se útil, para maior clareza de exposição, reflectir a evolução que acaba de ser descrita na estrutura deste capítulo. Descrever-se-á numa primeira secção a solução inicial e, em secções seguintes, as alterações nela introduzidas.

3.2. Abordagem inicial

3.2.1. *Data path*

3.2.1.1. Estrutura geral

A sequência de cálculos do *scattering pass* e as transferências de informação do *delay pass*, incluindo a comunicação com blocos adjacentes, podem ser implementadas por adequado controlo do fluxo de dados na estrutura representada na Fig. 11. Descrevem-se em seguida os seus diferentes elementos. Note-se que todos eles são síncronos, isto é, todas as operações de escrita e leitura são controladas por um sinal de relógio.

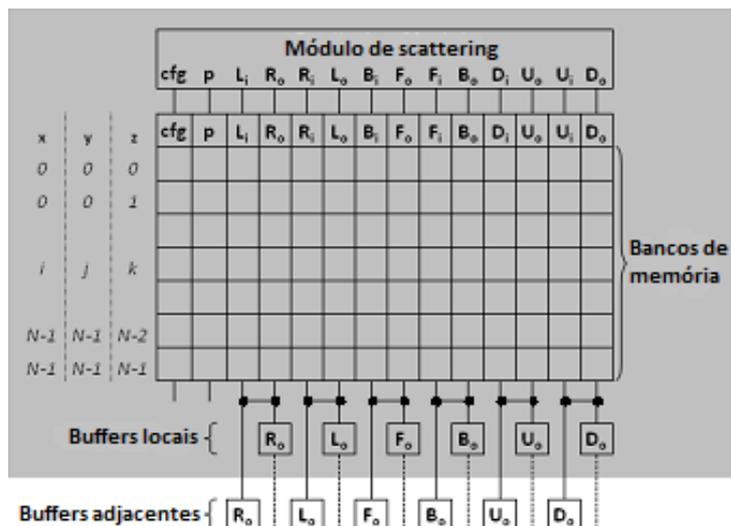


Fig. 11 – Encaminhamento de dados (*data path*) numa unidade do Meshotron na abordagem inicial.

3.2.1.2. Bancos de memória

Cada unidade do Meshotron opera sobre uma partição cúbica do modelo, constituída por uma matriz de N^3 nós. Cada nó é caracterizado por catorze registos de informação:

- um código de configuração, para definir se ele modela o meio de propagação (*nó de ar*) ou as superfícies que o delimitam (*nó fronteira*) e, neste caso, as respectivas propriedades acústicas (coeficiente de reflexão) – *cfg*;
- o valor corrente da pressão acústica – *p*;
- os valores dos seis pares de componentes da onda de pressão acústica que entram e saem do nó, como ilustra a Fig. 8: L_i/L_o e R_i/R_o para o eixo xx , B_i/B_o e F_i/F_o para o eixo yy , D_i/D_o e U_i/U_o para o eixo zz .

Assim, para armazenar a informação de todos os N^3 nós, a unidade do Meshotron integra catorze bancos de memória de N^3 elementos, como indicava a Fig. 11. Optou-se pela utilização de memórias *dual-port*, como mostra a Fig. 12, embora nesta abordagem inicial pudessem ter sido usadas memórias *single-port*. A vantagem desta opção ficará mais clara na secção 3.3.1, onde se explica o aumento de *throughput* através da paralelização das operações de leitura e escrita no módulo de *scattering*, só possível com memórias *dual-port*.

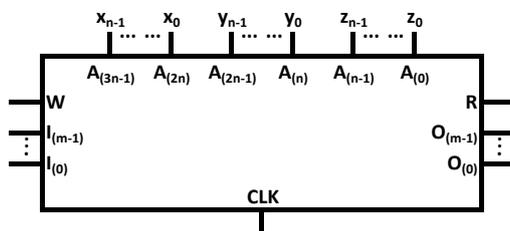


Fig. 12 – Interface externa de um banco de memória.

A escrita na memória (pelo barramento I) e a sua leitura (pelo barramento O), são controladas, respectivamente, pelos sinais presentes nos pinos W (*write*) e R (*read*).

O barramento de endereços é representado por A. O número de *bits* de endereço necessário para endereçar as N^3 posições de memória, é

$$b = \log_2(N^3) = 3\log_2 N. \quad \text{Eq. 15}$$

Para que b seja inteiro, N deve ser uma potência inteira de 2:

$$N = 2^n \quad (n \text{ inteiro}) \quad \text{Eq. 16}$$

Portanto,

$$b = 3\log_2(2^n) = 3n. \quad \text{Eq. 17}$$

Como evidencia a Fig. 12, no barramento de endereços podem distinguir-se 3 segmentos de n bits, que correspondem, por ordem decrescente de significância, às coordenadas x , y e z .

3.2.1.3. Buffers de comunicação

A comunicação com unidades adjacentes é efectuada através de seis *buffers dual-port* como o representado na Fig. 13. Estes *buffers* contêm N^2 elementos, correspondendo ao número de valores que é necessário transferir nas etapas $D1$ e $D3$ do *delay pass*.

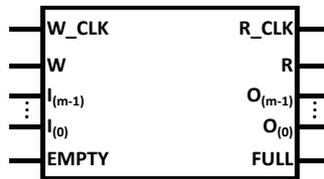


Fig. 13 – Interface externa de um *buffer* de comunicação.

Estas unidades de comunicação não requerem qualquer tipo de endereçamento, uma vez que constituem filas (*queues*) *first-in, first-out* (FIFO).

Tal como nos bancos de memória (vide Fig. 12), a escrita e a leitura de dados, respectivamente pelos barramentos I e O, são controladas pelos sinais presentes nos pinos W (*write*) e R (*read*). Existe um par de pinos adicionais que informam sobre o estado da FIFO: FULL (activo quando a FIFO se encontra completamente cheia) e EMPTY (activo quando a FIFO se encontra completamente vazia). São estes sinais que permitem definir, como se explica na secção 3.2.2.1, qual das duas unidades que trocam informação pelo *buffer* está autorizada a aceder-lhe em cada momento.

A partilha do controlo da FIFO entre duas unidades do Meshotron (aquela a que pertence e uma outra adjacente) obriga a que existam dois sinais de relógio independentes, respectivamente para escrita pela própria unidade em $D1$ (W_CLK) e leitura pela unidade adjacente em $D3$ (R_CLK).

3.2.1.4. Módulo de *scattering*

O módulo de *scattering* (vide Fig. 14) recebe os dados de entrada dos nós, provenientes dos bancos de memória c_f , L_p , R_p , B_p , F_p , D_p e U_p , e efectua os cálculos de *scattering*. Os dados de saída resultantes são recolhidos nos endereços correspondentes (determinados pelos *drivers* de endereçamento) dos bancos de memória p , R_o , L_o , F_o , B_o , U_o e D_o .

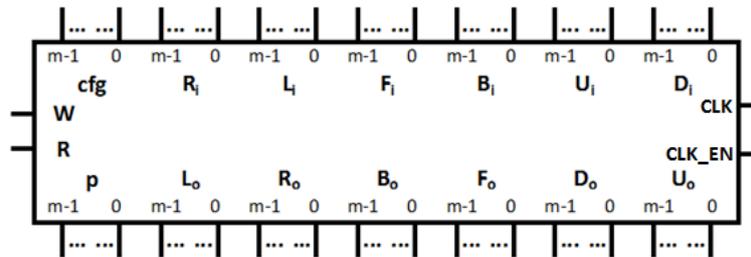


Fig. 14 – Interface externa do módulo de *scattering*.

Internamente, é constituído por um conjunto de d_s unidades de *scattering* funcionando em paralelo. Um *demultiplexer* e um *multiplexer*, sob controlo de um contador cíclico de d_s bits (vide Fig. 15), permitem escolher, em cada momento, que unidades são acedidas respectivamente para injeção e recolha de dados. Note-se como os sinais de controlo W (escrita) e R (leitura) actuam através da habilitação do *demultiplexer* e do *multiplexer*. A habilitação do contador cíclico tem que ser controlada por AEN (*Address Clock Enable*) – vide secção 3.2.4 – de modo a sincronizar a selecção das unidades de *scattering* com os ciclos de endereçamento.

Para assegurar o correcto funcionamento, é necessário que as unidades de *scattering* tenham tempo de completar o seu cálculo no intervalo entre a injeção e a recolha de dados. Esse intervalo corresponde exactamente a d_s ciclos de endereçamento; basta, pois, que d_s seja, no mínimo, igual ao tempo de cálculo medido em ciclos de endereçamento.

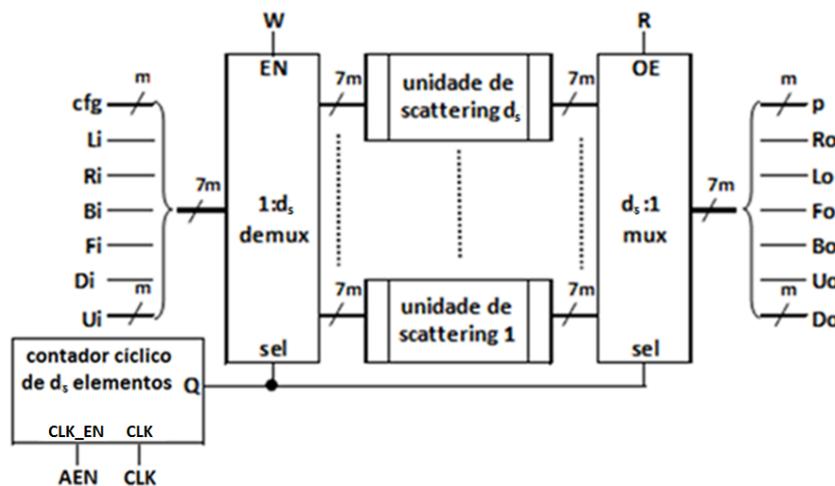


Fig. 15 – Estrutura interna do módulo de *scattering*.

Nesta abordagem, consideram-se fases alternadas de escrita e leitura nas unidades de *scattering*: A d_s ciclos de escrita (leitura da memória) seguem-se d_s ciclos de leitura (escrita na memória), e assim sucessivamente. Deste modo, as unidades de *scattering* são utilizadas durante apenas metade do tempo, como ilustra a Fig. 16 (onde se assume, a título de exemplo, que o cálculo de *scattering* demora quatro ciclos de endereçamento e $d_s=4$). De igual forma, os bancos de memória só operam 50% do tempo, pois a leitura de dados tem que ser suspensa nas fases de escrita, e vice-versa.

Estes inconvenientes são resolvidos na abordagem descrita na secção 3.3.

Operação	Leitura				Escrita				Leitura				Escrita				Leitura				Escrita		
Ciclo de endereçamento	1	2	3	4	1	2	3	4	5	6	7	8	5	6	7	8	9	10	11	12	9	10	11
Unidade de scattering 1	Cálculo 1								Cálculo 1								Cálculo 5						
Unidade de scattering 2		Cálculo 2								Cálculo 2								Cálculo 6					
Unidade de scattering 3			Cálculo 3								Cálculo 3								Cálculo 7				
Unidade de scattering 4				Cálculo 4										Cálculo 4							Cálculo 8		
Endereço de leitura nas memórias X_i	1	2	3	4	X	X	X	X	X	X	X	X	X	X	X	X	5	6	7	8	X	X	X
Unidade de scattering acedida	1	2	3	4	X	X	X	X	1	2	3	4	X	X	X	X	1	2	3	4	X	X	X
Endereço de escrita das memórias X_o	X	X	X	X	X	X	X	X	1	2	3	4	X	X	X	X	X	X	X	X	X	X	X

Fig. 16 – Abordagem inicial para a gestão da etapa de *scattering*.

O módulo de *scattering* pode ser visto externamente como uma fila *FIFO* de d_s elementos com duplo barramento de dados (*dual-port*). Tem, pois, estrutura similar aos *buffers* de comunicação, mas com um único sinal de relógio.

3.2.1.5. Formato dos dados

Nos diagramas de interface apresentados anteriormente (Fig. 12, Fig. 13 e Fig. 14), m é o número de *bits* dos dados. Exceptuando o caso do parâmetro cfg , este número condiciona a precisão numérica do modelo, uma vez que os cálculos de *scattering* envolvem inevitavelmente aproximações numéricas devidas à operação de divisão, explicada em detalhe no Apêndice B. Como consequência, a escolha do valor de m é um aspecto importante do projecto da unidade do Meshotron (vide secção 4.2.1). Por agora considerar-se-á, para simplificar, que m é igual para todos os dados, incluindo cfg .

3.2.1.6. Unidade de *scattering*

O elemento central da estrutura de processamento de dados do Meshotron é a unidade de *scattering*, que tem por função efectuar os cálculos do *scattering pass* de acordo com o código de configuração do nó (*nó fronteira* ou *nó de ar* - vide secção 2.2). Para a topologia adoptada (rectangular), os cálculos relativos aos *nós de ar* são descritos na Eq. 5 e Eq. 13, enquanto que o cálculo relativo aos *nós fronteira* é descrito pela Eq. 6. No actual estado de desenvolvimento, a unidade de *scattering* não diferencia os *nós de ar* dos *nós fronteira*. Esta circunstância não constitui uma limitação muito grave, uma vez que a validação geral do circuito não é afectada: é possível criar e testar modelos com significado físico recorrendo à simulação de paredes reflectoras nas faces do bloco cúbico. Para este efeito, basta injectar novamente nos *nós* que formam essas faces os valores armazenados nos respectivos *buffers* de comunicação; isto equivale a implementar *nós fronteira* com $R=1$.

A estrutura da unidade de *scattering* é representada na Fig. 17. Trata-se de um circuito constituído por um somador de seis entradas de m *bits*, um divisor por 3 e seis substractores de m *bits*.

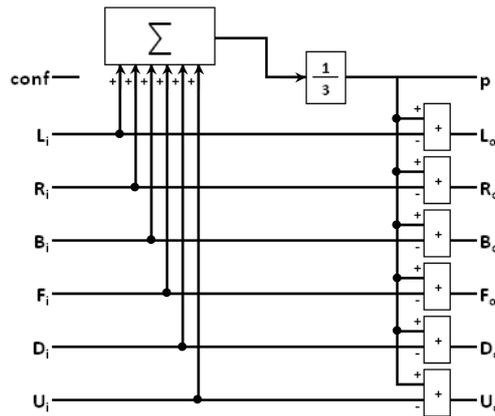


Fig. 17 – Estrutura de uma unidade de *scattering* para nós de ar.

A forma mais óbvia de obter uma unidade de *scattering* que contemple *nós de ar* e *nós fronteira* é utilizar um *multiplexer* controlado pelo registo *cfg* para seleccionar entre o circuito da Fig. 17 e o da Fig. 18, que é baseado na Eq. 6.

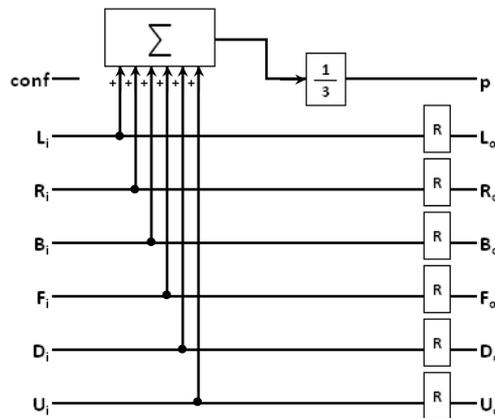


Fig. 18 - Estrutura de uma unidade de *scattering* para nós fronteira.

3.2.1.6.1. Divisor por três

O divisor por três é o elemento da unidade de *scattering* mais complexo de implementar; a operação de divisão não é, aliás, suportada directamente em *VHDL*. Para resolver este problema testaram-se três possibilidades de solução:

- Circuito especializado para divisão por uma constante;
- Bloco divisor da *Xilinx*;
- Multiplicador por $1/3$;

Consideradas as respectivas vantagens e desvantagens (vide Apêndice B), optou-se pela última.

3.2.1.6.2. Unidade de detecção e prevenção de *overflow*

A unidade de *scattering* está sujeita à possibilidade de *overflow* (vide Eq. 5 e Eq. 13). Por isso, utilizou-se um método de impedir que ele possa ser causado pelos cálculos internos. Este consiste na utilização três *bits* adicionais durante a manipulação dos dados. A principal razão que conduz à situação de *overflow* na Eq. 5 é o facto de a operação de soma ser

efectuada antes da operação de divisão por 3 (vide Fig. 17). Ao considerar o pior cenário possível para a soma (6 parcelas iguais ao máximo valor possível – vide Eq. 18) verifica-se que a extensão de três *bits* permite salvaguardar a situação de *overflow* (na verdade, ela permitiria somar até oito valores semelhantes – vide Fig. 19).

$$\text{Soma} = 6(2^{m-1} - 1) = 3(2^m) - 6 < 4(2^m) - 1. \quad \text{Eq. 18}$$

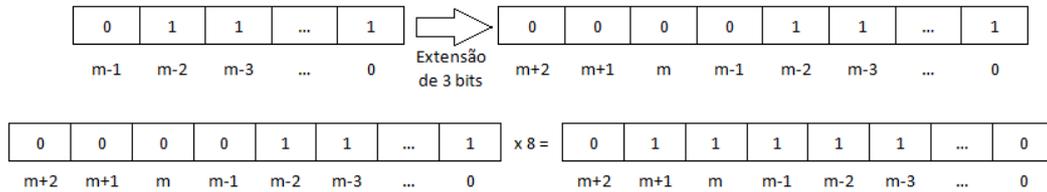


Fig. 19 – Salvaguarda da situação de *overflow* por extensão de 3 *bits*.

O número de *bits* utilizado na extensão poderia ser minimizado por recurso à propriedade distributiva para inverter a ordem das operações de adição e divisão. Todavia, esta opção é intrinsecamente pior em termos de recursos consumidos e de precisão numérica.

Os *overflows* causados naturalmente pela operação do modelo não podem ser evitados, uma vez que, no final das operações efectuadas pela unidade de *scattering*, os três *bits* adicionais (os mais significativos) têm que ser eliminados para que os dados de saída retornem ao formato de m *bits*. Construíram-se, por isso, detectores de *overflow* à saída do bloco de divisão por 3 e dos substractores. Estes componentes analisam os três *bits* mais significativos; se forem diferentes entre si, é sinalizada a ocorrência de *overflow*.

A Fig. 20 mostra o circuito final da unidade de *scattering* equipada com os elementos de detecção de *overflow* descritos.

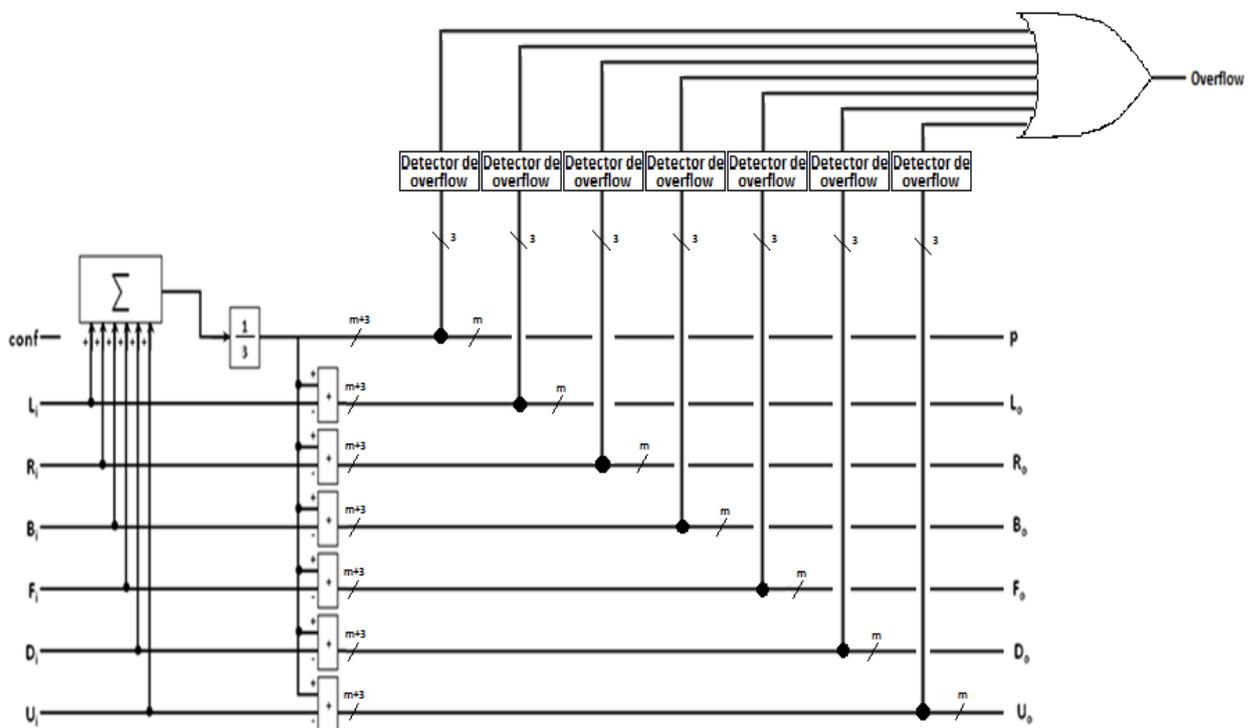


Fig. 20 – Estrutura da unidade de *scattering* para *nós de ar* com detecção de *overflow*.

3.2.1.7. Barramento de dados

Os bancos de memória *cfg* e *p* só estão envolvidos no *scattering pass*, pelo que, durante o processamento, só interagem com o módulo de *scattering*, como ilustra a Fig. 11. O fluxo de dados é extremamente simples: *cfg* é apenas lido e *p* apenas escrito. O banco *cfg* só necessita ser acedido para escrita na fase de inicialização (antes do processamento propriamente dito); durante o *delay pass*, permanece inactivo. O banco *p* só necessita ser acedido para leitura para efeito de extracção da *RIR* na posição de memória correspondente à localização do nó receptor. Isto pode ser feito durante o *delay pass*.

O fluxo de dados que envolve os restantes doze bancos de memória é mais complexo. Existe um total de seis estruturas de barramento de dados idênticas, que correspondem aos seis sentidos de propagação espacial: x^- , x^+ , y^- , y^+ , z^- e z^+ . Por exemplo, a estrutura no sentido positivo do eixo dos x (x^+) encontra-se representada na Fig. 21. As restantes estruturas são exactamente iguais, *mutatis mutandis*.

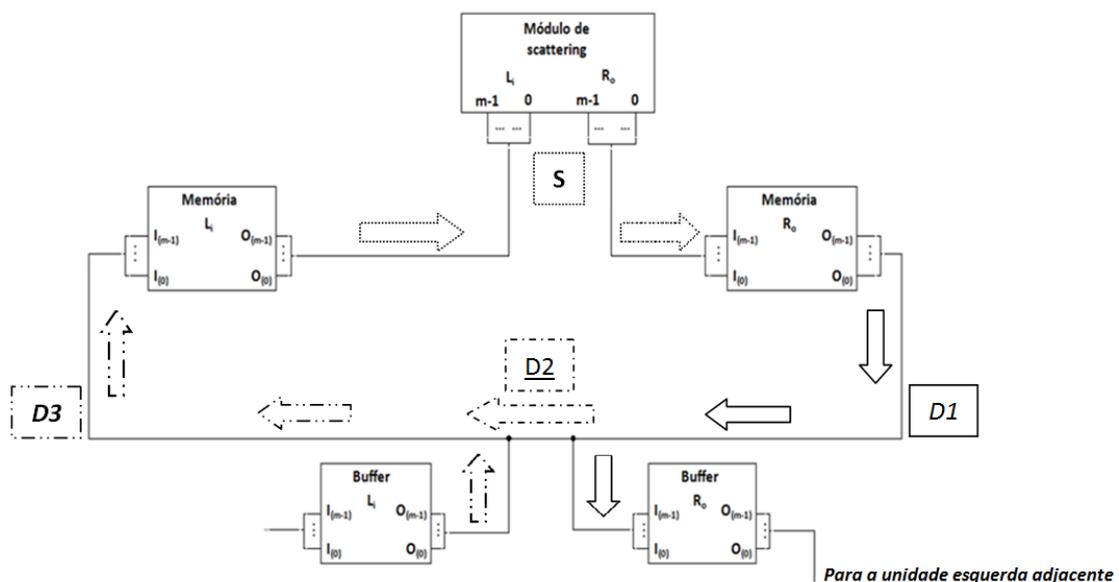


Fig. 21 – Barramento de dados para o sentido positivo do eixo x. As setas representam o fluxo de dados em cada etapa (S, D1, D2 e D3).

Legenda:

- Fluxo de dados na etapa S;
- Fluxo de dados na etapa D1;
- Fluxo de dados na etapa D2;
- Fluxo de dados na etapa D3.

O fluxo de dados depende da etapa em execução:

- Etapa *S* (*scattering pass*): este decorre em ciclos alternados de injeção a partir dos bancos de memória de entrada e extracção para os bancos de memória de saída;
- Etapa *D1* (*delay pass 1*): o fluxo de dados é feito dos bancos de memória de saída para os *buffers* locais;
- Etapa *D2* (*delay pass 2*): os dados são transferidos dos bancos de memória de saída para os bancos de memória de entrada;
- Etapa *D3* (*delay pass 3*): o fluxo de dados efectua-se dos *buffers* das unidades adjacentes para os bancos de memória de entrada.

3.2.2. Bloco de controlo

3.2.2.1. Sequência geral de computação

Ao bloco de controlo da unidade do Meshotron compete estabelecer a sequência iterativa de computação (*scattering pass* – *S* – seguido de *delay pass* – *D1*, *D2* e *D3*), estabelecendo os modos de endereçamento de memória necessários em cada etapa e emitindo sinais de controlo adequados para os diferentes componentes do *data path*. Este pode ser projectado como uma máquina de estados síncrona (vide Fig. 22).

Verifica-se a possível ocorrência de estados de espera antes das etapas *D1* e *D3* (respectivamente *W1* e *W2*). Isto deve-se ao facto de a etapa *D1* só poder iniciar-se após conclusão da etapa *D3* da iteração anterior em todas as unidades do Meshotron adjacentes, ou seja, quando se verifica a condição *albe* (*all local buffers empty* – vide Fig. 23). De forma análoga, a etapa *D3* apenas pode iniciar-se quando a etapa *D1* da mesma iteração é concluída em todas as unidades do Meshotron adjacentes, isto é, na condição *aabf* (*all adjacent buffers full* – vide Fig. 23).

O mecanismo de semáforos descrito na Fig. 23 disciplina as operações de transferência de informação entre unidades adjacentes, garantindo a integridade dos dados nos *buffers* de comunicação por elas partilhados.

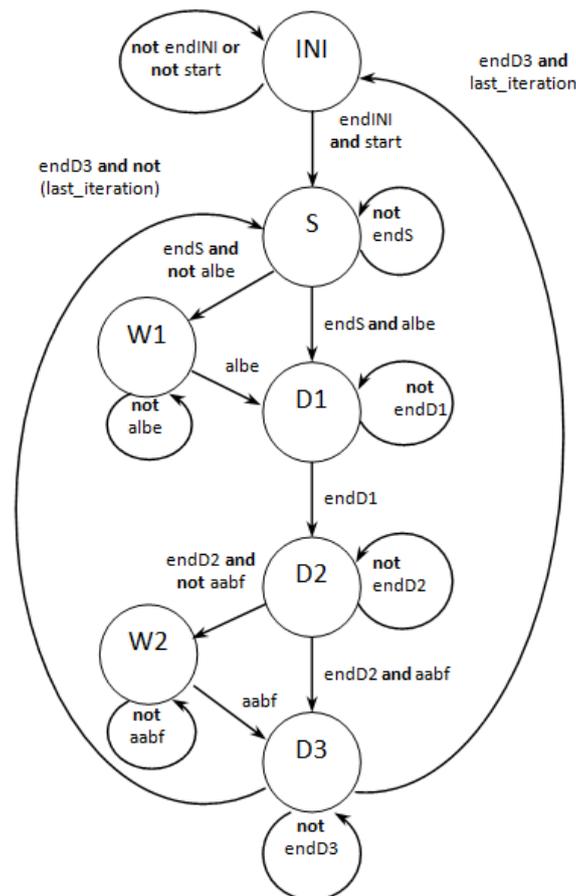


Fig. 22 – Diagrama de estados do bloco de controlo da unidade do Meshotron.

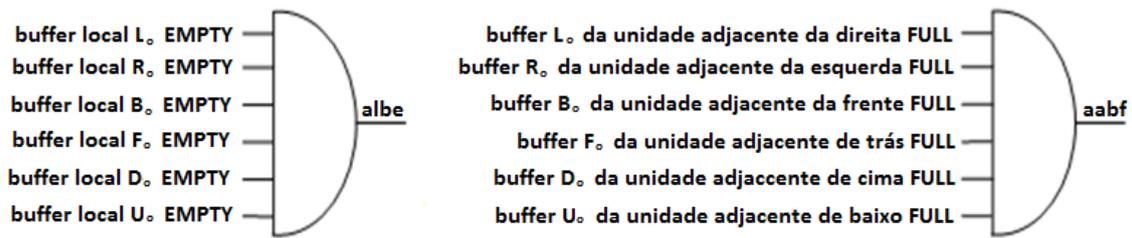


Fig. 23 – Obtenção das condições *albe* (*all local buffers empty*) e *aabf* (*all adjacent buffers full*) a partir dos semáforos dos *buffers*.

3.2.2.2. Endereçamento

3.2.2.2.1. Etapa de inicialização

A etapa de inicialização resume-se a uma operação de escrita de valores em todas as posições dos bancos de memória de entrada. Como não foi ainda contemplada uma fase explícita de excitação do modelo acústico, esta efectuou-se durante a etapa de inicialização. Assim, todas as posições de memória são preenchidas com 0 excepto as correspondentes ao nó fonte. Nestas, colocou-se um valor de excitação que simulasse a injeção de um impulso de *Dirac* no modelo acústico. Para conseguir um impulso de amplitude A , é necessário injectar em todos os registos de entrada do nó o valor $A/2$, como se pode verificar na Eq. 13. Assim, o limite máximo que não conduz a uma situação de *overflow* é 2^{m-2} .

Nesta etapa percorrem-se todas as posições de memória, como esquematiza a Tabela I.

Tabela I – Esquema do endereçamento da etapa *INI*.

<i>INI</i> :	Origem	Destino	
		Banco de memória	Endereço
Unidade de inicialização		L_i	(x,y,z)
		R_i	
		B_i	
		F_i	
		U_i	
		D_i	

Os bancos de memória de entrada são varridos na mesma sequência, desde o endereço $(0,0,0)$ até ao endereço $(N-1,N-1,N-1)$. O circuito que gera esta sequência de endereços é o simples contador de $3n$ bits representado na Fig. 24.

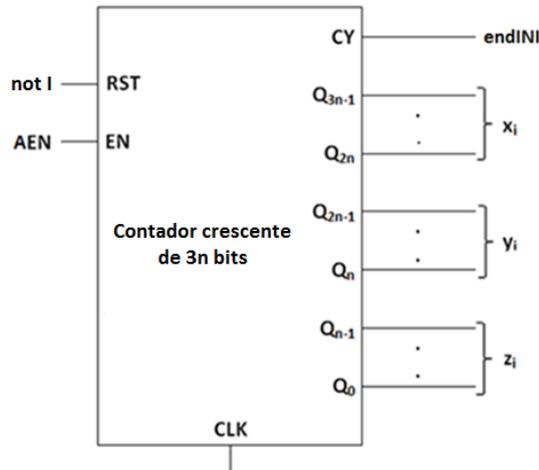


Fig. 24 – Gerador de endereços da etapa *INI*.

O pino EN (*Enable*) habilita o gerador de endereços e é controlado pelo sinal AEN (*Address Clock Enable* - vide secção 3.2.4); RST (*Reset*) inicializa o contador quando a etapa *INI* não está em execução e CY (*Max Count*) sinaliza quando é atingido o último endereço e, conseqüentemente, o final da etapa de inicialização (*endINI*).

Esta etapa dura N^3 ciclos de endereçamento.

3.2.2.2.2. *Scattering pass*

Nesta abordagem, existem fases alternadas de escrita e leitura de dados nas unidades de *scattering*. Em cada ciclo de escrita/leitura é tratado um determinado segmento dos dados; é necessário um duplo varrimento dos endereços de memória que lhe correspondem, para aceder, sucessivamente, aos bancos de memória de entrada (*SecondSweep=0*) e de saída (*SecondSweep=1*), como esquematiza a Tabela II. Os bancos de memória são varridos na mesma seqüência, desde o endereço (0,0,0) até ao endereço (N-1,N-1,N-1).

Tabela II – Esquema de endereçamento da etapa S (abordagem inicial).

S:	Origem		Através	Destino		
	SecondSweep	Banco de memória		Endereço	SecondSweep	Banco de memória
0		L_i	Módulo de <i>scattering</i>	1	L_o	(x,y,z)
		R_i			R_o	
		B_i			B_o	
		F_i			F_o	
		D_i			D_o	
		U_i			U_o	
		cfg			p	

O tamanho do segmento escolhido tem que corresponder ao número de unidades de *scattering*, d_s . O circuito da Fig. 25, constituído por dois contadores em cascata, implementa o esquema de endereçamento considerando $d_s=N$. Assim, em cada ciclo, apenas a coordenada z varia, sendo percorridos todos os valores possíveis: no primeiro varrimento do primeiro ciclo, é efectuada a leitura dos valores presentes nos bancos de memória de entrada desde o endereço (0,0,0) até ao endereço (0,0,N-1); no segundo varrimento, é

efectuada a escrita dos valores calculados nos mesmos endereços dos bancos de memória de saída. O processo repete-se para todas as combinações das restantes coordenadas espaciais (x e y).

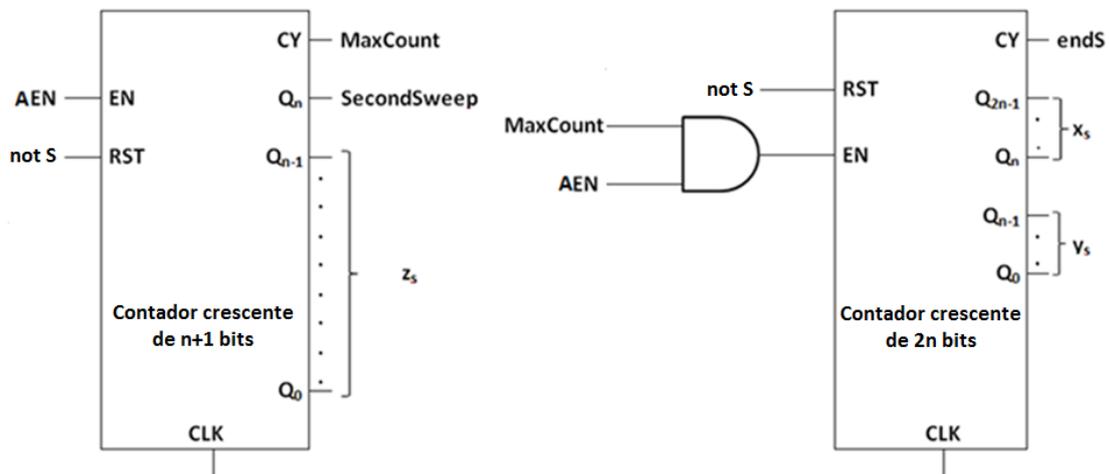


Fig. 25 – Gerador de endereços da etapa S na abordagem inicial.

O contador da esquerda é um contador de $n+1$ bits: o *bit* mais significativo à saída (Q_n) deste contador (designado *SecondSweep*) identifica o varrimento (0 ou 1, respectivamente para leitura ou escrita). Este contador gera os endereços da coordenada z . Por sua vez, o contador da direita (de $2n$ bits) só fica activo quando é atingida a contagem máxima no contador menos significativo. Desta forma, são gerados os endereços das restantes coordenadas espaciais.

Este gerador de endereços é inicializado através de RST (*Reset*) quando a etapa S não está em execução e habilitado (através de EN) através do sinal AEN (*Address Clock Enable*). O pino CY (*Max Count*) sinaliza o último valor em cada contador e, para o contador mais significativo, o final do *scattering pass* (*endS*).

Com a abordagem que acaba de ser descrita, esta etapa demora $2N^3$ ciclos de endereçamento. A abordagem otimizada a apresentar na secção 3.3.2 permite reduzir drasticamente tal duração.

3.2.2.2.3. *Delay pass*

O *delay pass* divide-se em três etapas: $D1$, $D2$ e $D3$.

3.2.2.2.3.1. Etapas $D1$ e $D3$

Durante estas etapas, a transferência de informação envolve unicamente os nós das seis faces do bloco cúbico e correspondentes *buffers* de comunicação (que têm N^2 elementos cada). A face do bloco desejada pode ser seleccionada fixando uma das coordenadas espaciais no valor mínimo ou máximo (respectivamente 0 ou $N-1$) e efectuando o varrimento das restantes. Os esquemas de endereçamento de $D1$ e $D3$ são análogos, como evidencia a Tabela III. As transferências são simultâneas em todos os seis sentidos espaciais, o que leva a que, quer $D1$, quer $D3$, durem apenas N^2 ciclos de endereçamento.

Tabela III – Esquemas de endereçamento das etapas D1 e D3.

	Origem		Destino
	Banco de memória	Endereços	Buffer local
Eixo do x	L _o	(0,y,z)	L _o
	R _o	(N-1,y,z)	R _o
Eixo do y	B _o	(x,0,z)	B _o
	F _o	(x,N-1,z)	F _o
Eixo do z	D _o	(x,y,0)	D _o
	U _o	(x,y,N-1)	U _o

	Origem	Destino	
	Buffer adjacente	Banco de memória	Endereços
L _o	R _i	(N-1,y,z)	
R _o	L _i	(0,y,z)	
B _o	F _i	(x,N-1,z)	
F _o	B _i	(x,0,z)	
D _o	U _i	(x,y,N-1)	
U _o	D _i	(x,y,0)	

O contador de $2n$ bits ilustrado na Fig. 26 gera uma sequência de (0,0) a (N-1, N-1). Os segmentos de n bits h_{D13} e l_{D13} definem as duas coordenadas não fixas; adequadamente combinados com sequências de n zeros (0) ou uns (N-1) para a terceira coordenada, permitem criar todas as sequências de endereçamento sumariadas na Tabela III.

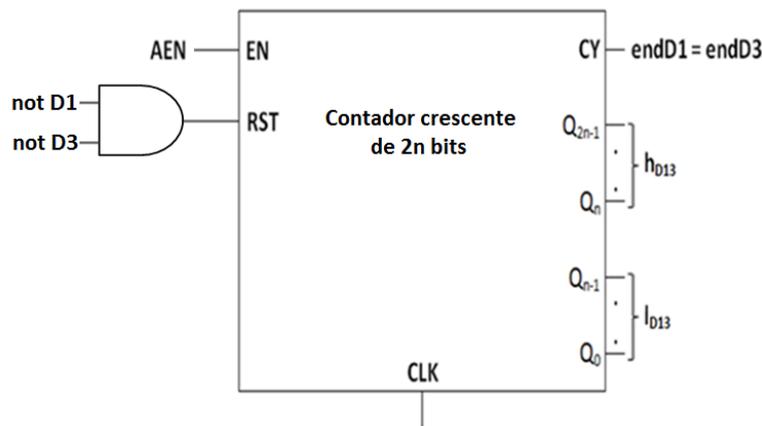


Fig. 26 – Gerador de endereços das etapas D1 e D3.

O circuito é habilitado pelo pino EN (*Enable*) que é controlado pelo sinal AEN (*Address Clock Enable*); o pino RST (*Reset*) garante que a contagem é efectuada nas etapas D1 ou D3. O sinal CY (*Max Count*) sinaliza o final da contagem ($endD1 = endD3$).

3.2.2.2.3.2. Etapa D2

Nesta etapa, as transferências de informação efectuam-se directamente entre bancos de memória (nenhum dos buffers de comunicação se encontra envolvido). Como a transferência de informação é efectuada entre nós adjacentes, os endereços de origem e destino diferem sempre por apenas uma unidade em uma das coordenadas espaciais, tal como mostra a Tabela IV.

Tabela IV – Esquemas de endereçamento da etapa D2.

D2:	Origem		Destino		Condição
	Banco de memória	Endereços	Banco de memória	Endereço	
Eixo x	L _o	(x,y,z)	R _i	(x-1,y,z)	x>0
	R _o	(x-1,y,z)	L _i	(x,y,z)	
Eixo y	B _o	(x,y,z)	F _i	(x,y-1,z)	y>0
	F _o	(x,y-1,z)	B _i	(x,y,z)	
Eixo z	D _o	(x,y,z)	U _i	(x,y,z-1)	z>0
	U _o	(x,y,z-1)	D _i	(x,y,z)	

O circuito da Fig. 27, que consiste numa cascata de três contadores de n bits, permite gerar as seqüências de endereços implícitas na Tabela IV; basta combinar adequadamente os segmentos menos significativos, l_{D2} e m_{D2} , e um dos mais significativos, h_{D2} ou $h_{D2}+$. O segmento h_{D2} está sempre atrasado por uma unidade relativamente ao segmento $h_{D2}+$ devido aos estados iniciais impostos (por força de PL) nos contadores mais significativos.

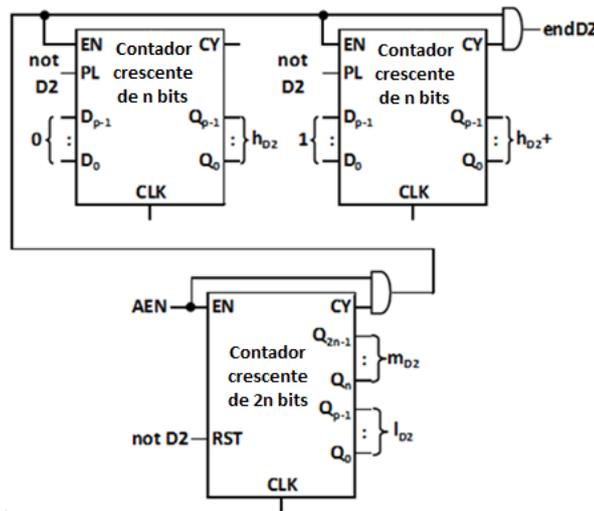


Fig. 27 – Gerador de endereços da etapa D2.

O circuito é habilitado por AEN (*Address Clock Enable*), de forma a contar ciclos de endereçamento. Através do pino RST (*Reset*) garante-se ainda que só o faz na etapa D2. Os sinais CY (*Max Count*) sinalizam o último valor de contagem; a partir deles, é gerada a condição que sinaliza o final da etapa (*endD2*). Note-se que esta depende do segmento $h_{D2}+$, que só percorre $N-1$ endereços, pois começa em 1 e não em 0. Assim, esta fase demora $(N-1)N^2$ ciclos de endereçamento.

Note-se que, tal como em D1 e D3, todas as transferências de informação são efectuadas simultaneamente nos seis sentidos espaciais.

3.2.3. Drivers de endereçamento

O endereçamento dos diferentes bancos de memória é efectuado por catorze *drivers* (um por banco de memória) operando em coordenação. Trata-se de *multiplexers* controlados pela etapa da máquina de estados que controla a unidade do Meshotron (vide Fig. 22), para seleccionar o gerador de endereço apropriado entre os apresentados na secção 3.2.2.2 (INI, S, D1/D3 ou D2).

As entradas de cada *driver* de endereçamento são descritas na Tabela V. O símbolo 'X' indica irrelevância. Note-se que nas etapas não contempladas na tabela (ou seja, nos *wait states W1* e *W2*), o endereçamento é também completamente irrelevante, uma vez que os elementos de armazenamento (bancos de memória e *buffers* de comunicação) estão inactivos.

Tabela V – Endereçamento de cada banco de memória nas diferentes etapas (abordagem inicial).

Etapas	Segmentos endereços	cfg	Eixo:		X		Y				Z				p	Gerador de endereços	
			Destino:														
			→	←	↓	↑	•	⊗									
			L_i	R_o	R_i	L_o	B_i	F_o	F_i	B_o	D_i	U_o	U_i	D_o			
INI	x	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i	x_i		Fig. 24
	y	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i	y_i		
	z	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i	z_i		
S	x	x_{Sr}	x_{Sr}	x_{Sw}	x_{Sw}		Fig. 25										
	y	y_{Sr}	y_{Sr}	y_{Sw}	y_{Sw}												
	z	z_{Sr}	z_{Sr}	z_{Sw}	z_{Sw}												
D1	x	X	X	N-1	X	0	X	h_{D13}	X	h_{D13}	X	h_{D13}	X	h_{D13}	X		Fig. 26
	y	X	X	h_{D13}	X	h_{D13}	X	N-1	X	0	X	l_{D13}	X	l_{D13}	X		
	z	X	X	l_{D13}	X	l_{D13}	X	l_{D13}	X	l_{D13}	X	N-1	X	0	X		
D2	x	X	h_{D2+}	h_{D2}	h_{D2}	h_{D2+}	m_{D2}	X		Fig. 27							
	y	X	m_{D2}	m_{D2}	m_{D2}	m_{D2}	h_{D2+}	h_{D2}	h_{D2}	h_{D2+}	l_{D2}	l_{D2}	l_{D2}	l_{D2}	X		
	z	X	l_{D2}	h_{D2+}	h_{D2}	h_{D2}	h_{D2+}	X									
D3	x	X	0	X	N-1	X	h_{D13}	X	h_{D13}	X	h_{D13}	X	h_{D13}	X	X		Fig. 26
	y	X	h_{D13}	X	h_{D13}	X	0	X	N-1	X	l_{D13}	X	l_{D13}	X	X		
	z	X	l_{D13}	X	l_{D13}	X	l_{D13}	X	l_{D13}	X	0	X	N-1	X	X		

3.2.4. Sinais de controlo

Considerando as operações de transferência de informação realizadas nas diferentes etapas e a estrutura do barramento de dados, é fácil estabelecer como devem ser controlados os diferentes elementos que compõem o *data-path*, i.e., que bancos de memória e *buffers* de comunicação devem estar activos e para que efeito em cada momento. A Tabela VI (onde se assumem sinais activos-altos) especifica em termos gerais (i.e. sem ter em conta os detalhes temporais dos ciclos de acesso às memórias), os valores lógicos que devem assumir, em cada etapa, os sinais de escrita e leitura.

Note-se como a sequência de sinais de controlo mais complexa ocorre na etapa de *scattering*, devido à necessidade de duplo varrimento da memória descrito na secção 3.2.2.2.2.

Tabela VI – Sinais de controlo de escrita e leitura em cada etapa (abordagem inicial).

Etapas	Módulo de <i>scattering</i>		Bancos de memória								Buffers		
			cfg		p		X _i		X _o		Locais	Adjacentes	
	R	W	R	W	R	W	R	W	R	W	W	R	
INI	0	0	0	1	0	1	0	1	0	0	0	1	0
S	Varrimento de escrita	0	1	1	0	0	0	1	0	0	0	0	0
	Varrimento de leitura	1	0	0	0	0	1	0	0	0	1	0	0
D1	0	0	0	0	1	0	0	0	1	0	1	0	
D2	0	0	0	0	1	0	0	1	1	0	0	0	
D3	0	0	0	0	1	0	0	1	0	0	0	1	

Os sinais de controlo têm que ser adequadamente temporizados para garantir que os endereços estejam válidos no barramento antes da activação dos sinais de leitura; de igual forma, a informação presente no barramento de dados tem de se encontrar válida antes dos sinais de escrita serem activados (estes requisitos são universais no acesso à memória em sistemas síncronos). Para este efeito, geraram-se três sinais de *Clock Enable* – AEN, REN e WEN – usando um contador cíclico de três *bits* sensível aos flancos descendentes do relógio (partindo do princípio que os restantes elementos síncronos da unidade são sensíveis aos flancos ascendentes do sinal de relógio), como ilustra a Fig. 28.

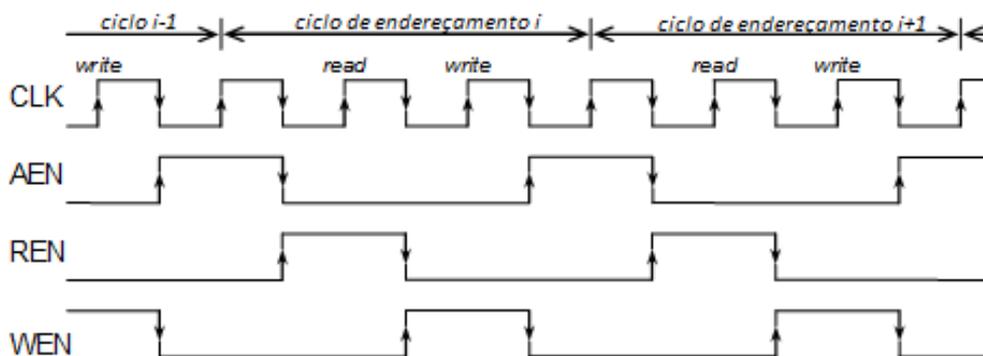


Fig. 28 – Sinais de *clock enable* para as operações de endereçamento, escrita e leitura.

A frequência do ciclo de endereçamento é definida pelo sinal AEN (*Addressing Clock Enable*), pois ele é usado para habilitar os geradores de endereços especificados nas secções 3.2.2.2, 3.3.2 e 3.4.2.2. Além disso, este sinal também constitui o *Clock Enable* da máquina de estados que controla a unidade do Meshotron (vide Fig. 22), garantindo assim que as transições entre etapas são sempre síncronas com a actualização de endereços.

Dentro do ciclo de endereçamento, as janelas temporais em que os sinais de escrita e leitura produzem realmente efeito nos componentes seleccionados, são definidas pelos sinais REN (*Read Enable*) e WEN (*Write Enable*). Combinando estes sinais com a informação existente na Tabela VI, resultam as funções booleanas da Tabela VII, que descrevem os diferentes sinais de controlo.

Tabela VII – Expressões booleanas dos sinais de controlo (abordagem inicial).

Módulo de scattering	R=S and SecondSweep and REN	W=S and (not SecondSweep) and WEN
Banco de memória cfg	R=S and REN	W =INI and WEN
Banco de memória p	R=REN and (D1 or D2 or D3)	W=(INI or S) and WEN
Bancos de memória X_i	R=S and REN	W=(INI or D2 or D3) and WEN
Bancos de memória X_o	R=(D1 or D2) and REN	W=S and WEN
Buffers locais		W=D1 and WEN
Buffers adjacentes	R=D3 and REN	

3.2.5. Tempo de computação

É importante avaliar o tempo de cálculo do algoritmo de modelação executado por uma unidade do Meshotron. Tal implica conhecer a duração, em ciclos de endereçamento, por iteração de cada etapa de processamento.

Nesta abordagem inicial, distinguem-se, além da etapa de inicialização, que só ocorre uma vez e demora N^2 ciclos de endereçamento, um ciclo iterativo constituído por quatro etapas (vide secção 3.2.2.1). A duração de cada uma depende do seu esquema de endereçamento (vide secção 3.2.2.2) e é, naturalmente, função do número de *bits* do barramento de endereços, que reflecte o volume da partição do modelo. A Tabela VIII apresenta um resumo.

Tabela VIII - Duração de cada etapa na abordagem inicial.

Etapas	Número de ciclos de endereçamento
S	$2N^3$
D1	N^2
D2	$(N-1)N^2$
D3	N^2

Esta contabilização não contempla os *wait states* eventualmente introduzidos pelo mecanismo de sincronização das unidades do Meshotron ($W1$ e $W2$ - vide secção 3.2.2.1), pois eles só ocorrem em unidades cujo sinal de relógio tenha frequência superior à dos restantes; por outras palavras, a duração contabilizada em ciclos de endereçamento deve ser convertida em tempo considerando o relógio mais lento do Meshotron.

Assim a duração de uma iteração, t_m , é dada pela Eq. 19.

$$t_m = 2N^3 + N^2 + [(N-1)N^2] + N^2 = 3N^3 + N^2 \cong 3N^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 19}$$

Considerando que o cálculo requer k iterações, o tempo de computação total, T_{RIR} , será:

$$T_{RIR} = k(3N^3 + N^2) \cong 3kN^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 20}$$

Atendendo a que, como explicado na secção 3.2.4, cada ciclo de endereçamento compreende três ciclos de relógio, a duração total em termos de ciclos de relógio é dada por:

$$T_{RIR} = \left(\frac{3}{f_{clk}} \right) kN^3 = \frac{9kN^3}{f_{clk}} \text{ (s)}, \text{ onde } f_{clk} \text{ como o frequência de relógio (Hz)}. \quad \text{Eq. 21}$$

Para uma unidade com $N=8$ ter-se-á uma duração de:

$$T_{RIR} = \frac{4608k}{f_{clk}} \text{ (s)}, \text{ onde } f_{clk} \text{ como frequência do sinal de relógio}. \quad \text{Eq. 22}$$

3.3. Segunda abordagem

O princípio de funcionamento do módulo de *scattering* da abordagem inicial limitava a eficiência de utilização das unidades de *scattering*, uma vez que estas permaneciam inactivas durante metade do tempo no *scattering pass* (vide secção 3.2.1.4).

Para otimizar o funcionamento do módulo de *scattering*, aproveitando melhor as suas unidades, abandonou-se a ideia de varrimento segmentado; optou-se por paralelizar as operações de escrita e leitura, com endereçamento simultâneo dos bancos de memória de entrada e saída de dados. Isto implicou modificações no gerador de endereços da etapa de *scattering* e nos sinais de controlo relacionados.

3.3.1. Módulo de *scattering*

A estrutura geral do módulo (Fig. 15) mantém-se inalterada modificando-se apenas o controlo de alguns dos seus elementos. O comportamento do contador cíclico de d_s elementos mantém-se. Porém, os sinais de controlo W (escrita) e R (leitura) passam a estar activos simultaneamente, fazendo o *demultiplexer* e o *multiplexer*, operar em sincronismo. Isto significa que, embora continue a haver apenas uma unidade de *scattering* acedida em cada momento, ela passa a sê-lo simultaneamente para escrita de novos dados e leitura dos resultados do cálculo anterior. Deste modo, optimiza-se o desempenho do módulo de *scattering*, uma vez que os dados podem ser escritos e lidos simultaneamente e de forma contínua à taxa mais elevada possível de acesso à memória. Isto é ilustrado na Fig. 29, onde se assume (tal como na Fig. 16, com que deve ser comparada), que o cálculo da unidade de *scattering* demora quatro ciclos de endereçamento e $d_s = 4$.

Ciclo de endereçamento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Unidade de <i>scattering</i> 1	Cálculo 1				Cálculo 5				Cálculo 9							
Unidade de <i>scattering</i> 2		Cálculo 2			Cálculo 6			Cálculo 10								
Unidade de <i>scattering</i> 3			Cálculo 3		Cálculo 7		Cálculo 11									
Unidade de <i>scattering</i> 4				Cálculo 4			Cálculo 8			Cálculo 12						
Endereço de leitura nas memórias X_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Unidade de <i>scattering</i> acedida	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Endereço de escrita das memórias X_o	X	X	X	X	1	2	3	4	5	6	7	8	9	10	11	12

Fig. 29 – Segunda abordagem para a gestão da etapa de *scattering*.

Note-se que a paralelização da operação de escrita e leitura obriga a que os bancos de memória sejam *dual-port*, uma vez que, doutro modo, ocorreria colisão no mesmo barramento entre os dados de entrada e saída do módulo de *scattering*.

Comparando a Fig. 16 com a Fig. 29, comprova-se a vantagem que adveio da paralelização das operações de escrita e leitura no módulo de *scattering*.

3.3.2. Endereçamento do *scattering pass*

A paralelização das operações de escrita e leitura no módulo de *scattering* obriga ao endereçamento simultâneo dos bancos de memória de entrada e saída. O esquema de endereçamento está sumariado na Tabela IX.

Tabela IX – Esquema de endereçamento da etapa S (segunda abordagem).

S:	Origem		Através	Destino	
	Banco de memória	Endereços		Banco de memória	Endereços
	L_i	(x,y,z)	Módulo de <i>scattering</i>	L_o	(x,y,z)
	R_i			R_o	
	B_i			B_o	
	F_i			F_o	
	D_i			D_o	
	U_i			U_o	
	cfg			p	

Os bancos de memória são varridos na mesma sequência de endereços, desde o endereço $(0,0,0)$ até ao endereço $(N-1,N-1,N-1)$. No entanto, o endereçamento dos bancos de memória de saída tem de ser atrasado d_s ciclos, para se ajustar ao atraso de processamento no módulo de *scattering*. Esta condição é conseguida através do circuito com dois contadores de $3n$ bits e um *flip-flop* SR da Fig. 30.

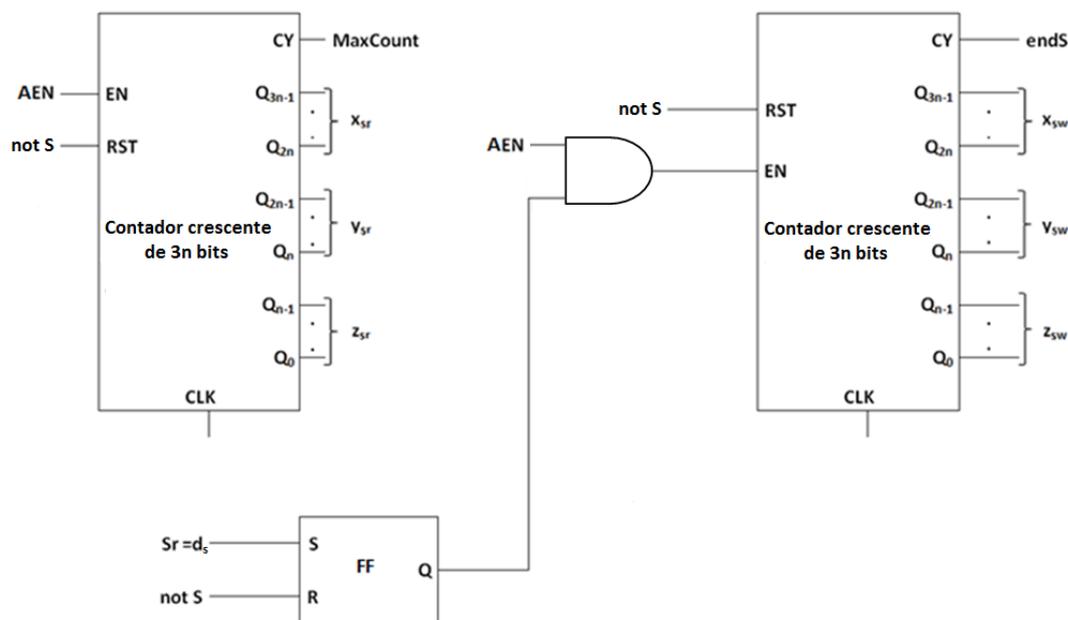


Fig. 30 – Gerador de endereços da etapa S (segunda abordagem).

O contador da esquerda, habilitado pelo sinal AEN (*Address Clock Enable*), gera a sequência de endereços para leitura da memória e só activa o contador da direita, através do *flip-flop* SR, quando atingir d_s . Este segundo contador proporciona assim a sequência de endereçamento atrasada, como pretendido, para a operação de escrita nos bancos de memória de saída. O pino RST garante que a contagem é efectuada na etapa S.

O pino CY (*Max Count*) do contador da direita sinaliza o final do *scattering pass* (*endS*). Nesta abordagem, a etapa S demora $N^3 + d_s$ ciclos de endereçamento, devido ao atraso de d_s ciclos existente no contador da direita.

A informação inválida escrita no endereço (0,0,0) da memória de saída durante os d_s ciclos iniciais não é prejudicial para o funcionamento, uma vez que é substituída pela informação correcta no ciclo seguinte, d_s+1 . De forma análoga, a informação carregada para o módulo de *scattering* nos últimos d_s ciclos de endereçamento não tem qualquer tipo de consequência.

3.3.3. Sinais de controlo

A alteração do princípio de funcionamento do módulo de *scattering* implica modificações nos sinais de controlo; a Tabela VI transforma-se na Tabela X, continuando a assumir que todos os sinais em análise são activos-altos.

Tabela X – Sinais de controlo de escrita e leitura em cada etapa (segunda abordagem).

Etapas	Módulo de <i>scattering</i>		Bancos de memória								Buffers	
			cfg		p		X_i		X_o		Locais	Adjacentes
	R	W	R	W	R	W	R	W	R	W	W	R
INI	0	0	0	1	0	1	0	1	0	0	1	0
S	1	1	1	0	0	1	1	0	0	1	0	0
D1	0	0	0	0	1	0	0	0	1	0	1	0
D2	0	0	0	0	1	0	0	1	1	0	0	0
D3	0	0	0	0	1	0	0	1	0	0	0	1

Como o mecanismo de geração dos sinais de *Clock Enable* não sofre alterações, assim, da Tabela X obtêm-se as expressões booleanas da Tabela XI.

Tabela XI – Expressões booleanas para os diferentes sinais de controlo na segunda abordagem.

Módulo de <i>scattering</i>	R=S and REN	W=S and WEN
Banco de memória <i>cfg</i>	R=S and REN	W =INI and WEN
Banco de memória <i>p</i>	R=REN and (D1 or D2 or D3)	W=(INI or S) and WEN
Banco de memória X_i	R=S and REN	W=(INI or D2 or D3) and WEN
Banco de memória X_o	R=(D1 or D2) and REN	W=S and WEN
Buffers locais		W=D1 and WEN
Buffers adjacentes	R=D3 and REN	

3.3.4. Tempo de computação

Nesta abordagem efectuou-se uma optimização do método de endereçamento do *scattering pass*, que vai levar a reduções do número de endereços na etapa *S* como ilustra a Tabela XII. As restantes etapas permanecem inalteradas em termos de duração.

Tabela XII – Duração de cada etapa na segunda abordagem.

Etapas	Número de ciclos de endereçamento
S	N^3+d_s
D1	N^2
D2	$(N-1)N^2$
D3	N^2

Pelas mesmas razões mencionadas na secção 3.2.5 a duração dos *wait states* ($W1$ e $W2$ – vide secção 3.2.2.1) não será contemplada; os cálculos tomarão mais uma vez como base a unidade mais lenta.

Logo a duração de uma iteração, t_m , é dada pela Eq. 23.

$$t_m = (N^3 + d_s) + N^2 + (N-1)N^2 + N^2 = 2N^3 + N^2 + d_s \cong 2N^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 23}$$

Considerando que se pretende a duração de k iterações então o tempo de computação total, T_{RIR} , é dado por:

$$T_{RIR} = 2kN^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 24}$$

Como cada ciclo de endereçamento corresponde a três ciclos de relógio (vide secção 3.2.4), e sendo f_{clk} a frequência de relógio (Hz) ter-se-á:

$$T_{RIR} = \frac{6kN^3}{f_{clk}} \text{ (s)}. \quad \text{Eq. 25}$$

Para uma unidade com $N=8$ ter-se-á uma duração de:

$$T_{RIR} = \frac{3072k}{f_{clk}} \text{ (s)}. \quad \text{Eq. 26}$$

A duração de uma iteração diminuiu cerca de 1/3 relativamente à abordagem inicial (vide Eq. 19 e Eq. 23), um melhoramento muito significativo que otimiza o desempenho do Meshotron.

3.4. Abordagem final

A abordagem final é a mais complexa. Para perceber quais as alterações a serem efectuadas torna-se relevante explicar o raciocínio seguido que conduziu a esta abordagem. Para tal fez-se uma breve introdução sobre o procedimento efectuado na secção 3.3, segunda abordagem para as etapas S e $D2$. A partir deste, descreveram-se quais as alterações a efectuar, de modo a garantir a junção das três etapas mencionadas.

Recapitulando na segunda abordagem as transferências efectuadas para as etapas S e $D2$ estão ilustradas na Fig. 31.

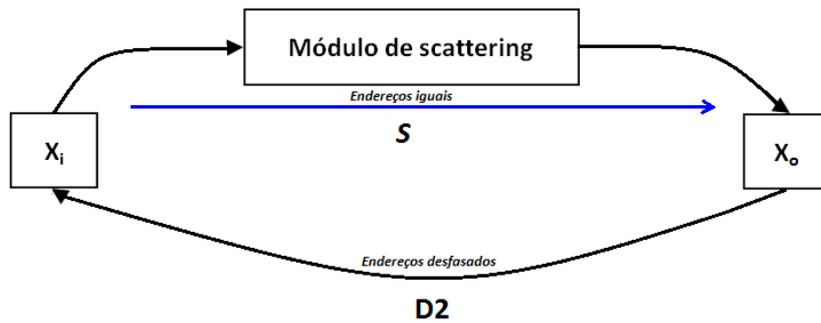


Fig. 31 – Transferências efectuadas nas etapas S e $D2$ da segunda abordagem.

Vejamos agora a abordagem final: numa primeira etapa (correspondente à etapa $SD12$ – *scattering delay pass 12*) os valores dos bancos de memória de entrada são transferidos para o módulo de *scattering*. Mas, em vez de os resultados obtidos serem transferidos para os mesmos endereços dos bancos de memória de saída como na abordagem anterior, estes são transferidos para os endereços destino da etapa $D2$ sem quaisquer restrições (as condições impostas para cada direcção deixam de ter efeito – vide Tabela IV). Desta forma garante-se que todos os valores presentes nos bancos de memória de saída são exactamente os

mesmos presentes nos bancos de memória de entrada no final da etapa *D2* na segunda abordagem.

No entanto esta alteração acarreta um problema: os valores presentes nos bancos de memória de saída deveriam estar nos bancos de memória de entrada o que vai comprometer as iterações seguintes do Meshotron. A solução passa por abandonar a ideia de memórias com uma definição estática (memória de saída e memória de entrada). No final de cada iteração efectua-se uma comutação às memórias, e deste modo as memórias de saída na iteração anterior são definidas como memórias de entrada na iteração seguinte (vide Fig. 32).

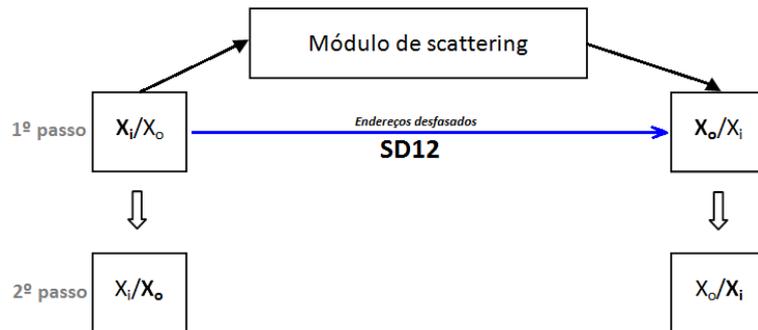


Fig. 32 – Transferências de informação na da etapa *S* e *D2* efectuadas na etapa *SD12* da abordagem final.

A explicação permite depreender que esta abordagem implica alterações no barramento de dados, no bloco de controlo, nos sinais de controlo e nos *drivers* de endereçamento, com a construção de um gerador de endereços que contemple as condições impostas pelas três etapas.

3.4.1. *Data path*

3.4.1.1. Estrutura geral

A complexidade desta abordagem leva a que a apresentação da estrutura geral se divida em duas partes: numa primeira é apresentada a estrutura para uma das direcções espaciais e numa segunda é apresentada a estrutura estendida a todas a direcções espaciais. A sequência de cálculos e transferências de informação nesta abordagem é implementada por adequado controlo do fluxo de dados na estrutura representada na Fig. 33 correspondente à direcção positiva do eixo do *x*.

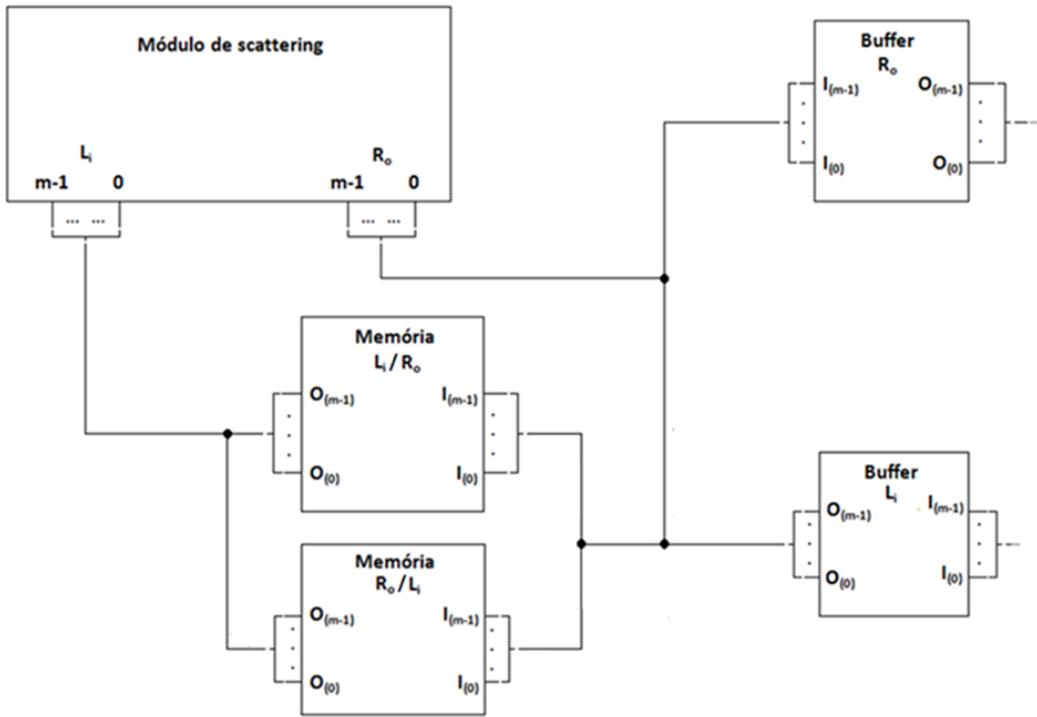


Fig. 33 - Estrutura de dados (*data path*) de uma unidade do Meshotron na abordagem final para a direcção positiva do eixo do x.

Partindo deste esquema a estrutura geral que contempla todas as direcções espaciais está ilustrada na Fig. 34.

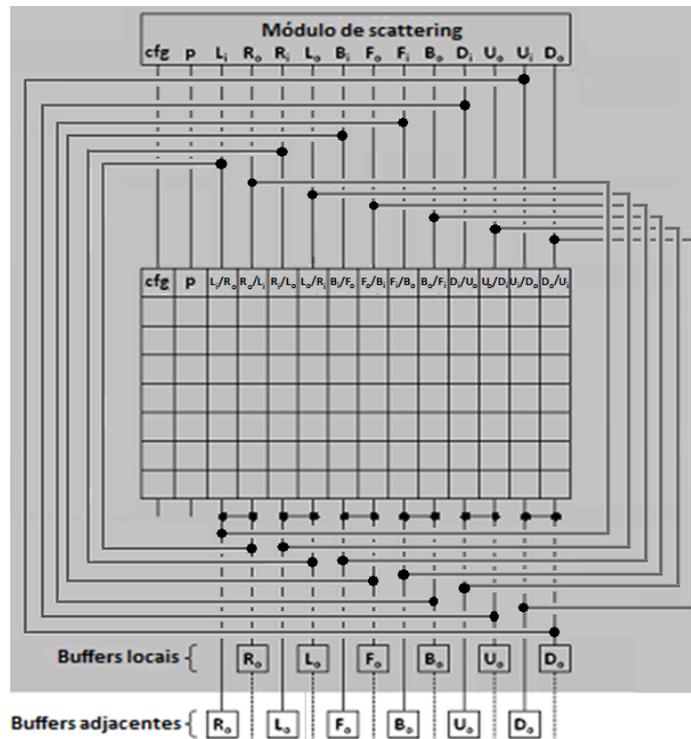


Fig. 34 - Encaminhamento de dados (*data path*) numa unidade do Meshotron na abordagem final.

3.4.1.2. Barramento de dados

Em termos de estrutura do barramento de dados efectuou-se a sua colocação e interligação com os diferentes elementos de armazenamento (bancos de memória e *buffers* de comunicação) e de cálculo (módulo de *scattering*), como ilustra a Fig. 35. Note-se que a estrutura representada é unicamente para um dos eixos do sistema tridimensional (eixo dos xx).

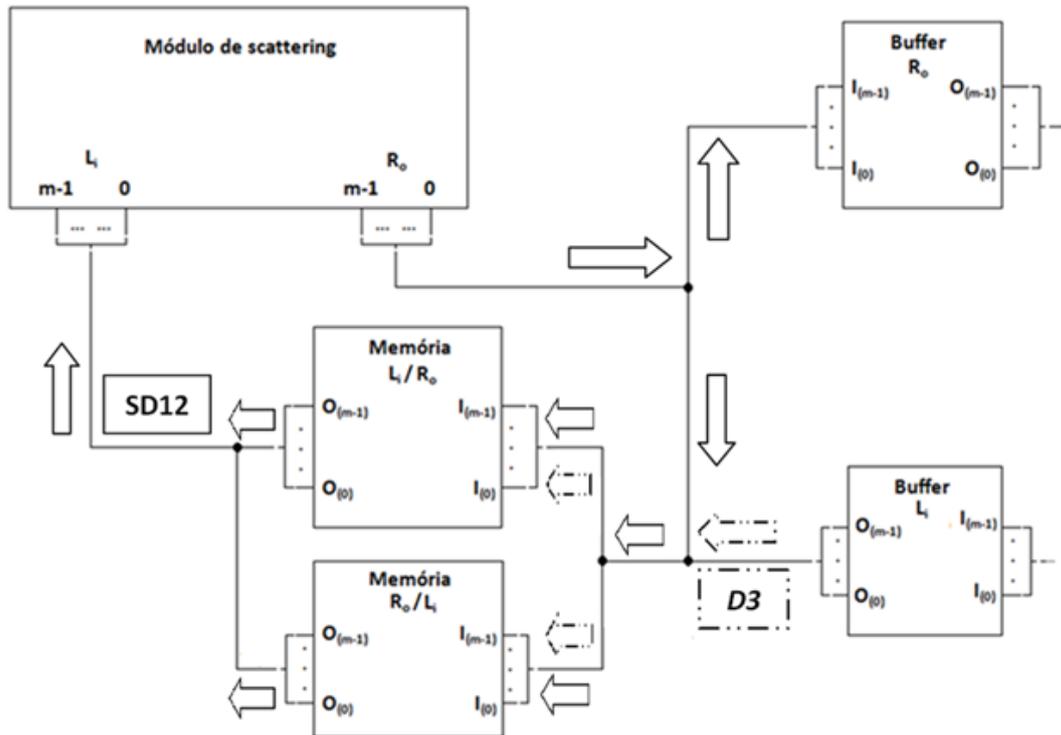


Fig. 35 – Barramento de dados para a direcção positiva do eixo do x na abordagem final. As setas representam o fluxo de dados em cada etapa (*SD12* e *D3*).

Legenda:

- Fluxo de dados na etapa *SD12*;
- Fluxo de dados na etapa *D3*.

O fluxo de dados depende da etapa em execução:

- Etapa *SD12* (*scattering delay pass 12*): os dados são transferidos dos bancos de memória definidos como entrada para o módulo de *scattering*, e assim que disponíveis os resultados obtidos são transferidos para os bancos de memória definidos como saída se pertencerem a nós interiores à malha, ou para os *buffers* locais se pertencerem aos nós de superfície do bloco.
- Etapa *D3* (*delay pass 3*): o fluxo de dados efectua-se dos *buffers* adjacentes para os bancos de memória definidos como saída;

No final da etapa *D3* as memórias de saída da iteração actual são definidas como as memórias de entrada na iteração seguinte.

O princípio base da optimização implica a comutação em cada iteração entre os bancos de memória. Comutação esse possibilitada por uma máquina de estados que selecciona qual os bancos de memória de entrada e saída em cada iteração. Esta comutação vai resumir a

operação do *delay pass 2* a um ciclo de relógio, desde que durante o *scattering pass* os resultados calculados sejam transferidos para os endereços destino do *delay pass 2*. O sinal de saída da máquina de estados consiste num *bit* que define se um determinado banco de memória é um banco de memória de entrada ou de saída. O sinal que controla a máquina de estados consiste no sinal que sinaliza o final do *delay pass 3* (*endD3* – final de um iteração), como ilustra a Fig. 36.

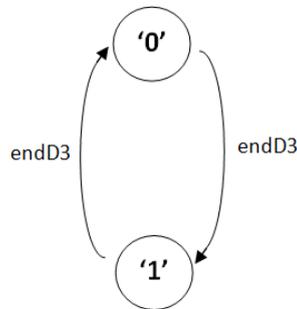


Fig. 36 – Máquina de estados de definição dos bancos de memória de entrada e saída.

A otimização não implicou qualquer tipo de alterações no fluxo de dados dos bancos *cfg* e *p*. O banco *cfg* é escrito na etapa de inicialização, lido quando as operações do *scattering pass* são efectuadas (etapa *SD12*) e para as restantes etapas permanece inactivo. Por sua vez o banco *p* é escrito durante a etapa *SD12* e lido durante a etapa *D3* para extracção da *RIR*.

3.4.2. Bloco de controlo

3.4.2.1. Sequência de controlo geral

Em termos de diagrama de estados esta abordagem levou a modificações na estrutura inicial mencionada na secção 3.2.2.1, como ilustra a Fig. 37.

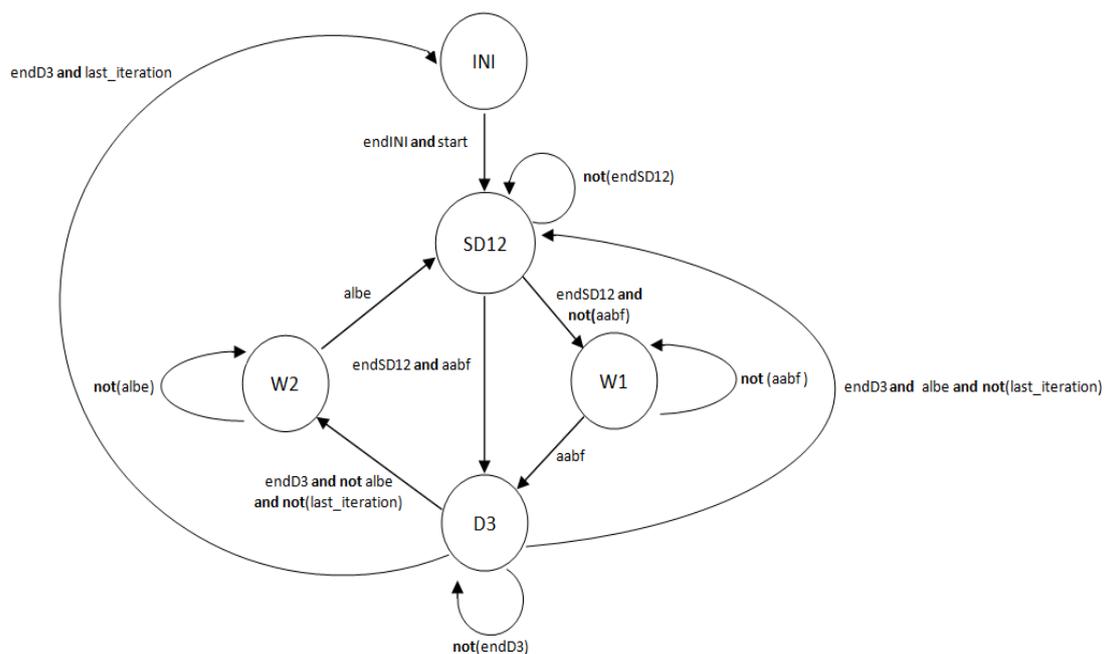


Fig. 37 - Diagrama de estados do bloco de controlo da abordagem final de uma unidade do Meshotron.

Os sinais de controlo explicitados são exactamente os mesmos cujas funções booleanas estão representadas na Fig. 23. Os *wait states* não sofreram alterações, uma vez que a optimização não modificou o método de sincronização entre unidades do Meshotron. A secção seguinte explicará o endereçamento para a nova etapa *SD12* representada na Fig. 37.

3.4.2.2. Endereçamento do *scattering delay pass 12* – Etapa *SD12*

A nova etapa implicou um varrimento sequencial dos bancos de memória definidos como entrada, desde o endereço (0,0,0) até ao endereço (N-1,N-1,N-1). O endereçamento aos bancos de memória definidos como saída teve de ser semelhante ao *delay pass 2* (vide Tabela IV), de modo a garantir que os resultados das operações são colocados nas posições de memória da etapa *D2*. No entanto para garantir que os cálculos de *scattering* são efectuados antes das operações do *delay pass* o endereçamento aos bancos de memória definidos como saída foi atrasado d_s ciclos. Note-se que deste modo o princípio de paralelização da operação de leitura e escrita do módulo de *scattering* permaneceu inalterado (vide secção 3.3.2).

O princípio de funcionamento do gerador de endereços está sumariado na Tabela XIII.

Tabela XIII – Esquemas de endereçamento da etapa *SD12*.

<i>SD12:</i>	Origem		Através	Destino		
	Banco de memória	Endereços		Banco de memória	Endereços	Buffer local
Eixo do x	L _i /R _o	(0,y,z)	Módulo de <i>scattering</i>			L _o
		(x,y,z)		R _o /L _i	(x-1,y,z)	
	R _i /L _o	(N-1,y,z)				R _o
		(x,y,z)		L _o /R _i	(x+1,y,z)	
Eixo do y	B _i /F _o	(x,0,y)				B _o
		(x,y,z)		F _o /B _i	(x,y-1,z)	
	F _i /B _o	(x,N-1,z)				F _o
		(x,y,z)		B _o /F _i	(x,y+1,z)	
Eixo do z	D _i /U _o	(x,y,0)			D _o	
		(x,y,z)	U _o /D _i	(x,y,z-1)		
	U _i /D _o	(x,y,N-1)			U _o	
		(x,y,z)	D _o /U _i	(x,y,z)		

O circuito da Fig. 38; que consiste num conjunto de dois contadores de $3n$ bits, num *flip-flop* SR e num conjunto de três somadores e substractores; implementa as condições especificadas na Tabela XIII.

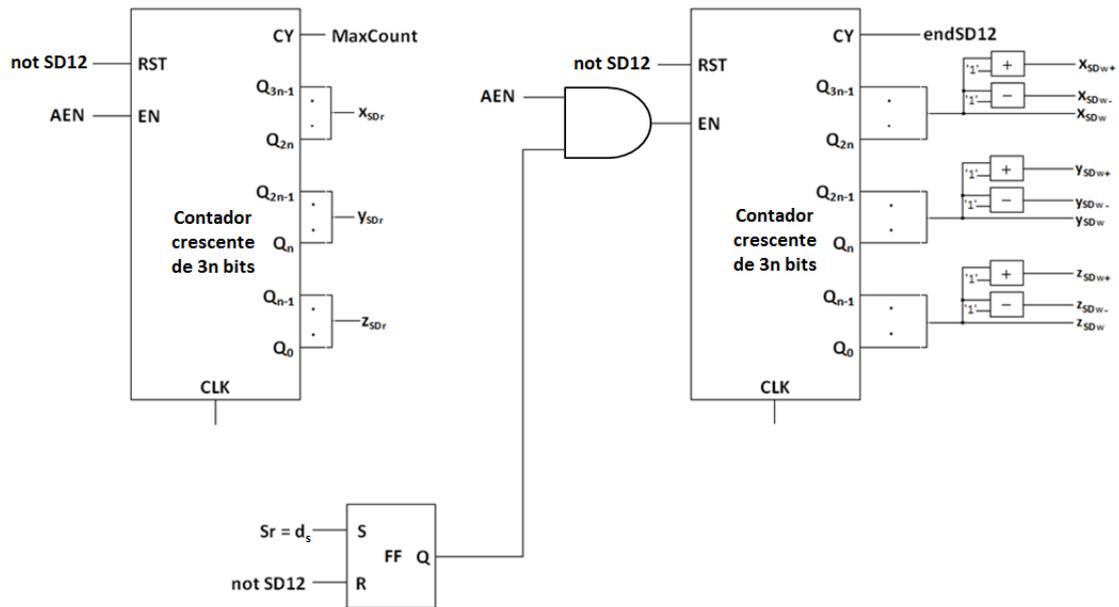


Fig. 38 – Gerador de endereços da etapa SD12.

O contador da esquerda, habilitado pelo sinal AEN (*Address Clock Enable*), activa o *flip-flop* SR quando atingir d_s . O contador da direita controlado pelo *flip-flop* SR é activo com d_s ciclos de atraso relativamente ao contador da esquerda. O conjunto de três somadores e substractores impõem as condições de endereçamento descritas na Tabela XIII para os bancos de memória definidos como saída. O pino RST inicializa o gerador de endereços quando a etapa SD12 não está em execução; o pino CY (*Max Count*) sinaliza o final da etapa SD12 (*endSD12*).

Apesar de o gerador de endereços fazer uso de um conjunto de somadores e substractores para implementar as condições associadas a cada uma das direcções a sua utilização não é obrigatória. Existem alternativas de implementação das condições de endereçamento mais eficientes, uma vez que a utilização destes elementos combinatórios introduz um atraso no gerador de endereços. Uma alternativa passa pela divisão do contador de $3n$ bits em três contadores de n bits e na utilização de um conjunto de seis contadores em paralelo, três inicializados no valor 1 e os restantes inicializados no valor $N-1$ ligados de acordo com a Fig. 39.

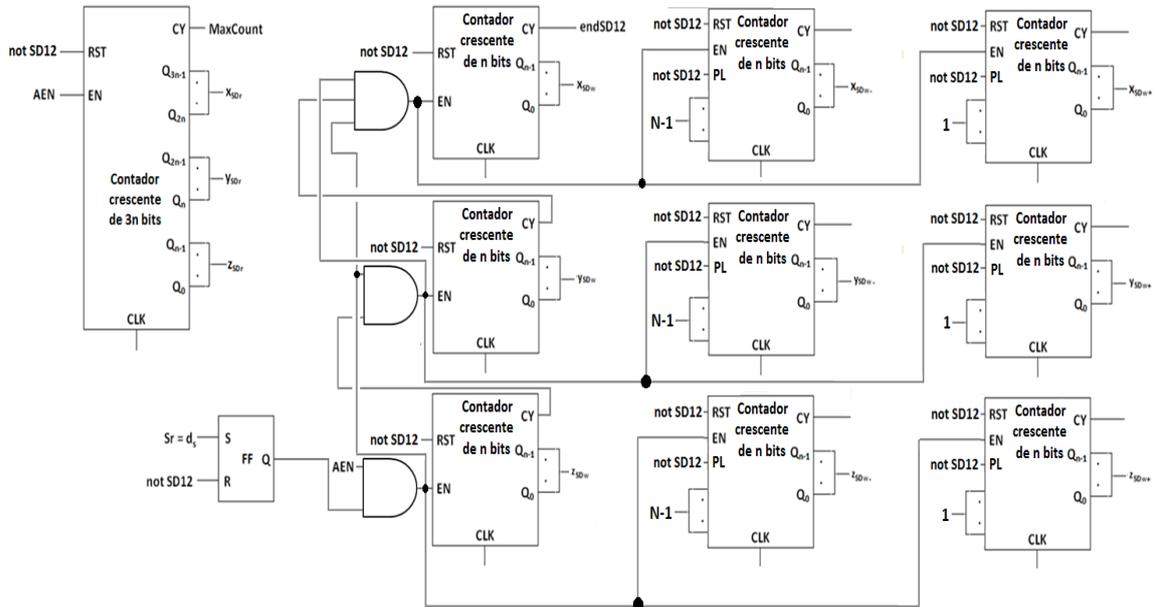


Fig. 39 – Alternativa ao conjunto de somadores e substractores.

Com este esquema de endereçamento concluir-se-á que esta nova etapa demora $N^3 + d_s$ ciclos de endereçamento.

Note-se que as etapas $D3$ e INI não sofreram alterações em termos de esquema de endereçamento nesta abordagem (vide Tabela I e Tabela III).

3.4.3. Drivers de endereçamento

O endereçamento dos diferentes bancos de memória é similar ao efectuado na abordagem inicial, isto é, baseia-se na utilização de catorze drivers de endereçamento, que operam em sincronismo. No entanto devido à optimização sofrida, os *drivers* de endereçamento vão consistir em *multiplexers 3:1*, cujo sinal de selecção provém da máquina de estados referida anteriormente (vide Fig. 37). De acordo com a etapa da máquina de estados, os *drivers* de endereçamento vão seleccionar o gerador de endereços correspondente (INI , $SD12$ ou $D3$).

As entradas de cada driver de endereçamento estão descritas na Tabela XIV. Tal como na abordagem inicial dos *drivers* de endereçamento, o símbolo 'X' significa que o endereçamento para os bancos de memória seleccionados é irrelevante: durante este período os bancos de memória encontram-se inactivos. Para as etapas correspondentes aos *wait states* introduzidos, o endereçamento é completamente irrelevante, pois durante estas etapas os elementos de armazenamento (bancos de memória e *buffers* de comunicação) estão inactivos.

Tabela XIV – Endereçamento de cada banco de memória em cada etapa na abordagem final.

		Eixo:		X				Y				Z				
		Destino:		→		←		↓		↑		•		⊗		
Etapas	Segmentos endereços	cfg	L _i /R _o	R _o /L _i	R _i /L _o	L _o /R _i	B _i /F _o	F _o /B _i	F _i /B _o	B _o /F _i	D _i /U _o	U _o /D _i	U _i /D _o	D _o /U _i	p	
INI	X	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	X _i	
	Y	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	Y _i	
	Z	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	Z _i	
SD12	X	X _{SDr}	X _{SDr}	X _{SDw-}	X _{SDr}	X _{SDw+}	X _{SDr}	X _{SDw}	X _{SDw}							
	Y	Y _{SDr}	Y _{SDr}	Y _{SDw}	Y _{SDr}	Y _{SDw}	Y _{SDr}	Y _{SDw-}	Y _{SDr}	Y _{SDw+}	Y _{SDr}	Y _{SDw}	Y _{SDr}	Y _{SDw}	Y _{SDw}	
	Z	Z _{SDr}	Z _{SDr}	Z _{SDw}	Z _{SDr}	Z _{SDw-}	Z _{SDr}	Z _{SDw+}	Z _{SDw}							
D3	x	X	0	X	N-1	X	h _{D13}	X	X							
	y	X	h _{D13}	X	h _{D13}	X	0	X	N-1	X	l _{D13}	X	l _{D13}	X	X	
	z	X	l _{D13}	X	l _{D13}	X	l _{D13}	X	l _{D13}	X	0	X	N-1	X	X	

3.4.4. Sinais de controlo

Na abordagem final o elemento que recebe a informação do módulo de *scattering* depende do endereço destino. Caso o endereço pertença a uma das faces do bloco a transferência é feita para o *buffer* local da direcção correspondente. Por sua vez se o endereço pertencer a um nó interior à malha a transferência é feita para os bancos de memória definidos como saída. Uma consequência desta abordagem consiste no aumento da complexidade dos sinais de escrita dos *buffers* locais relativamente à segunda abordagem, uma vez que estes estão dependentes dos endereços destino durante a etapa *SD12*. Esta dependência implica a existência de seis sinais de escrita, um por cada uma das direcções espaciais. Os sinais de controlo dos bancos de memória tiveram de contemplar um sinal adicional, *Inout* (gerado pela máquina de estados – vide Fig. 37), que permite diferenciar o tipo de memória na iteração em execução: banco de memória de saída ou entrada. Tendo em consideração as operações de transferência de informação realizadas em cada etapa e a estrutura do barramento de dados, verifica-se que cada um dos elementos de armazenamento (bancos de memória e *buffers* de comunicação) que participam nas operações de transferência de informação têm de ser controlados, tal como se encontra descrito na Tabela XV e Tabela XVII, assumindo que todos os sinais intervenientes são activos a alto.

Tal como nas abordagens anteriores, a divisão clara entre as diferentes etapas de execução foi algo de que não se abdicou com a optimização, e, como consequência, os sentidos dos fluxos de dados para as diferentes etapas permaneceram distintos. Deste modo, garantiu-se uma característica intrínseca da unidade do Meshotron: a inexistência de colisão de informação, como se pode observar na Fig. 35.

Tabela XV – Sinais de controlo dos bancos de memória de entrada/saída.

<i>Inout</i>	Etapas	R	W
0	INI	0	1
0	SD12	1	0
0	D3	0	0
1	INI	0	0
1	SD12	0	1
1	D3	0	1

Legenda:

Inout=‘0’ – Banco de memória de entrada;

Inout=‘1’ – Banco de memória de saída.

As condições impostas a cada sinal de escrita dos *buffers* locais estão descritas na Tabela XVI. Note-se que desta forma a etapa *D1* é efectuada durante a etapa *SD12*.

Tabela XVI – Sinais de controlo dos *buffers* locais.

Buffer local	Endereços	Sinais de controlo dos <i>buffers</i> locais
L _o	x=0	W_left=SD12 and WEN
R _o	x=N-1	W_right= SD12 and WEN
B _o	y=0	W_back= SD12 and WEN
F _o	y=N-1	W_front= SD12 and WEN
D _o	z=0	W_down= SD12 and WEN
U _o	z=N-1	W_up= SD12 and WEN

Relativamente aos *buffers* adjacentes, ao módulo de *scattering* e aos bancos *p* e *cfg* os seus sinais de controlo são semelhantes aos sinais de controlo nas abordagens anteriores. Os sinais dos *buffers* adjacentes estavam dependentes da etapa *D3* que permaneceu inalterada. O princípio de funcionamento no *scattering pass* do módulo de *scattering* e dos bancos *p* e *cfg* é o mesmo, mas em vez de ser executado na etapa *S* é executado na etapa *SD12*. Os sinais encontram-se descritos na Tabela XVII.

Tabela XVII – Sinais de controlo dos *buffers* adjacentes e do módulo de *scattering*.

Etapas	Módulo de <i>scattering</i>		Bancos de memória				<i>Buffers</i> adjacentes	
			cfg		p			
	R	W	R	W	R	W	R	W
INI	0	0	0	1	0	1	0	0
SD12	1	1	1	0	0	1	0	0
D3	0	0	0	0	1	0	1	0

Note-se que o gerador de *Clock Enables* não sofreu qualquer tipo de alteração (vide secção 3.2.4). Através da combinação da informação presente na Tabela XV e na Tabela XVII é possível obter as seguintes funções booleanas dos respectivos sinais de controlo, como se encontra demonstrado na Tabela XVIII.

Tabela XVIII – Expressões booleanas dos sinais de controlo da unidade do Meshotron na abordagem final.

Módulo de scattering	R=SD12 and REN	W=SD12 and WEN
Banco de memória cfg	R=SD12 and REN	W=INI and WEN
Banco de memória p	R=D3 and REN	W=(INI or SD12) and WEN
Banco de memória X	R=not(inout) and SD12 and REN	W=((INI and not(inout)) or (inout and (SD12 or D3)) and WEN)
Buffers adjacentes	R=D3 and REN	

3.4.5. Tempo de computação

Nesta abordagem, a duração da etapa *INI* não sofre alteração. A duração das restantes etapas (por iteração) está indicada na Tabela XIX.

Tabela XIX - Duração de cada etapa na abordagem final.

Etapas	Número de ciclos de endereçamento
SD12	N^3+d_s
D3	N^2

Pelas mesmas razões mencionadas na secção 3.2.5, a duração dos *wait states* (*W1* e *W2* – vide secção 3.2.2.1) não será contemplada; os cálculos tomarão como base, como nas secções 3.2.5 e 3.3.4, a unidade mais lenta.

Repetindo o procedimento das secções 3.2.5 e 3.3.4, a duração de uma iteração, t_m , é dada pela Eq. 27.

$$t_m = N^3 + d_s + N^2 = N^3 + N^2 + d_s \approx N^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 27}$$

Considerando k iterações, o tempo de computação total é dado por:

$$T_{RIR} = k(N^3 + N^2 + d_s) \approx kN^3 \text{ (ciclos de endereçamento)}. \quad \text{Eq. 28}$$

Como cada ciclo de endereçamento tem três ciclos de relógio (vide secção 3.2.4), e sendo f_{clk} a frequência de relógio (Hz), vem:

$$T_{RIR} = \frac{3k}{f_{clk}}(N^3 + N^2 + d_s) \approx \frac{3kN^3}{f_{clk}} \text{ (s)}. \quad \text{Eq. 29}$$

Para o caso particular de uma unidade com $N=8$, virá:

$$T_{RIR} = \frac{1536k}{f_{clk}} \text{ (s)}. \quad \text{Eq. 30}$$

Comparando a Eq. 23 (segunda abordagem) e a Eq. 26 (abordagem final) com a Eq. 19 (abordagem inicial), verifica-se um aumento de desempenho bastante significativo, com reduções do tempo de computação para cerca de 2/3 e 1/3, respectivamente.

Repetindo os cálculos efectuados na secção 2.3 com as novas estimativas temporais o tempo de computação de um nó por iteração é dado por:

$$t_{m1} = \frac{t_m}{N^3} = \frac{3N^3}{f_{clk}N^3} = \frac{3}{f_{clk}} \text{ (s)}, \text{ onde } f_{clk} \text{ é a frequência do sinal de relógio}. \quad \text{Eq. 31}$$

A paralelização proporcionada pelo Meshotron leva à existência de uma nova variável no cálculo do tempo de obtenção da *RIR*, o número de unidades do Meshotron em utilização. Como consequência este é dado por:

$$T_{RIR} = \frac{1}{(c\sqrt{3})^3} \left(\frac{V}{N_{um}} \right) RT_{60} f_s^4 t_{m1} \text{ (s)}, \text{ onde } N_{um} \text{ é o número de unidades.} \quad \text{Eq. 32}$$

Assumindo uma $f_{clk}=27$ MHz para a unidade mais lenta; um volume $V=10000$ m³; um tempo de reverberação $RT_{60}=1.5$ s; uma frequência de amostragem $f_s = 44,1$ KHz e 1000 unidades o tempo necessário para obter a RIR, T_{RIR} , é cerca de:

$$T_{RIR} = \frac{1}{(c\sqrt{3})^3} \left(\frac{V}{N_{um}} \right) RT_{60} f_s^4 t_{m1} = \frac{1}{(c\sqrt{3})^3} \left(\frac{V}{N_{um}} \right) RT_{60} f_s^4 \left(\frac{3}{f_{clk}} \right) = 8,27 \text{ horas.} \quad \text{Eq. 33}$$

Como é visível a paralelização garante uma redução bastante significativa do tempo necessário para obter a RIR. Note-se que quanto maior o número de unidades em utilização maior a redução, logo a existência de tempos aceitáveis passa pela utilização de um grande número de unidades.

Capítulo 4 Modelação, simulação, síntese e depuração

4.1. Introdução

O desenvolvimento do Meshotron passou por três fases: modelação e simulação, síntese e implementação, com o objectivo de especificar alguns dos parâmetros do Meshotron, como por exemplo o número óptimo de unidades de *scattering* e o número de *bits* do barramento de dados, e validar o seu funcionamento. A validação do funcionamento baseou-se na comparação de respostas impulsivas (*RIR*) de um conjunto de modelos criados por ferramentas distintas (*VHDL* e *Matlab*) nas mesmas condições de funcionamento. Para tal é necessário um programa de modelação *DWM-3D* que gere os mesmos modelos e que seja utilizado como termo de comparação. Os programas criados em *Matlab* foram utilizados para esse efeito.

Na primeira fase foi efectuada a simulação comportamental de um conjunto de modelos criados em *VHDL* e comparada a *RIR* de cada um dos modelos em condições de funcionamento específicas com a *RIR* obtida através do programa de modelação *DWM-3D* criado em *Matlab*.

Na segunda fase foi efectuado um estudo dos recursos consumidos e da frequência máxima de funcionamento da unidade de *scattering*, de modo a determinar o número óptimo a utilizar na unidade do Meshotron. Este estudo é fundamental para a implementação da unidade do Meshotron na *FPGA*.

Na terceira fase é efectuada a implementação de um conjunto de modelos descritos em *VHDL* na *FPGA* e comparada a *RIR* extraída pela ferramenta de depuração em utilização com a obtida pelo programa de modelação *DWM-3D* em *Matlab*.

Note-se que a comparação baseia-se na verificação da existência de alguma diferença entre as *RIR* obtidas por ferramentas distintas.

4.2. Modelação e simulação

Com o objectivo de criar um modelo sintetizável e simulável do Meshotron foi realizada a sua descrição em *VHDL*. Nesta primeira fase é efectuada a simulação e modelação de um conjunto de modelos acústicos descritos em *VHDL*. Esta divide-se em duas fases: uma primeira fase em que se efectua um estudo do impacto do número de *bits* na precisão numérica e uma segunda fase que se baseia na simulação comportamental de um conjunto de modelos criados em *VHDL* e na comparação da *RIR* obtida com a *RIR* obtida pelos mesmos modelos criados pelo programa de modelação *DWM-3D* criado em *Matlab* em condições similares de funcionamento.

O objectivo da primeira fase passou pela determinação do número óptimo de *bits* do barramento de dados que minimizam o erro relativo e maximizam a relação sinal-ruído. Para tal utilizou-se o programa de modelação *DWM-3D* criado em *Matlab* e compararam-se as *RIRs* de modelos com uma unidade do Meshotron obtidas em grandezas de 16 e 32 *bits* com a *RIR* obtida em *double*. A comparação baseou-se na análise do erro e da relação sinal-ruído associada a cada uma das grandezas inteira com diferente número de *bits*.

Com o número de *bits* do barramento de dados definido procedeu-se à simulação comportamental de um conjunto de modelos criados em *VHDL* com: 1, 2, 4 e 8 unidades

do Meshotron. Para efeitos de validação extraíram-se as *RIRs* obtidas para cada modelo e compararam-se com as *RIRs* obtidas pelo programa de modelação *DWM-3D* nas mesmas condições de funcionamento. O procedimento nesta fase está ilustrado na Fig. 40.

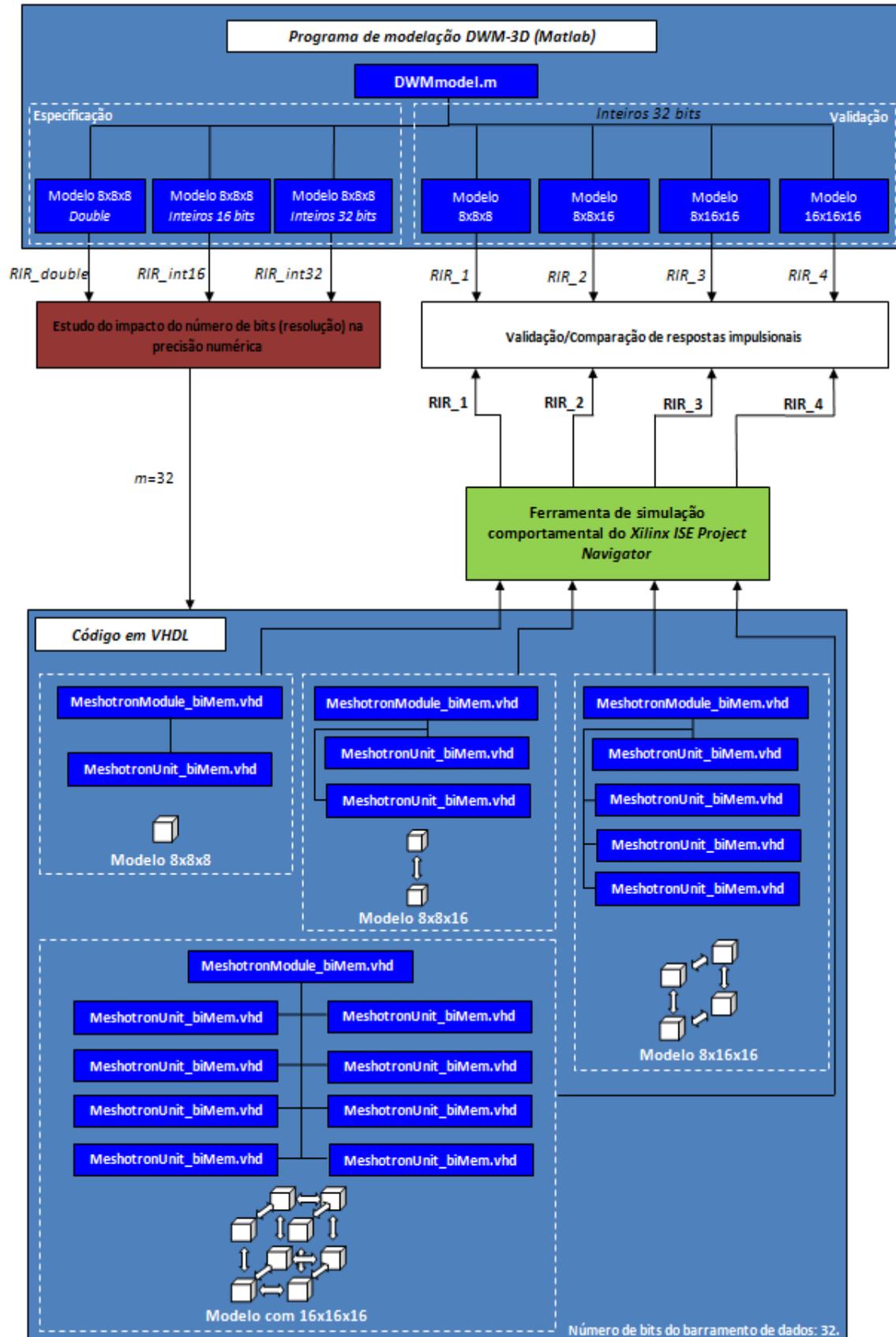


Fig. 40 – Procedimento de simulação.

Com base nas descrições em *VHDL* desenvolvidas de todos os elementos especificados nas secções anteriores, o protótipo da unidade do Meshotron foi simulado a partir da ferramenta de simulação da *Xilinx*.

A excitação de cada modelo partiu da injeção de um impulso de *Dirac* de amplitude P na primeira iteração do Meshotron. O processo pode ser efectuado através da inicialização dos bancos de memória de entrada do nó, estipulado previamente como origem, com o valor $P/2$. O valor de inicialização P pode ser aumentado para aumentar a precisão numérica das operações efectuadas ao nível de cada unidade do Meshotron, desde que o *overflow* na unidade do Meshotron seja evitado. Contudo o processo de inicialização não constituiu uma preocupação primária o que levou à escolha de valores de inicialização conservadores: $P=3 \times 10^8$ e $P=4 \times 10^3$ respectivamente para os testes e as simulações com barramento de dados de $m=32$ e $m=16$.

Outro parâmetro a especificar consistiu no coeficiente de reflexão das paredes à volta dos modelos criados. Por uma questão de simplicidade optou-se por simular paredes com reflexão total ($R=1$ na Eq. 6). Deste modo, as saídas de cada *buffer* que estabelecem comunicação com a face do modelo sem qualquer tipo de blocos adjacentes como vizinhos foram ligadas novamente às entradas dos nós de superfície. Isto é, a informação transferida durante do estado *D3* consiste na mesma informação que o mesmo nó transferiu para os *buffers* de comunicação no estado *D1*. Note-se que esta operação é completamente transparente para as diversas unidades; esta operação é exactamente a mesma independentemente da posição na rede do Meshotron.

4.2.1. Estudo do impacto do número de *bits* na precisão numérica

A escolha do número de *bits* do barramento de dados baseou-se na utilização do programa de modelação *DWM-3D* criado em *Matlab* (vide Apêndice D). Este permitiu analisar o erro relativo e a relação sinal-ruído da *RIR* obtida em 16 e 32 *bits*. O erro relativo foi calculado através da diferença entre a *RIR* obtida em formato inteiro (16 e 32 *bits*) e em formato *double*.

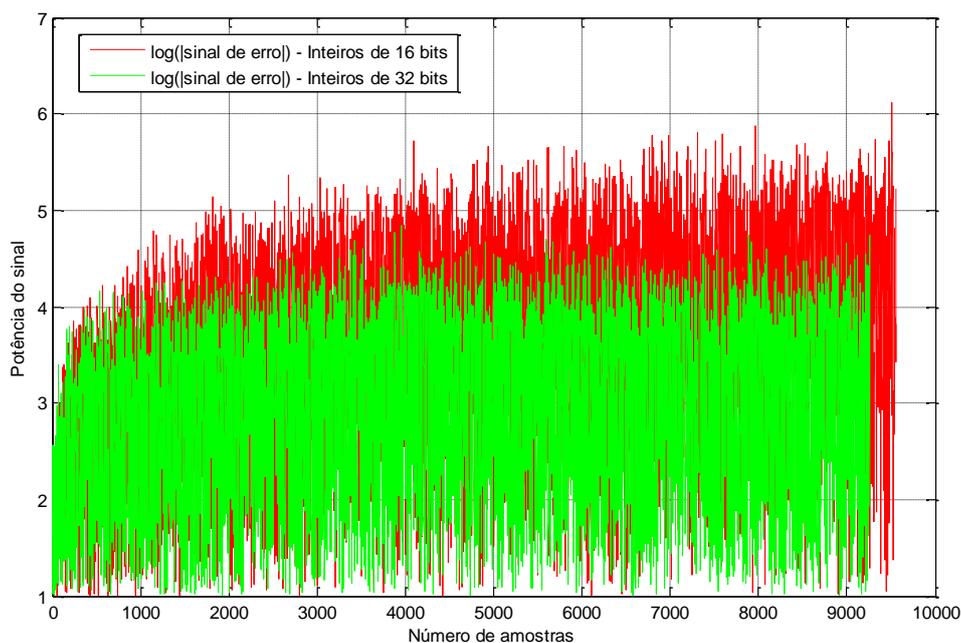


Fig. 41 – Erro existente entre a *RIR* em formato *double* e em formato inteiro de 32 e 16 *bits*.

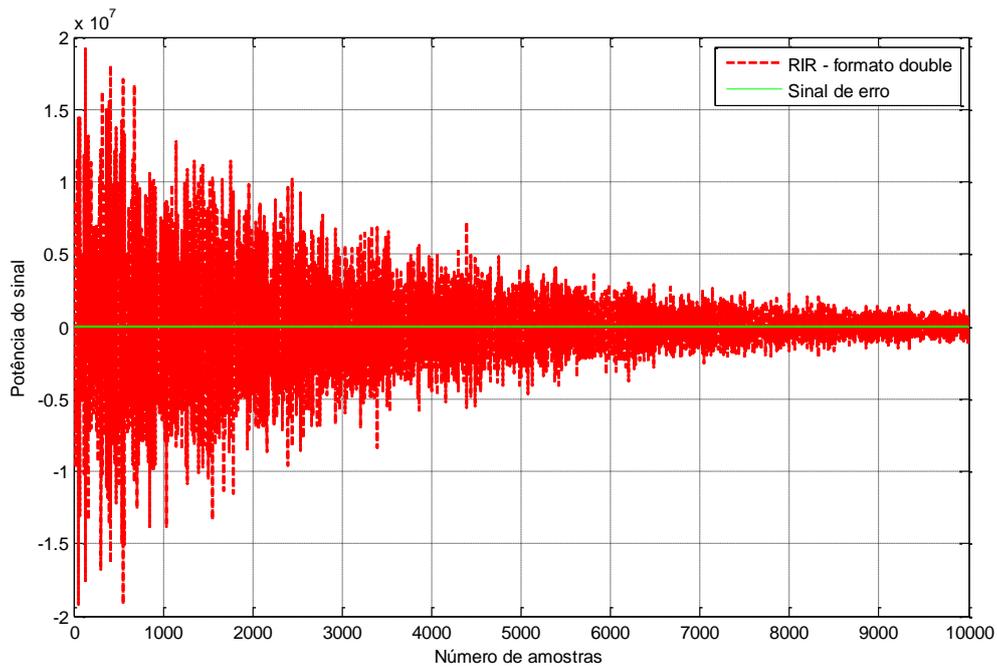


Fig. 42 - Evolução do sinal de erro e da RIR com o número de amostras para grandezas de 32 bits.

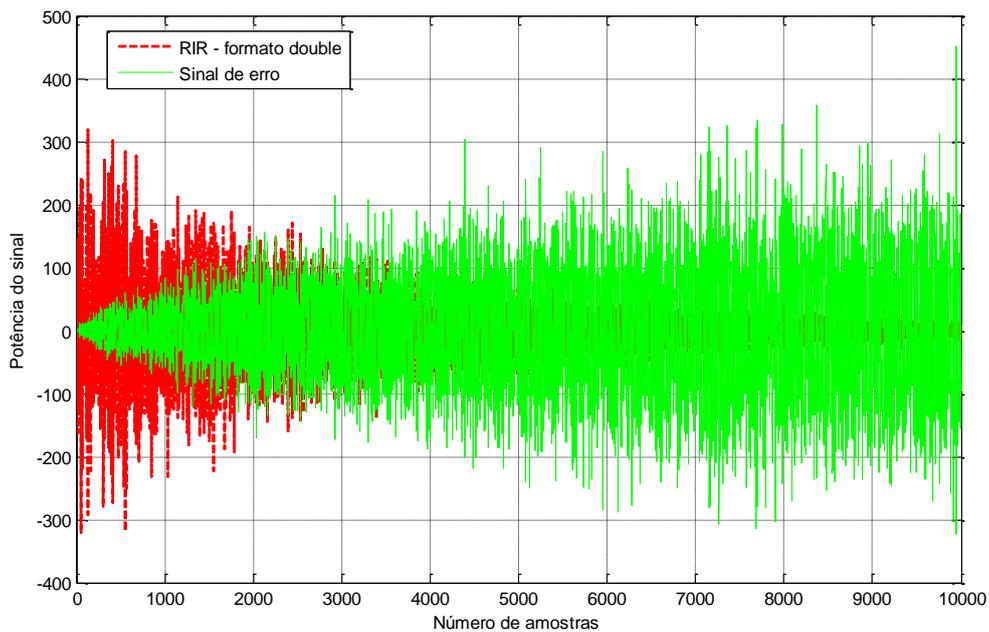


Fig. 43 - Evolução do sinal de erro e da RIR com o número de amostras para grandezas de 16 bits.

Tabela XX – Relação sinal ruído para grandezas de 16 e 32 bits.

Número de bits em utilização	SNR (dB)
16	-4,9
32	98,7

Note-se que o logaritmo do sinal de erro em 16 bits começa a ultrapassar o logaritmo do sinal de erro em 32 bits a partir da iteração 1000 (vide Fig. 41). Para além disso a diferença entre a relação sinal-ruído para as duas grandezas é extremamente significativa (vide Tabela XX), para o caso das grandezas de 16 bits esta relação é negativa o que impossibilita a sua utilização. Como consequência por análise dos resultados obtidos

conclui-se que o erro relativo é minimizado e a relação sinal-ruído maximizada quando são utilizadas grandezas de 32 *bits* (vide Fig. 41, Fig. 42, Fig. 43 e a Tabela XX).

4.2.2. Modelo com uma unidade do Meshotron

Este modelo foi sujeito a dois tipos de simulação que visaram validar o modelo em termos de resposta impulsional e determinar qual a relação entre o número de unidades de *scattering* e o número de ciclos de endereçamento.

4.2.2.1. Resposta impulsional

O modelo a ser simulado apresenta uma unidade com $N=8$ (512 blocos de nós – vide Fig. 44). O nó origem foi definido na coordenada (4,4,4), que consiste grosseiramente no centro do bloco, o nó receptor foi definido como o nó do canto inferior na coordenada (0,0,0). O processo de simulação foi repetido para uma coordenada do nó receptor aleatória, respectivamente a coordenada (1,2,3). Simulações análogas às descritas foram efectuadas para $N=16$ (4096 nós por bloco), mas neste caso o nó origem foi colocado na coordenada (8,8,8). Em todos os casos foram retiradas e comparadas 200 amostras das respostas impulsionais do modelo em *VHDL* com os resultados obtidos através do programa de modelação *DWM-3D* em *Matlab* em condições de funcionamento idênticas. Os resultados foram exactamente os mesmos para todos os exemplos.

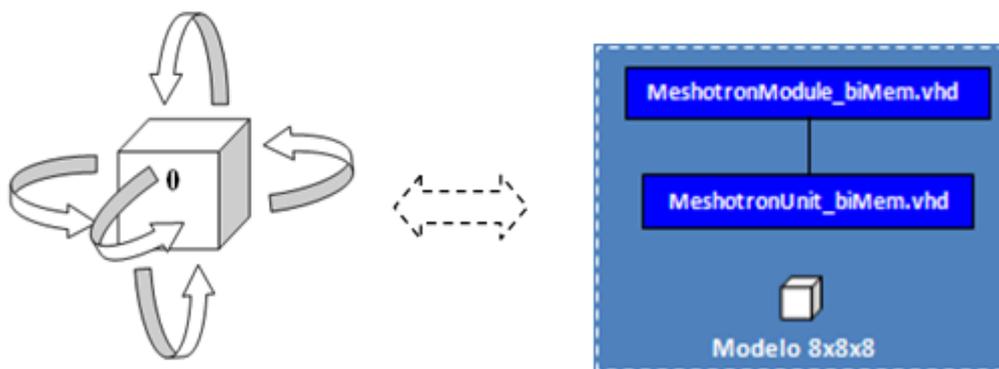


Fig. 44 – Modelo com uma unidade do Meshotron.

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo simulado dever-se-á consultar o Apêndice C e o Apêndice D.

4.2.2.2. Tempo máximo de cálculo da unidade de *scattering*

Durante a simulação efectuada os atrasos temporais associados a cada elemento da unidade do Meshotron não foram tidos em conta, devido à análise efectuada ser meramente comportamental. Como consequência o tempo necessário para a unidade de *scattering* completar uma operação de cálculo foi resumido inteiramente aos registos de entrada e saída deste elemento - dois ciclos de relógio (um ciclo para escrita dos valores na unidade de *scattering* e um ciclo para leitura dos valores calculados da unidade de *scattering*). Como cada ciclo de endereçamento corresponde a três ciclos de relógio, uma única unidade de *scattering* é suficiente para garantir a correcta execução da operação de cálculo. Como consequência foram efectuadas simulações adicionais, de modo a demonstrar que o número de unidades de *scattering*, d_s , deve ser no mínimo igual ao número de ciclos de

endereçamento executados numa operação de cálculo do módulo de *scattering*, como é observável na secção 3.2.1.4. Para este fim a operação de cálculo da unidade de *scattering* foi atrasada artificialmente através da introdução de um conjunto extra de registos (*dummy registers*). A segunda coluna da Tabela XXI consiste no limite superior a partir do qual o cálculo da resposta impulsional do modelo criado começa a falhar em função do número de unidades de *scattering* existentes.

Tabela XXI – Duração total admissível da operação de *scattering*.

Número de unidades de <i>scattering</i> , d_s	Número máximo de registos de atraso	Duração total (ciclos de relógio)
1	1	$1+2=3=3d_s$
2	4	$4+2=6=3d_s$
4	10	$10+2=12=3d_s$
8	22	$22+2=24=3d_s$

4.2.3. Modelo com duas unidades do Meshotron

O segundo modelo simulado apresenta duas unidades do Meshotron (vide Fig. 45), cada uma com 512 nós ($N=8$), interligadas entre si através dos *buffers* de comunicação. Note-se que as duas unidades do Meshotron foram sujeitas a sinais de relógio distintos, de modo a verificar o correcto funcionamento do mecanismo de sincronização entre unidades adjacentes.

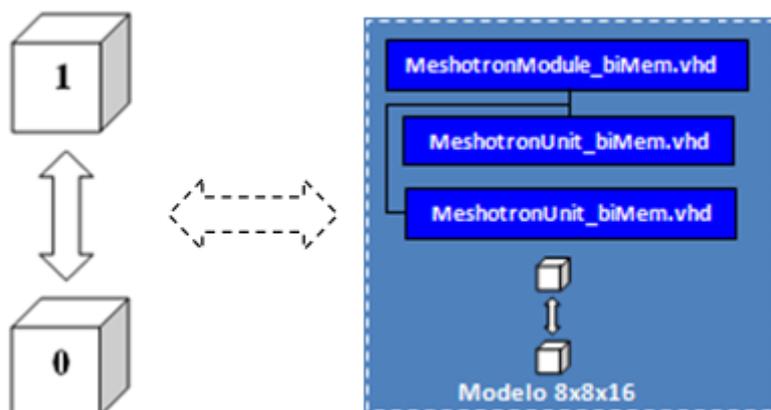


Fig. 45 – Modelo com duas unidades do Meshotron.

Durante a simulação deste modelo o nó emissor foi colocado, numa primeira fase, na coordenada (4,4,4) da unidade 0 e o nó receptor nas coordenadas (1,1,1) e (4,1,5) da unidade 1. Numa segunda fase o nó emissor foi colocado na coordenada (4,4,4) da unidade 1 e o nó receptor nas coordenadas (0,1,0) e (1,5,6) da unidade 0. Por comparação com os valores da *RIR* nas mesmas condições de funcionamento obtida pelo programa de modelação *DWM-3D* em *Matlab* verifica-se que os resultados são exactamente os mesmos independentemente do sinal de relógio aplicado a cada uma das unidades. Deste modo demonstra-se que o mecanismo de sincronização entre as duas unidades está a operar correctamente.

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo simulado dever-se-á consultar o Apêndice C e o Apêndice D.

4.2.4. Modelo acústico com quatro unidades do Meshotron

O terceiro modelo simulado apresenta quatro unidades do Meshotron (vide Fig. 46), cada uma com 512 nós ($N=8$), interligadas entre si a partir dos respectivos *buffers* de comunicação. Cada unidade do modelo foi sujeita a sinais de relógio distintos. Para além disso neste modelo cada unidade estabelece comunicação por duas direcções distintas o que aumenta a complexidade da gestão das operações de transferência de informação entre unidades adjacentes. O papel das condições de entrada e saída nos *wait states* torna-se mais evidente.

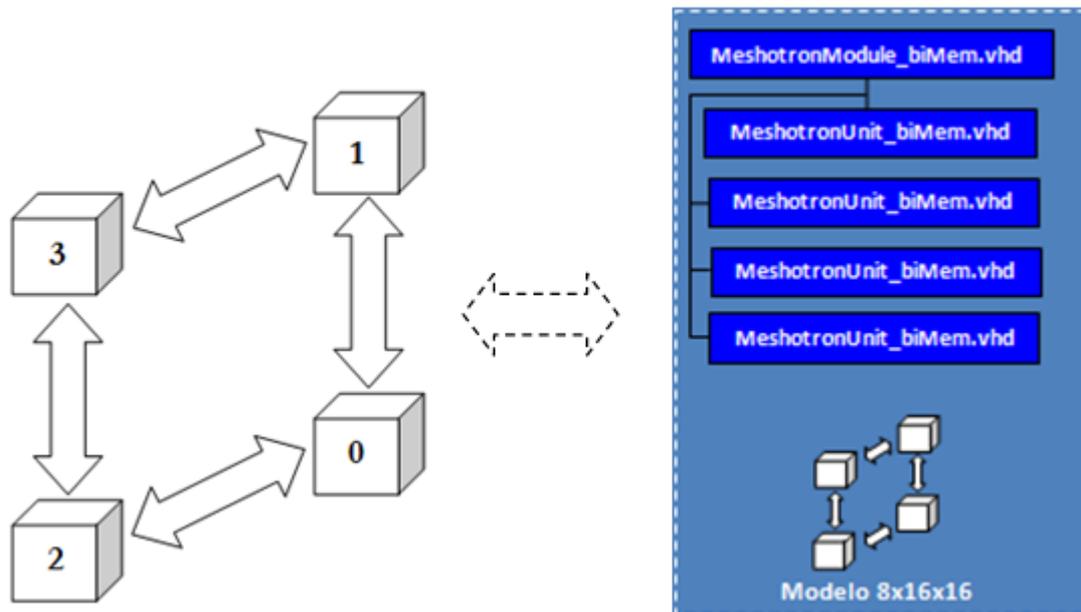


Fig. 46 – Modelo com quatro unidades do Meshotron.

A validação do modelo passou pela recolha de 11 amostras da *RIR* obtida para três posições distintas. O nó emissor ficou definido na coordenada (1,2,1) da unidade 0 e o nó receptor nas coordenadas: (5,5,5) da unidade 1, (1,2,0) da unidade 2 e (7,0,1) da unidade 3. Por comparação com os valores da *RIR* obtidos no mesmo modelo sujeito às mesmas condições de funcionamento no programa de modelação *DWM-3D* criado no *Matlab* verifica-se que os resultados são exactamente os mesmos independentemente do sinal de relógio aplicado. Logo concluir-se-á que o mecanismo de sincronização entre unidades do Meshotron adjacentes está a operar correctamente apesar do aumento da complexidade do modelo.

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo simulado dever-se-á consultar o Apêndice C e o Apêndice D.

4.2.5. Modelo com oito unidades do Meshotron

O terceiro modelo apresenta oito unidades do Meshotron (vide Fig. 47), cada uma com 512 nós ($N=8$). O objectivo desta simulação é comparar o modelo de oito unidades (criado em *VHDL*) com o modelo de uma unidade (criado em *Matlab*) com o mesmo número de nós existentes no modelo com oito unidades – vide Fig. 48.

Neste modelo cada unidade do Meshotron estabelece a comunicação com as suas vizinhas por três direcções distintas. Logo existe um aumento da complexidade na gestão

das operações de transferência de informação. Foram aplicados diferentes sinais de relógio para as oito unidades do Meshotron.

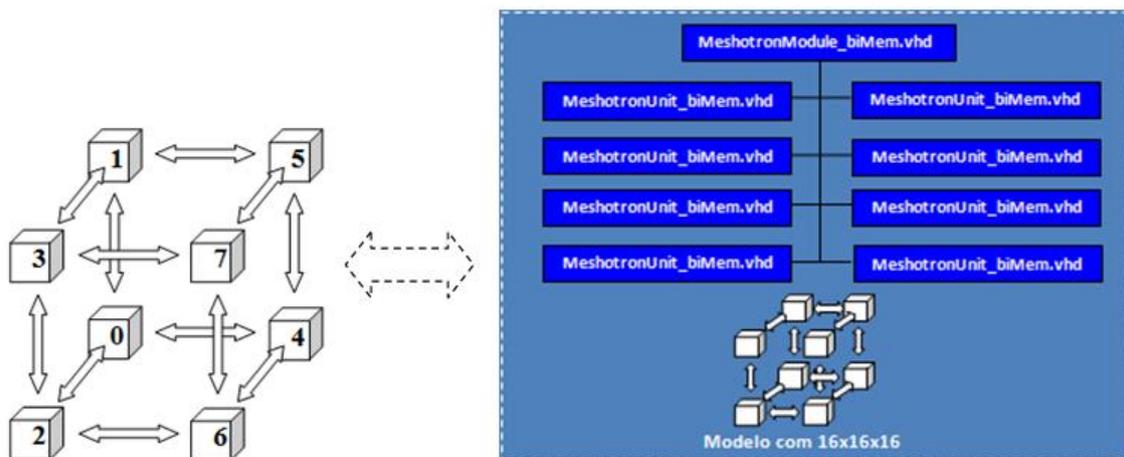


Fig. 47 – Modelo com oito unidades do Meshotron.

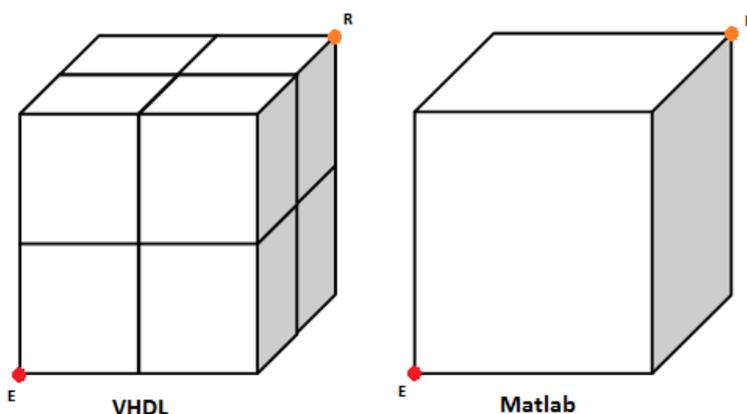


Fig. 48 – Comparação entre o modelo criado em *VHDL* com o modelo criado em *Matlab*.

Legenda:

- - Nó emissor (posição meramente exemplificativa);
- - Nó receptor (posição meramente exemplificativa).

A validação deste modelo passou pela colocação do nó emissor na coordenada (0,0,0) da unidade 0 e o nó receptor na coordenada (0,0,0) da unidade 7. Seguidamente comparou-se a *RIR* do modelo em *VHDL* com a do modelo com 4096 nós ($N=16$) obtida através do programa de modelação *DWM-3D* sujeito às mesmas condições de funcionamento (nó receptor e emissor geometricamente nas mesmas posições como exemplificado na Fig. 48). Ambos os modelos criados são geometricamente iguais, uma vez que o terceiro modelo consiste num bloco de 4096 nós resultante da interligação de oito unidades de 512 ($N=8$) nós.

Durante todas as simulações os sinais de relógio estiveram na gama do 1 a 3 ns. A escolha desta gama de sinais de relógio no processo de simulação deve-se a uma questão de facilitar a visualização dos sinais. Como era expectável, os *wait states* (*W1* e *W2* – vide Fig. 22 e Fig. 37) foram introduzidos nas unidades com os sinais de relógio mais rápidos, o que levou a que a simulação estivesse dependente da unidade mais lenta (com o período de 3 ns).

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo simulado dever-se-á consultar o Apêndice C e o Apêndice D.

4.2.6. Análise dos resultados obtidos

O estudo do impacto do número de *bits* na precisão numérica baseou-se na comparação das *RIRs* obtidas em grandezas de 16 e 32 *bits* com as obtidas em formato *double*. Por análise dos resultados obtidos depreende-se que o sinal de erro é menos significativo, quando comparado com a ordem de grandeza da *RIR*, em grandezas de 32 *bits* (vide Fig. 41, Fig. 42 e Fig. 43). Note-se que o sinal de erro quando a *RIR* é obtida em 16 *bits* começa a ultrapassar em termos de potência a *RIR* a partir da iteração 3000 (vide Fig. 42). Por análise da relação sinal-ruído verifica-se que a diferença entre grandezas de 32 e 16 *bits* é significativa (vide Tabela XX), cerca de 113 dB. Como consequência de modo a não limitar o algoritmo de modelação do Meshotron em termos de iterações e a minimizar o sinal de erro optou-se por utilizar o barramento de dados em 32 *bits*.

A simulação do conjunto de modelos descritos em *VHDL* (de 1, 2, 4 e 8 unidades do Meshotron) permite depreender que o Meshotron está a operar correctamente em termos meramente comportamentais, uma vez que as *RIRs* obtidas por simulação foram exactamente iguais às *RIRs* obtidas pelos mesmos modelos nas mesmas condições de funcionamento pelo programa de modelação *DWM-3D*.

4.3. Síntese

A segunda fase baseou-se num estudo dos recursos e da frequência máxima de funcionamento a unidade do Meshotron. Note-se que estes dependem do número de unidades de *scattering* existentes, logo foi imperativo o estudo deste elemento para determinar qual número óptimo a utilizar que garante o melhor desempenho da unidade do Meshotron com o menor número de recursos em utilização. Devido ao facto do código do Meshotron ser pesado em termos de recursos utilizou-se uma *FPGA Virtex 5, device XC5VLX110T e package FF1136*.

4.3.1. Recursos e frequência máxima de funcionamento da unidade de *scattering*

O primeiro teste baseou-se na análise dos recursos consumidos e da frequência máxima de funcionamento da unidade de *scattering*. Note-se que a unidade de *scattering* efectua os cálculos que permitem resolver a equação de onda, logo é um elemento relevante no funcionamento do Meshotron (vide Eq. 5 e Eq. 13). Os resultados obtidos estão representados na Tabela XXII.

Tabela XXII – Recursos consumidos e frequência máxima de funcionamento da unidade de *scattering*.

		Número de recursos utilizados	Taxa de utilização (%)
Recursos consumidos	Número de <i>Flips-Flops</i>	478	1
	Número de <i>Slices</i> de LUTs	571	1
	Número de <i>Route-thru</i>	37	
	Número de <i>Slices</i> ocupadas	243	1
	Número de pares de LUTs <i>flip-flops</i> utilizadas	776	
	Número de <i>input/output buffers</i>	423	1
	Número de <i>BUFG/BUFGCTRLs</i>	1	1
	Número de <i>DSP48Es</i>	4	6
Frequência máxima de funcionamento (MHz)		57,995	

4.3.2. Variação da frequência máxima de funcionamento em função do número de unidades de *scattering*

O segundo teste analisou a variação da frequência máxima de funcionamento da unidade do Meshotron com o número unidades de *scattering* existentes. Este teste visa a optimização da unidade do Meshotron, logo o objectivo será determinar o número de unidades de *scattering* que maximiza a frequência máxima de funcionamento. Note-se que devido à utilização de um *demultiplexer* e um *multiplexer*, que efectuam o controlo do fluxo de entrada e saída de dados no módulo de *scattering*, o número de unidades de *scattering* existente tem de ser potência de 2 (vide secção 3.2.1.4).

A variação da frequência máxima em função do número de unidades de *scattering* existente encontra-se representada na Tabela XXIII.

Tabela XXIII – Frequência máxima de funcionamento em função do número de unidades de *scattering*.

Número de unidades de <i>scattering</i>	Frequência máxima (MHz)
1	57,773
2	57,773
4	57,773
8	57,773
16	57,995
32	57,773
64	57,773

Note-se que não existem variações significativas da frequência máxima de funcionamento com o número de unidades de *scattering* (este parâmetro mantém-se constante). Logo concluir-se-á que este parâmetro não vai ser um elemento condicionante na escolha do número de unidades de *scattering* a utilizar.

4.3.3. Caminho crítico

O terceiro teste visou a determinação do principal caminho crítico da unidade do Meshotron em função do número de unidades de *scattering* (1, 2, 4, 8, 16 unidades de *scattering*). Através de uma aplicação da ferramenta de compilação da *Xilinx (PlanAhead)* verifica-se que o principal caminho crítico está associado aos detectores de *overflow* na unidade de *scattering* (vide Fig. 50). De forma a observar-se de forma mais perceptível a variação do atraso total no principal caminho crítico, associado a um dos detectores de *overflow*, construiu-se o gráfico da sua variação em função do número de unidades de *scattering* (vide Fig. 49).

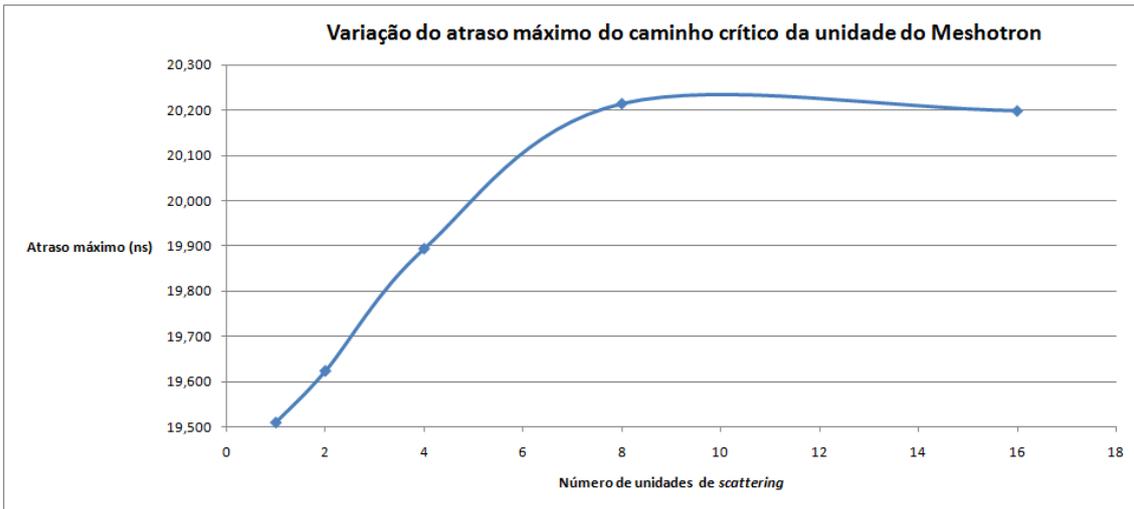


Fig. 49 - Gráfico de variação do atraso total do principal caminho crítico em função do número de unidades de scattering.

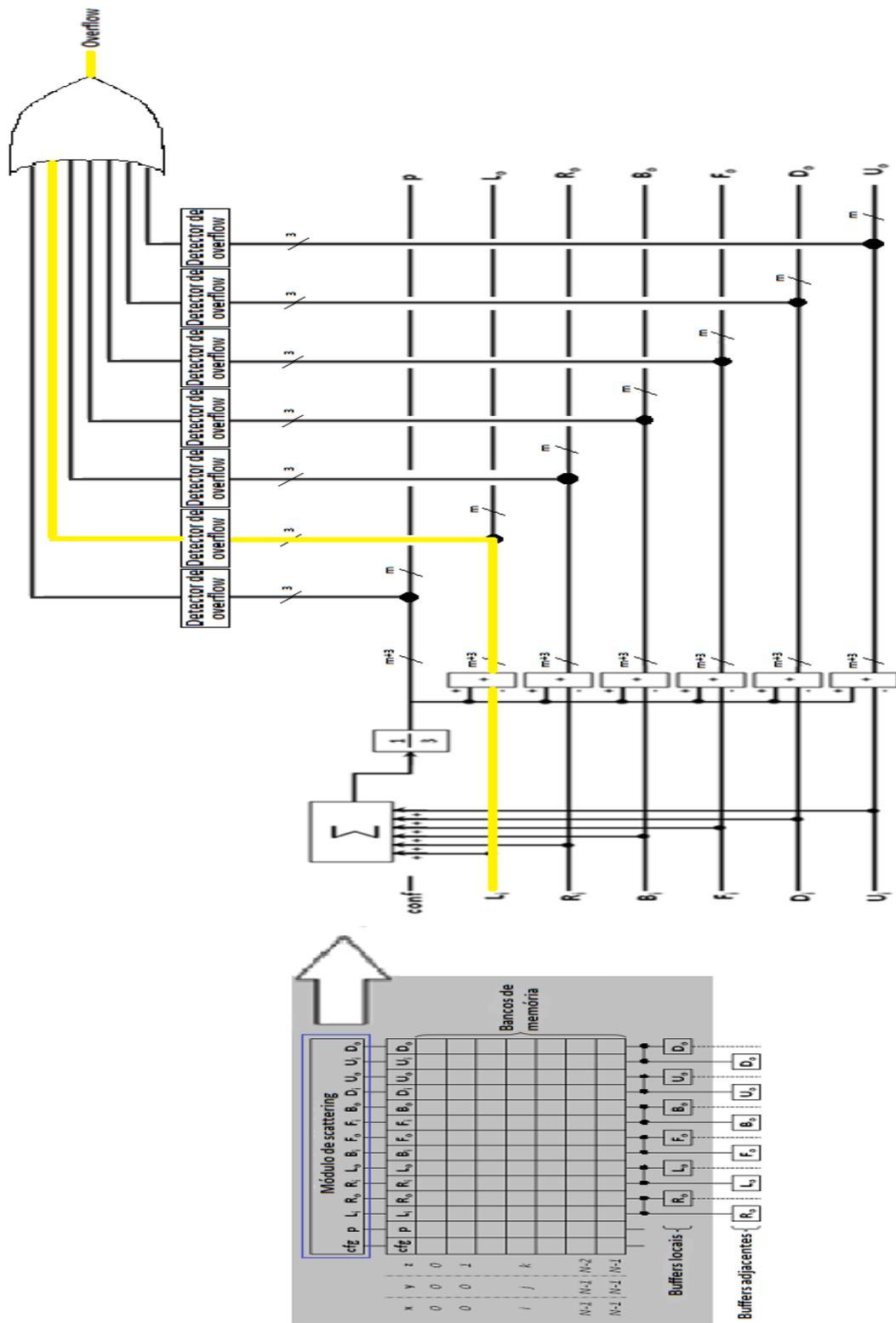


Fig. 50 – Principal caminho crítico da unidade do Meshotron.

Legenda:
 - *Caminho crítico.*

O atraso máximo está associado aos blocos de detecção de *overflow* e é mínimo quando a unidade do Meshotron apresenta uma unidade de *scattering*.

Logo com base nos testes ao nível de recursos e caminhos críticos concluir-se-á que o número óptimo de unidades de *scattering* a utilizar que permite minimizar estes parâmetros é uma unidade de *scattering*. A variação da frequência máxima de funcionamento não foi tida em conta, uma vez que este parâmetro permanece constante.

Note-se que existe perfeita consciência que o detector de *overflow* está a prejudicar a frequência máxima de funcionamento, que se deve à cascata de portas lógicas implementada por este detector (vide Tabela XXIV).

Tabela XXIV – Frequência máxima de funcionamento da unidade do Meshotron com o número óptimo de unidades de *scattering* com e sem detecção de *overflow*.

	Frequência máxima de funcionamento (MHz)
Com detecção de <i>overflow</i>	57,773
Sem detecção de <i>overflow</i>	65,802

Uma solução possível para o problema passava pela colocação passaria pela colocação de nas entradas do circuito combinatório do detector de *overflow*, como se encontra representado na Fig. 51.

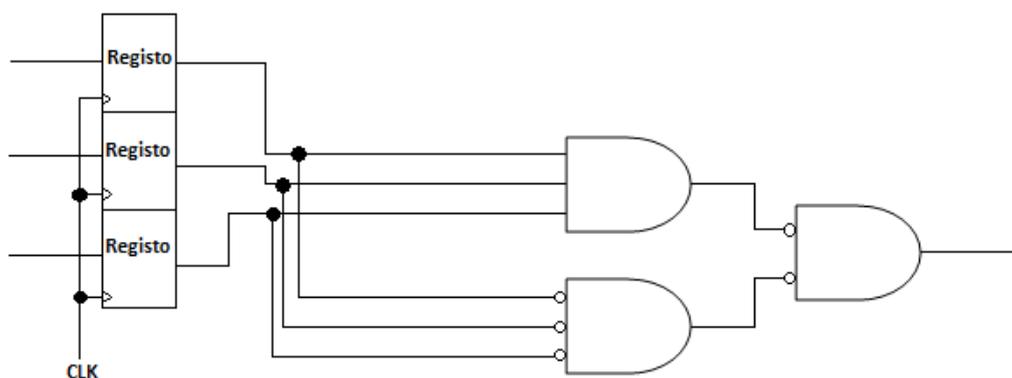


Fig. 51 – Circuito combinatório do detector de *overflow* com as entradas registradas.

Para os diferentes testes efectuados nas seguintes secções manteve-se a unidade de detecção de *overflow* sem as entradas registradas.

4.3.4. Recursos e frequência máxima de funcionamento da unidade do Meshotron

O último teste baseou-se na análise dos recursos consumidos e da frequência máxima de funcionamento para uma unidade do Meshotron com um número de unidades de *scattering* óptimo determinado na secção 4.3.3. A unidade do Meshotron foi definida com os parâmetros especificados na secção 4.3.3 e com $N=8$ – 512 nós por bloco. Como consequência obtiveram-se os resultados apresentados na Tabela XXV.

Tabela XXV – Recursos consumidos e frequência máxima de funcionamento de uma unidade do Meshotron.

		Número de recursos utilizados	Taxa de utilização (%)
Recursos consumidos	Número de <i>Slices</i> de Registos	1619	2
	Número de <i>Slices</i> de LUTs	5549	8
	Número de <i>Route-thrus</i>	88	
	Número de <i>Slices</i> ocupadas	1775	10
	Número de pares de LUTs de <i>flip-flops</i> utilizadas	5895	
	Número de <i>input/output buffers</i>	531	82
	Número de <i>BlockRAM/FIFO</i>	6	4
	Número de <i>BUFG/BUFGCTRLs</i>	7	2
	Número de <i>DSP48Es</i>	4	6
Frequência máxima de funcionamento (MHz)		57,773	

4.4. Implementação e depuração

A especificação do número óptimo de unidades de *scattering* na unidade do Meshotron permitiu a sua implementação na *FPGA*. Para efeitos de teste foram utilizados dois modelos com 1 e 2 unidades do Meshotron. A validação dos modelos criados passou pela recolha e comparação da *RIR* obtida pela ferramenta de depuração com a *RIR* obtida pelo programa de modelação *DWM-3D* quando sujeitas a condições de precisão numérica iguais e distintas. Quando sujeitas à mesma precisão numérica é expectável verificar que a *RIR* obtida é igual. No entanto quando sujeitas a condições de precisão numérica distintas espera-se observar uma diferença pouco significativa entre as *RIRs* obtidas pelo programa de modelação e pela ferramenta de depuração.

Note-se que devido ao seu grau de complexidade o modelo com duas unidades foi sujeito a dois tipos de teste:

- O primeiro teste baseou-se na colocação de diferentes sinais de relógio em cada uma da unidade, de modo a validar o funcionamento do mecanismo de sincronização entre unidades adjacentes. Este teste visa demonstrar que a *RIR* é independente do sinal de relógio aplicado na unidade do Meshotron.
- O segundo teste baseou-se na variação da coordenada do nó emissor.

A Fig. 52 ilustra o procedimento efectuado nesta fase.

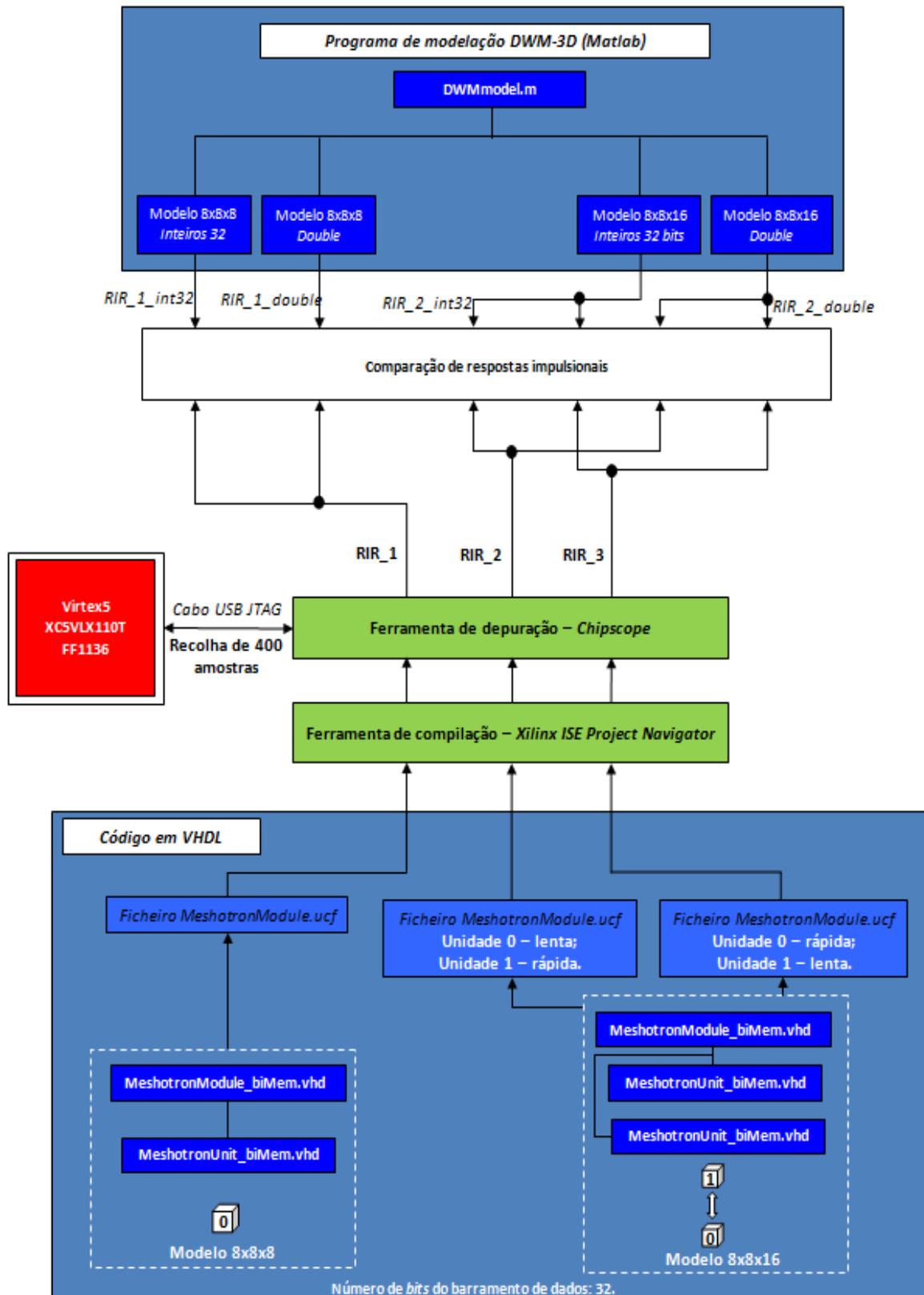


Fig. 52 – Procedimento de implementação.

A ferramenta de depuração utilizada na recolha dos sinais de saída de pressão acústica consistiu no *ChipScope*. Note que a implementação do código em *VHDL* implicou a selecção dos pinos de entrada e saída a utilizar (definição do *Constraints file*). Durante a

selecção dos pinos de entrada e saída da *FPGA* optou-se, durante a fase de síntese executada pelo compilador da *Xilinx*, por manter a estrutura hierárquica da descrição do Meshotron, de modo a que os sinais pretendidos fossem observáveis. Como consequência os modelos criados não foram optimizados ao máximo pelo compilador da *Xilinx* o que levou a uma redução significativa da frequência máxima de funcionamento da unidade do Meshotron ($f_{max} = 40,942$ MHz).

4.4.1. Configurações iniciais

A validação dos diferentes modelos implicou a definição dos sinais de entrada e saída, entre estes destacam-se o sinal de relógio e o sinal de *trigger* (sinal que controla recolha de amostras), e a definição do número de amostras a recolher.

O sinal de *trigger* consistiu no sinal de leitura do banco de memória *p*. A selecção do sinal de relógio efectuou-se com base na frequência máxima de funcionamento da unidade do Meshotron. Entre os sinais de relógio globais fornecidos pela *FPGA* apenas dois proporcionavam uma margem considerável ($f_{relógio\ 1} = 33$ MHz e $f_{relógio\ 2} = 27$ MHz [20]) que tinha em conta o atraso máximo existente na unidade do Meshotron, $f_{max} = 40,942$ MHz. A margem associada a cada sinal de relógio está calculada nas Eq. 34 e na Eq. 35.

$$\text{Margem 1(\%)} = 100 - \left(\frac{f_{relógio1}}{f_{máxima}} \right) 100 = 100 - \frac{33}{40,942} \times 100 = 19,40\% . \quad \text{Eq. 34}$$

$$\text{Margem 2(\%)} = 100 - \left(\frac{f_{relógio2}}{f_{máxima}} \right) 100 = 100 - \frac{27}{40,942} \times 100 = 34,05\% . \quad \text{Eq. 35}$$

Para o modelo com uma unidade optou-se por recorrer à utilização do sinal de relógio de 33 MHz que apresenta uma margem significativa. Por sua vez para o modelo com duas unidades utilizaram-se os dois sinais de relógio especificados.

Outro parâmetro a ser especificado consistiu no número de amostras a recolher pelo *Chipscope*. A Fig. 53 e a Fig. 54 ilustram a janela de configuração do compilador da *Xilinx* que permite definir o número de amostras a recolher. Optou-se pela recolha de 65536 amostras durante o flanco ascendente do sinal de relógio quando o sinal de *trigger* é activo. As amostras recolhidas foram salvaguardadas para um ficheiro texto. Uma análise deste ficheiro permite constatar que eram recolhidas 200 amostras por *trigger*. No entanto estas correspondem ao mesmo valor de pressão acústica calculada na etapa *SD12*. Como consequência apenas recolheram-se 327 amostras distintas (vide Eq. 36).

$$\text{Número efectivo de amostras} = \frac{N_{total}}{N_{D3}} = \frac{65536}{200} = 327 \text{ amostras} . \quad \text{Eq. 36}$$

Legenda:

N_{total} – Número total de amostras a recolher;

N_{D3} – Número de amostras recolhidas quando o *trigger* activa.

Valor esse perfeitamente aceitável tendo em conta as 400 de iterações que foram definidas na descrição em *VHDL* do Meshotron.

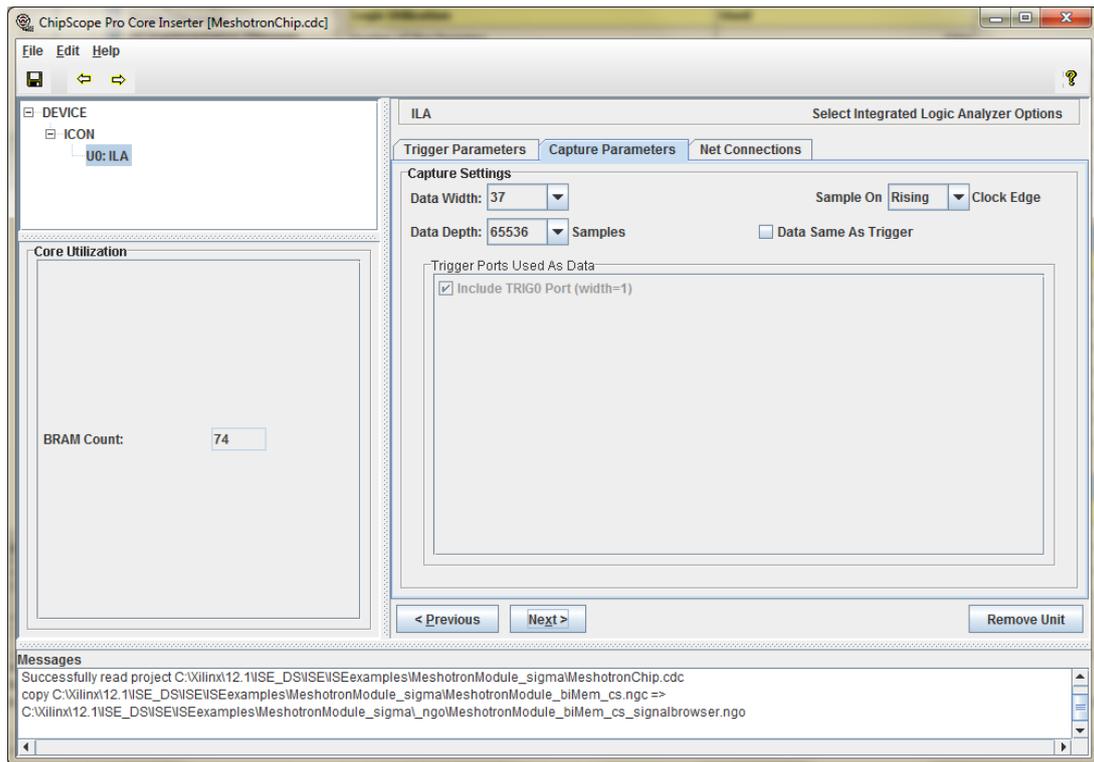


Fig. 53 – Interface gráfico de configuração do número de amostras a recolher pelo *Chipscope* no modelo com uma unidade do Meshotron.

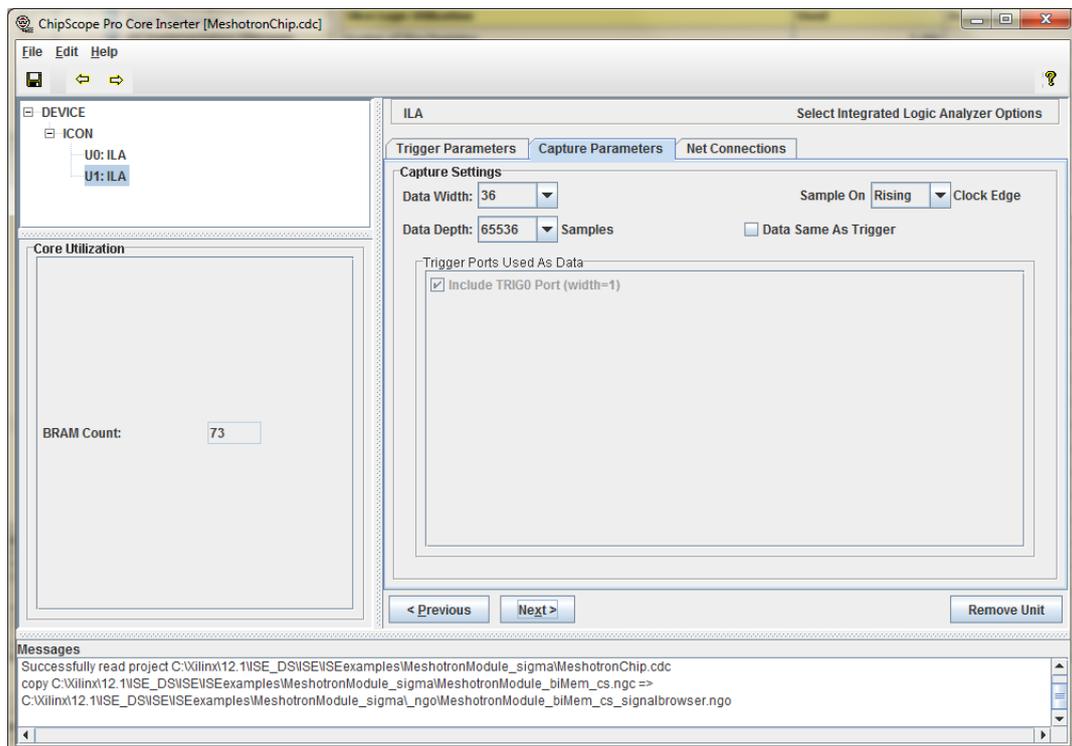


Fig. 54 – Interface gráfico de configuração do número de amostras a recolher pelo *Chipscope* no modelo com duas unidades do Meshotron.

4.4.2. Modelo com uma unidade do Meshotron

O modelo utilizado no primeiro teste apresenta uma unidade do Meshotron (vide Fig. 44) com $N=8$ (512 nós). Este foi testado com o nó emissor na coordenada (4,4,4), e o nó receptor nas coordenadas (0,0,0), (1,2,3) e (5,1,0). Em termos de precisão numérica numa primeira fase sujeitou-se a *RIR* obtida pelo *Chipscope* e pelo *Matlab* à mesma precisão numérica (inteiros de 32 *bits*) e, numa segunda fase, a precisões numéricas distintas (inteiros de 32 *bits* e *double*). Os sinais obtidos através do *Chipscope* foram guardados num ficheiro de texto.

O programa que permitiu validar o modelo construído está descrito no diagrama de blocos ilustrado na Fig. 55.

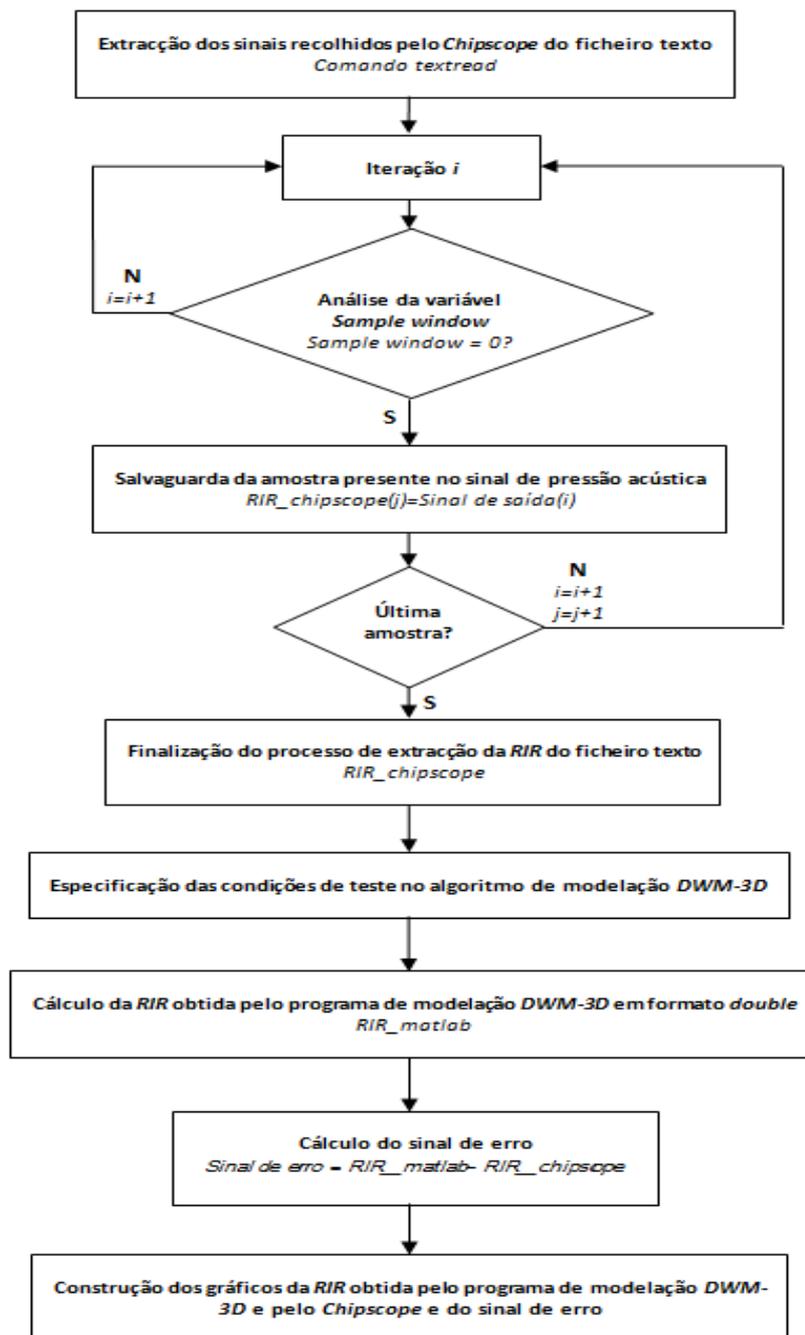


Fig. 55 – Diagrama de blocos do programa de validação do modelo com uma unidade do Meshotron.

O diagrama de blocos divide-se em quatro fases:

- Construção da *RIR* obtida pelo *Chipscope*;
- Cálculo da *RIR* obtida pelo programa de modelação *DWM-3D*;
- Cálculo do sinal de erro existente entre as *RIRs* sujeitas a precisões numéricas distintas;
- Comparação entre a *RIR* obtida pelo *Chipscope*, a *RIR* obtida pelo programa de modelação *DWM-3D* e o sinal de erro.

Note-se que durante a construção da *RIR* obtida pelo *Chipscope* é referenciada uma variável designada por *Sample window*. Por observação do ficheiro texto concluiu-se que esta variável ia a zero sempre que uma nova amostra era identificada. Como consequência recorreu-se a esta condição para extrair os valores da *RIR*. Outro aspecto a ser levado em consideração no diagrama de blocos é a utilização do comando *readtext*, que lê os valores inteiros presentes no ficheiro texto e armazena-os numa matriz.

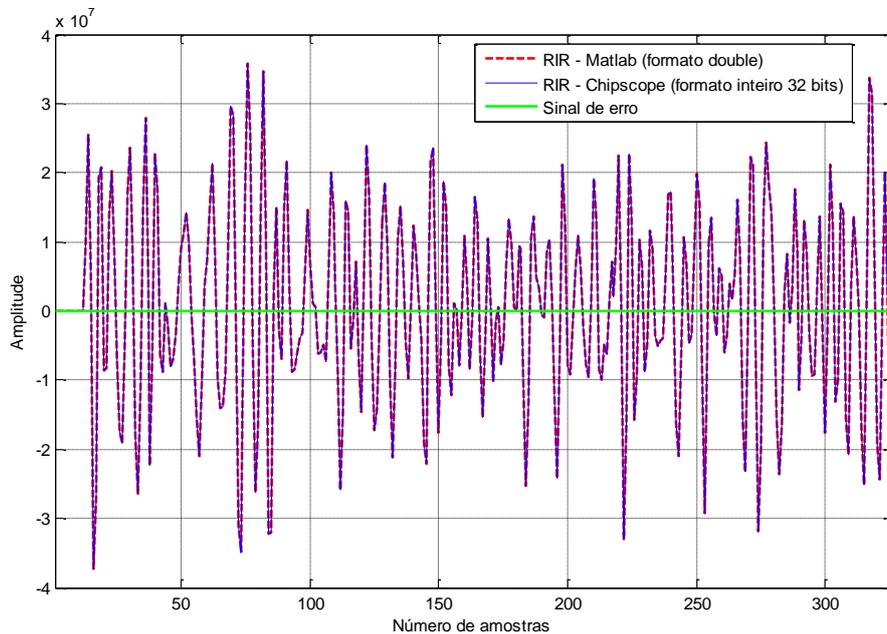


Fig. 56 – *RIR* obtida pelo *Chipscope*, pelo *Matlab* e o respectivo sinal de erro na coordenada (0,0,0).

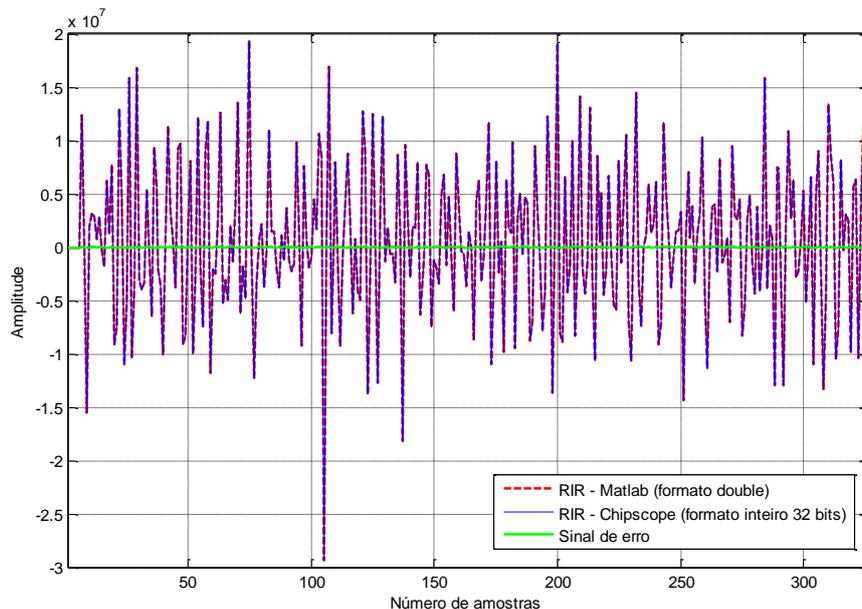


Fig. 57 – *RIR* obtida pelo *Chipscope*, pelo *Matlab* e respectivo sinal de erro para a coordenada (1,2,3).

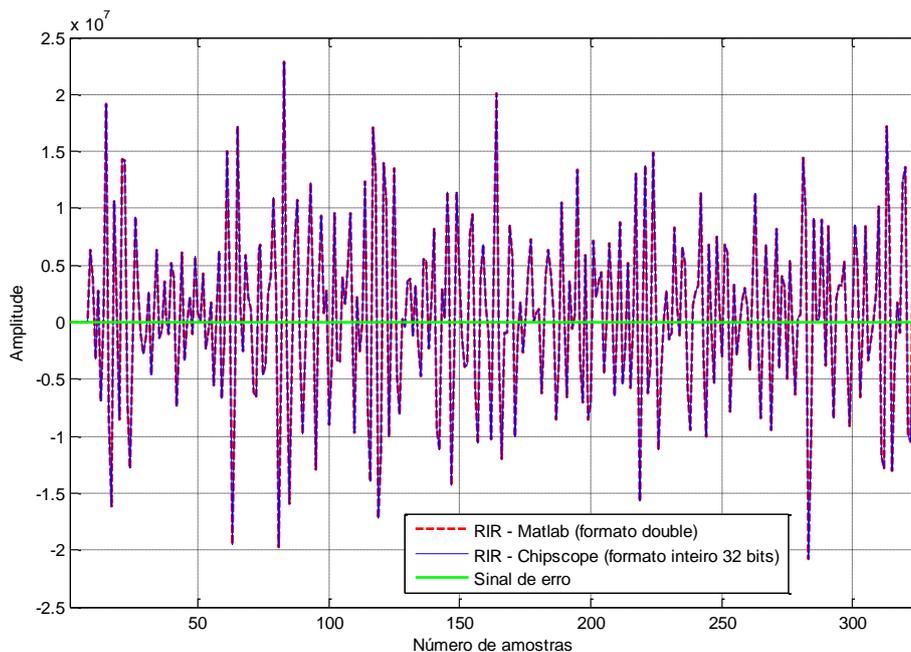


Fig. 58 – RIR obtida pelo *Chipscope*, pelo *Matlab* e respectivo sinal de erro para a coordenada (5,1,0).

A diferença entre a RIR obtida pelo *Matlab* e pelo *Chipscope* nas mesmas condições de precisão numérica foi nula. No entanto quando sujeitas a precisões numéricas distintas a diferença entre as duas RIRs obtidas é da ordem das dezenas, valores pouco significativos quando se compara com a ordem de grandeza dos valores da RIR (vide Fig. 56, Fig. 57 e Fig. 58).

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo em teste dever-se-á consultar o Apêndice C e o Apêndice D.

4.4.3. Modelo com duas unidades do Meshotron

Para este modelo (vide Fig. 45) efectuaram-se dois tipos de teste: variação do sinal de relógio e variação da coordenada do nó emissor.

4.4.3.1. Teste 1 - Variação do sinal de relógio

O primeiro teste baseou-se na verificação da comunicação entre as duas unidades do Meshotron por obtenção da RIR do modelo criado quando as unidades estão sujeitas a sinais de relógio distintos. Para tal colocou-se o nó emissor na coordenada (4,4,4) da unidade 0 e os nós receptores nas coordenadas (4,1,5) e (5,1,0) da unidade 1. Em termos de sinais de relógio adoptados numa primeira fase sujeitou-se a unidade 0 ao sinal de 27 MHz e a unidade 1 ao sinal de 33 MHz. Numa segunda fase a unidade 0 foi sujeita ao sinal de 33 MHz e a unidade 1 ao sinal de 27 MHz.

A verificação dos resultados passou pela execução do programa em *Matlab* cujos passos são descritos pelo diagrama de blocos ilustrado na Fig. 59.

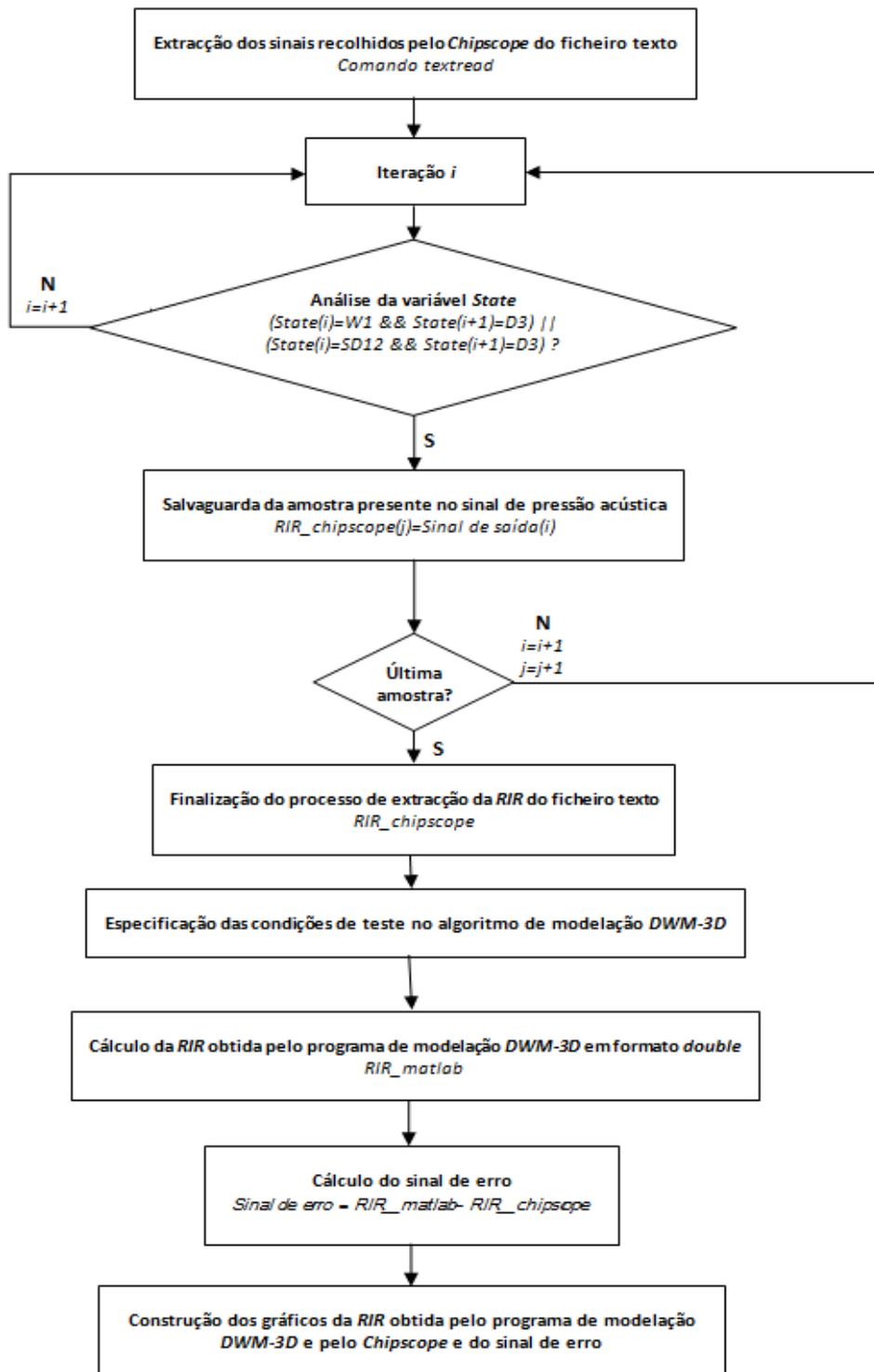


Fig. 59 – Diagrama de blocos do programa de validação do modelo com duas unidades do Meshotron.

O diagrama de blocos divide-se em quatro fases:

- Construção da *RIR* obtida pelo *Chipscope*;
- Construção da *RIR* obtida pelo programa de modelação *DWM-3D*;
- Cálculo do sinal de erro entre a *RIR* obtida pelo *Chipscope* e a *RIR* obtida pelo programa de modelação *DWM-3D* sujeitos a precisões numéricas distintas;

- Comparação entre as RIR_s obtidas no *Chipscope* e no programa de modelação *DWM-3D* e o sinal de erro;

O aumento da complexidade do modelo impediu a utilização da condição da variável *Sample window* utilizada na secção 4.4.2, uma vez que esta é continuamente incrementada como se verifica pela análise do ficheiro texto. Como consequência analisou-se o estado em execução na unidade receptora, uma vez que a condição de *trigger* depende do estado. A nova condição de identificação de uma nova amostra passa pela verificação de quando é efectuada a transição entre o estado $W1$ para o estado $D3$, se a unidade 1 for definida como unidade mais rápida; ou de quando é efectuada a transição do estado $SD12$ para o estado $D3$, se a unidade 1 for definida como unidade mais lenta. Isto é, a condição de identificação baseia-se na análise da passagem para o estado $D3$.

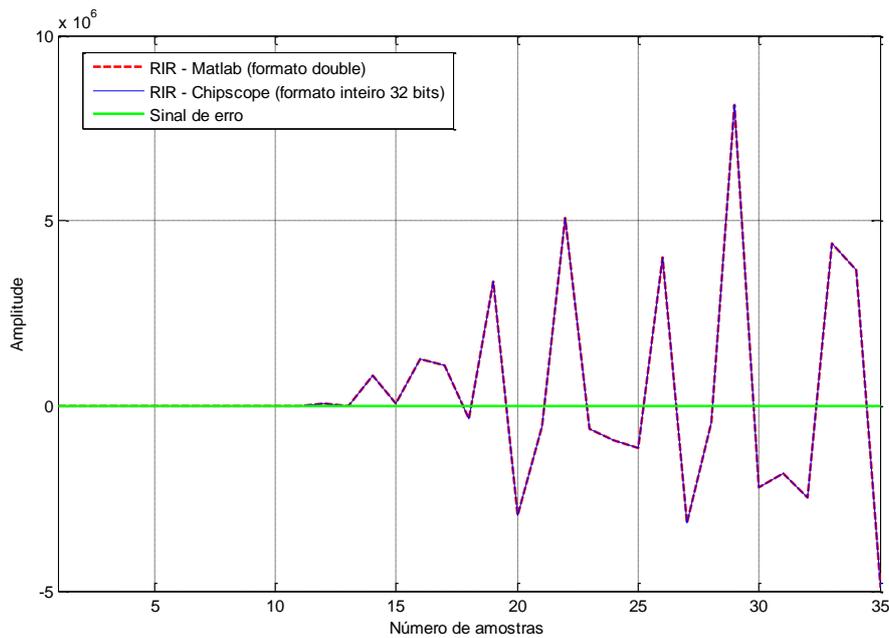


Fig. 60 – RIR obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 33 MHz.

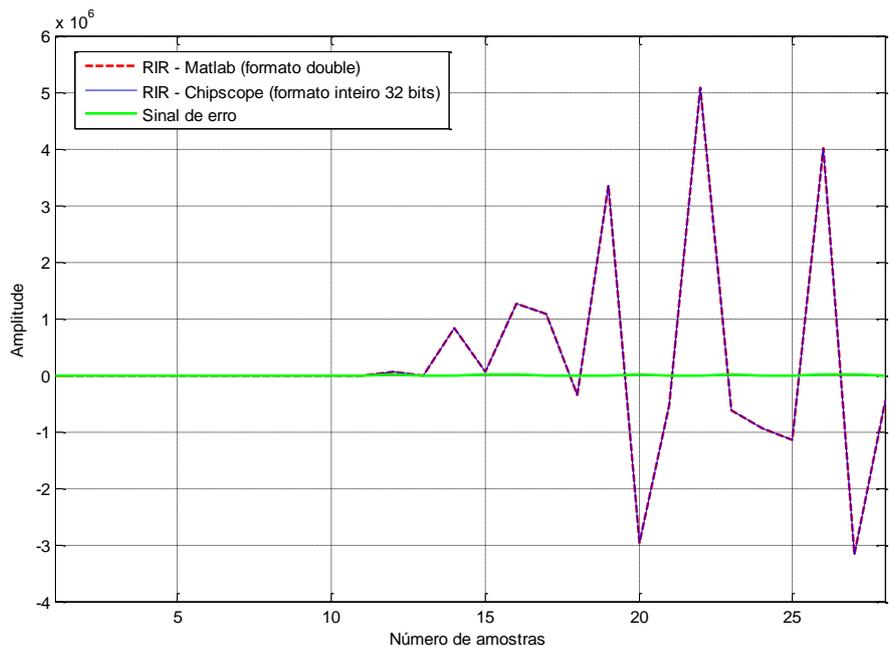


Fig. 61 - RIR obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 27 MHz.

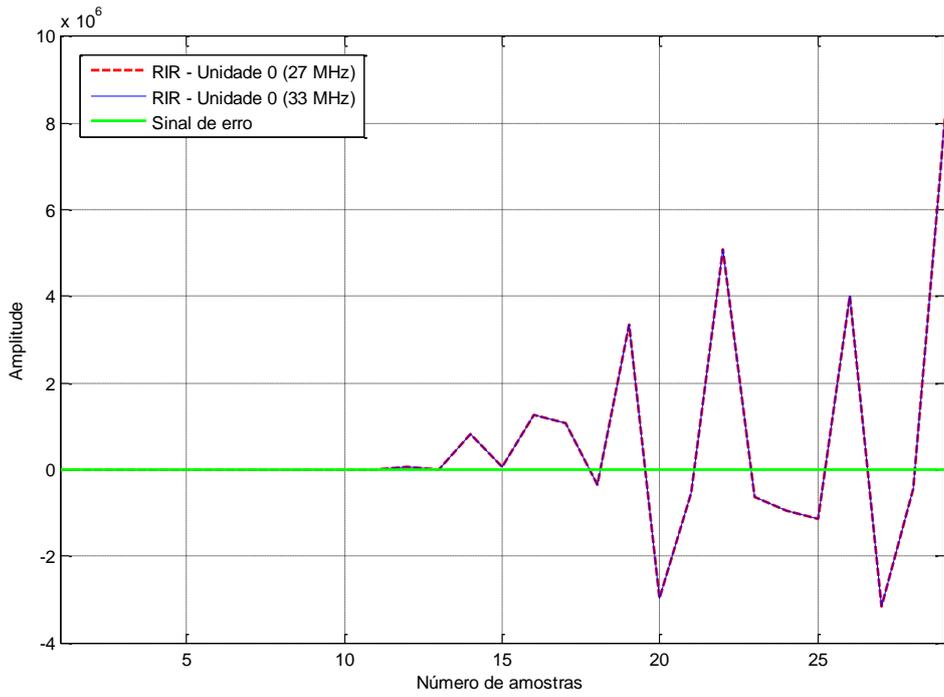


Fig. 62 - RIR obtida no *Chipscope* para a coordenada (4,1,5) com a unidade 0 com um sinal de relógio de 27 MHz e de 33 MHz.

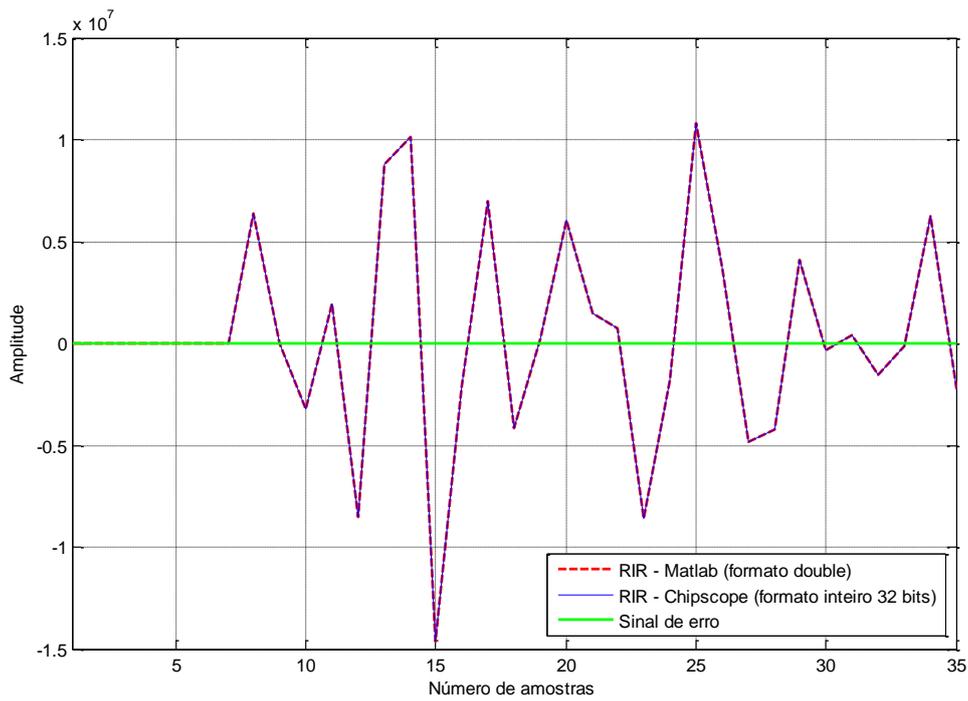


Fig. 63 - RIR obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 33 MHz.

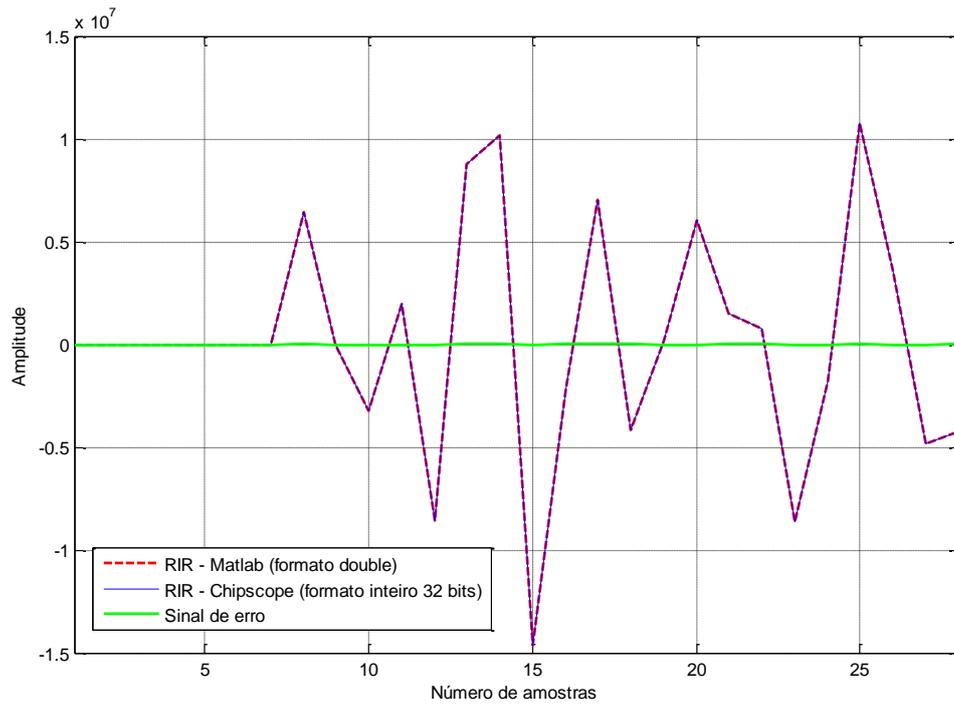


Fig. 64 - RIR obtida no Matlab e no Chipscope em diferentes precisões numéricas para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 27 MHz.

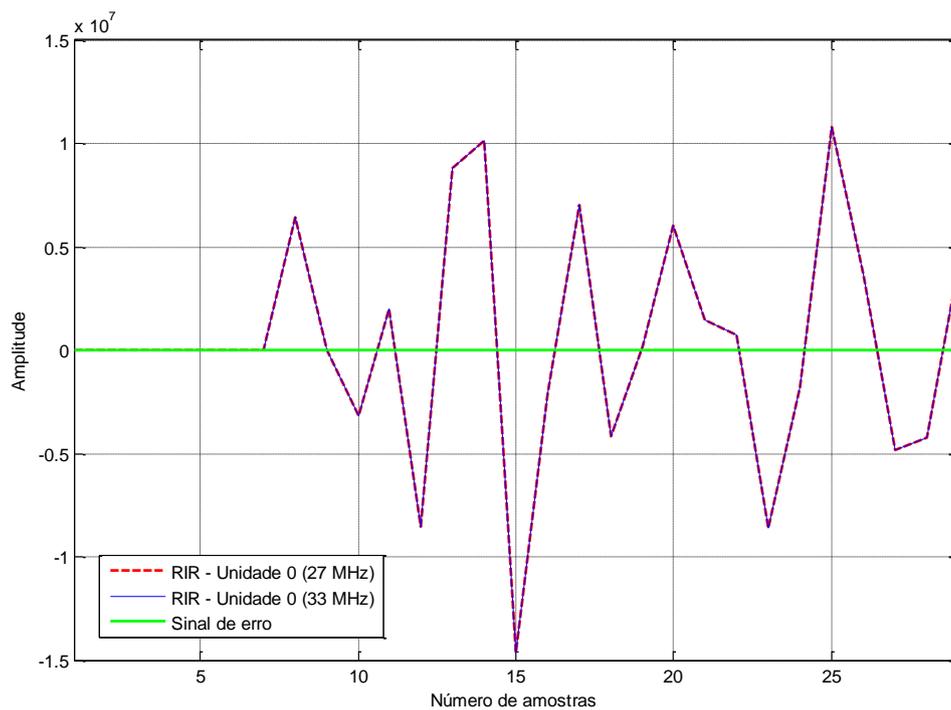


Fig. 65 - RIR obtida no Chipscope para a coordenada (5,1,0) com a unidade 0 com um sinal de relógio de 27 MHz e de 33 MHz.

A diferença entre as RIRs nas mesmas condições de precisão numérica foi nula. Por sua vez quando sujeitas a precisões numéricas distintas a diferença é da ordem das dezenas, valores pouco significativos quando se compara com a ordem de grandeza dos valores da RIR (vide Fig. 60, Fig. 61, Fig. 63 e Fig. 64). Outro aspecto a referir é o facto de a RIR ser completamente independente do sinal de relógio a que cada unidade do Meshotron está sujeita, uma vez que os sinais de relógio associados a cada unidade apenas vão influenciar a

sua entrada em *wait state* (vide Fig. 62 e Fig. 65). Logo concluir-se-á que o mecanismo de sincronização entre unidades do Meshotron adjacentes está a operar correctamente (vide secção 3.2.2.1).

Ainda relativamente aos gráficos anteriores observa-se que o número de amostras para o modelo em análise (vide Fig. 45) é menor que para o modelo anterior (vide Fig. 44). Isto deve-se à diferença entre os sinais de relógio das duas unidades do Meshotron que provocam a entrada da unidade mais rápida em *wait state* condicionando deste modo o número de amostras distintas a serem recolhidas pelo *Chipscope*.

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo em teste dever-se-á consultar o Apêndice C e o Apêndice D.

4.4.3.2. Teste 2 - Variação da coordenada do nó emissor

O segundo teste baseia-se na variação da coordenada do nó emissor. Isto é, recolheu-se a *RIR* obtida pelo *Chipscope* quando o nó receptor é fixado na coordenada (0,0,0) da unidade 1 e o nó emissor é colocado nas coordenadas (1,2,3), (3,1,0) e (7,2,1) da unidade 0. Por último colocou-se o nó receptor na coordenada (1,4,5) e o nó emissor na coordenada (1,6,7).

Para este teste definiu-se para a unidade 0 um sinal de relógio de 33 MHz e para a unidade 1 um sinal de relógio de 27 MHz. Em termos de condições de precisão numérica a *RIR* encontra-se sujeita, para este teste adoptaram-se as condições impostas na secção 4.4.2.

O procedimento de verificação dos resultados é igual ao descrito no diagrama de blocos na secção 4.4.2.

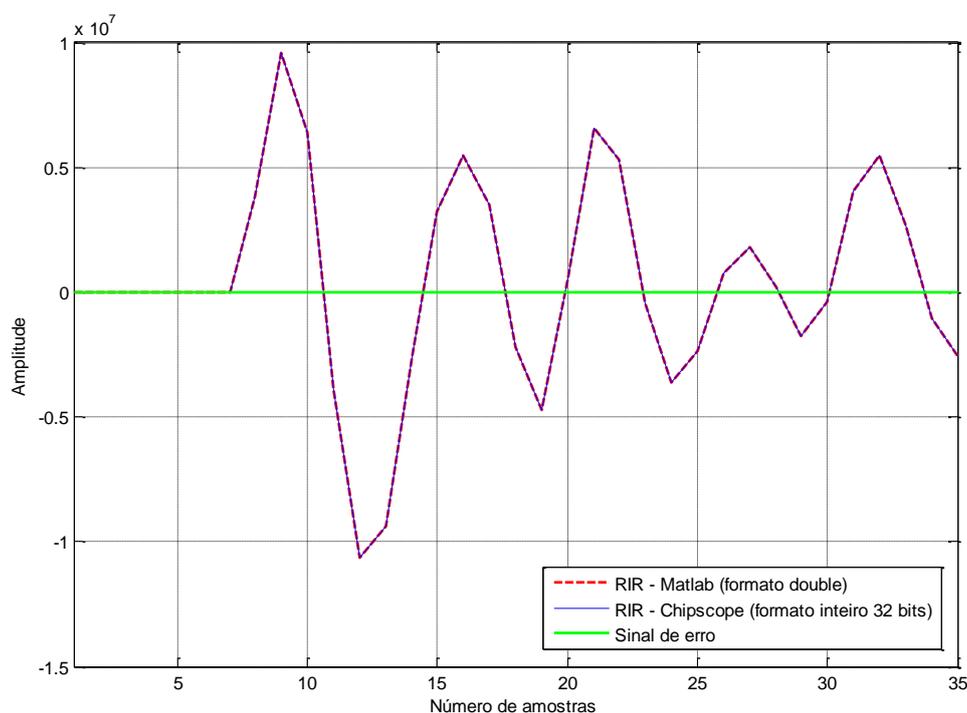


Fig. 66 - *RIR* obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas com o nó emissor na coordenada (1,2,3) e o nó receptor na coordenada (0,0,0).

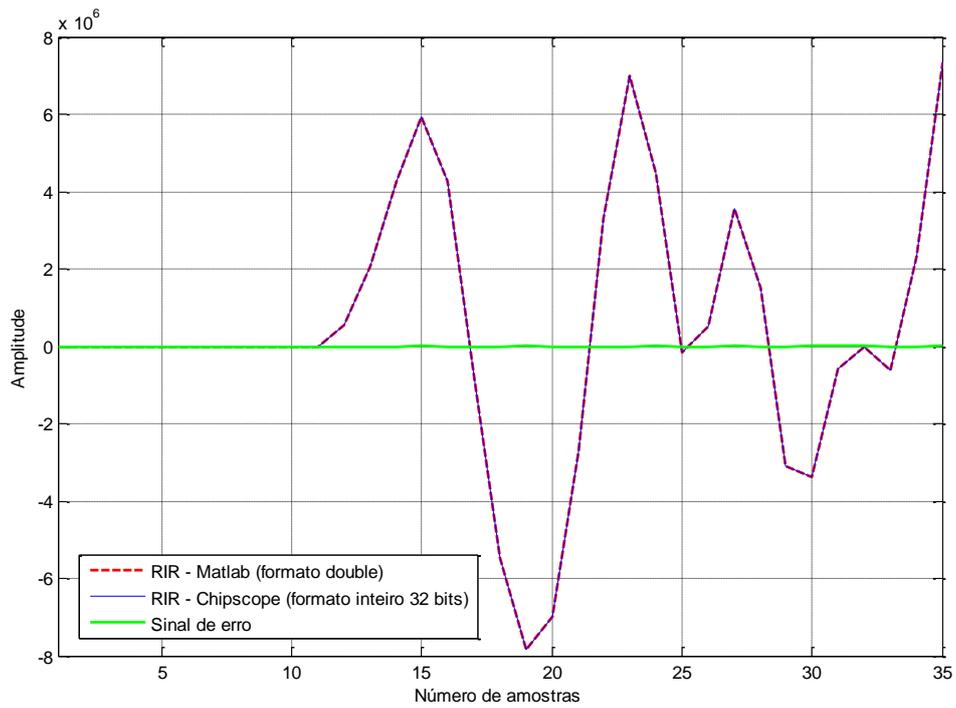


Fig. 67 – RIR obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas com o nó emissor na coordenada (3,1,0) e o nó receptor na coordenada (0,0,0).

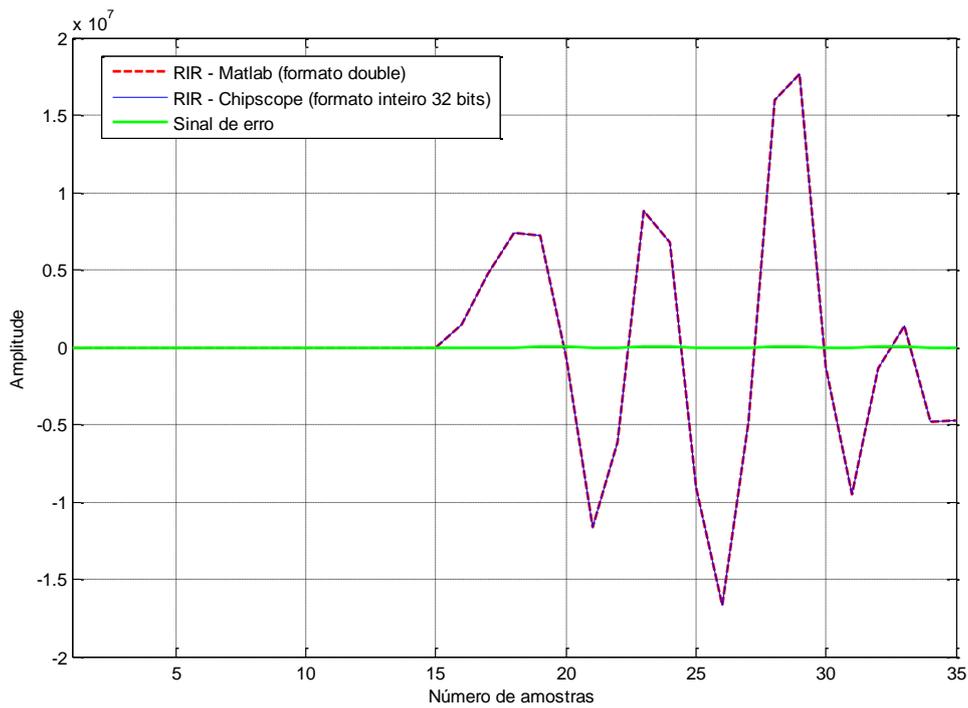


Fig. 68 - RIR obtida no *Matlab* e no *Chipscope* em diferentes precisões numéricas com o nó emissor na coordenada (7,2,1) e o nó receptor na coordenada (0,0,0).

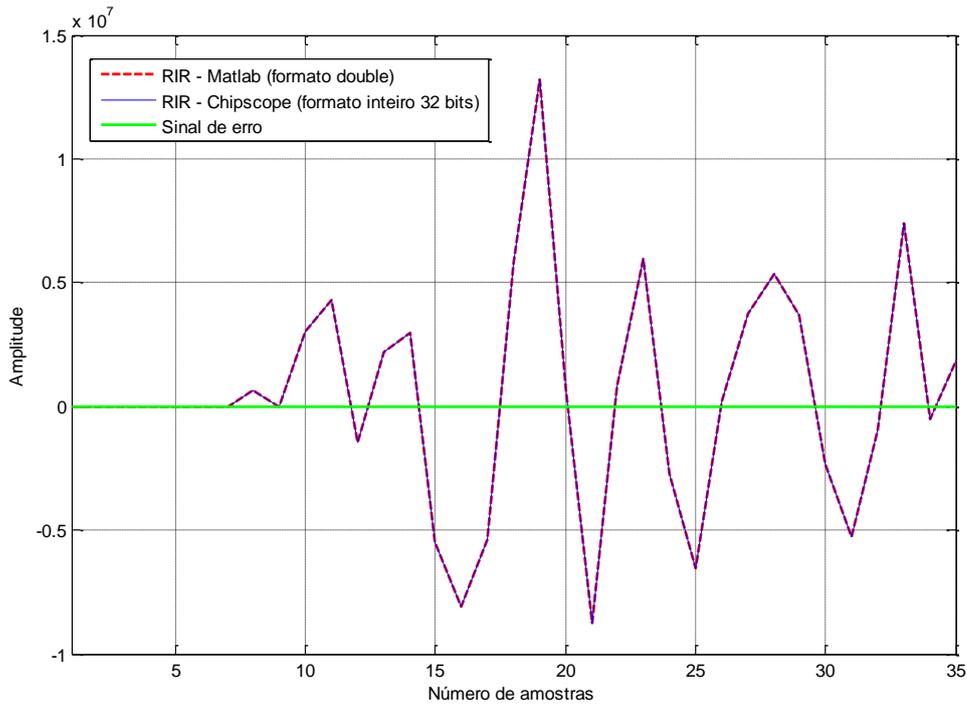


Fig. 69 - RIR obtida no Matlab e no Chipscope em diferentes precisões numéricas com o nó emissor na coordenada (1,6,7) e o nó receptor na coordenada (1,4,5).

A diferença entre as *RIRs* nas mesmas condições de precisão numérica foi nula. Por sua vez quando sujeitas a precisões numéricas distintas a diferença é da ordem das dezenas, valores pouco significativos quando se compara com a ordem de grandeza dos valores da *RIR* na coordenada do nó receptor em análise (vide Fig. 66, Fig. 67, Fig. 68 e Fig. 69).

Para mais detalhes sobre o código criado em *VHDL* e em *Matlab* que permitiram validar o modelo em teste dever-se-á consultar o Apêndice C e o Apêndice D.

4.4.4. Análise dos resultados obtidos

Os resultados associados ao modelo de 1 unidade permitem concluir que a unidade do Meshotron está a operar correctamente, uma vez que a *RIR* obtida pela ferramenta de depuração *Chipscope* para as diferentes coordenadas do nó receptor é similar à *RIR* obtida pelo programa de modelação *DWM-3D*. Note-se que mesmo quando as *RIRs* obtidas por *software* e pela ferramenta de depuração *Chipscope* são sujeitas a precisões numéricas distintas, o sinal de erro é praticamente desprezável quando se compara com a ordem de grandeza da *RIR*. Logo a escolha de parâmetros de dados de 32 *bits* efectuada na secção 4.2.1 foi bem efectuada.

A análise dos resultados do modelo de 2 unidades permitiu depreender que a *RIR* obtida é independente do sinal de relógio aplicado a cada uma das unidades e que o mecanismo de sincronização entre unidades do Meshotron adjacentes está a operar correctamente, uma vez que as *RIRs* obtidas pela ferramenta de depuração *Chipscope* são iguais às obtidas pelo programa de modelação *DWM-3D*. Mesmo quando sujeitas a precisões numéricas distintas a ordem de grandeza do sinal de erro torna-o desprezável quando comparado com a ordem de grandeza da *RIR*.

Capítulo 5 Conclusão e trabalho futuro

5.1. Análise final dos resultados e conclusões

Nesta dissertação foi apresentada e discutida a arquitectura e o desenvolvimento da unidade-base do Meshotron. A validação do seu funcionamento passou pela simulação, implementação e depuração de um conjunto de modelos acústicos. Estas fases tinham como objectivo extrair a *RIR* de um conjunto de modelos descritos em *VHDL*, e seguidamente comparar os resultados obtidos com os obtidos pelo programa de modelação *DWM-3D* criado em *Matlab*. Por análise destes resultados conclui-se que o Meshotron efectua a descrição acústica de um espaço fechado com base num impulso de *Dirac*, com resultados iguais aos obtidos nas mesmas condições funcionamento de um programa de modelação *DWM-3D* desenvolvido em *Matlab*.

As simulações para o conjunto de modelos criados permitiram concluir que os resultados obtidos estão de acordo com os obtidos através programa de modelação *DWM-3D* nas mesmas condições de funcionamento. Em termos de testes de implementação e depuração, os resultados da *RIR* para os modelos testados estão de acordo com os resultados obtidos pelo programa de modelação *DWM-3D*, quando sujeitos a condições similares de precisão numérica (inteiros de 32 *bits*). No entanto, quando sujeitos a condições de precisão numérica distintas (*double* e inteiros de 32 *bits*), existe uma diferença entre os resultados obtidos através da ferramenta *Chipscope* e através do programa de modelação *DWM-3D* desenvolvido em *Matlab*, diferença essa perfeitamente desprezável tendo em conta a ordem de grandeza dos valores envolvidos nas operações. Note-se que os resultados obtidos para o modelo com duas unidades permitiu concluir que o mecanismo de sincronização entre unidades adjacentes está a operar correctamente e que as unidades do Meshotron podem operar a sinais de relógio distintos. Desde que a sua frequência máxima de funcionamento seja contemplada na selecção dos sinais de relógio.

Para além disso os elementos de armazenamento e os respectivos barramentos de dados, de controlo e de endereçamento foram devidamente estruturados de modo a possibilitar o seu desempenho actual. Em suma poder-se-á afirmar que o conceito de paralelização do Meshotron abre um caminho para a implementação de modelos *Digital Waveguide Mesh-3D* com um desempenho muito aceitável.

5.2. Trabalho futuro

Em termos de trabalho futuro as tarefas a realizar são:

- Refinar a unidade de *scattering*, de modo a que esta seja capaz de lidar com os *nós fronteira* (terminação *Mesh 1D*). Isto pode ser feito pela utilização de um *multiplexer* que selecione, de acordo com o registo *cfg* de cada nó, o mecanismo de cálculo associado aos nós fronteira ou nós de ar (vide secção 3.2.1.6);
- Efectuar um estudo de implementações de condições fronteiras mais sofisticadas;
- Avaliar em detalhe a duração temporal das várias operações do Meshotron, como por exemplo as operações de endereçamento, leitura e escrita nos bancos de memória, de forma a otimizar/balancear a duração das várias fases de processamento;

- Aumentar a versatilidade do elemento que efectua a excitação do modelo *DWM-3D*, por introdução de outro tipo de sinais diferentes do impulso de *Dirac*;
- Efectuar a ligação do Meshotron a um computador por ligação de cabo de rede para efectuar-se a preparação do modelo acústico. Como consequência ferramentas de *software* têm de ser desenvolvidas para tornar automático o processo de:
 - Configuração do modelo acústico – consiste na transferência da informação de configuração para o banco de memória *cfg* durante o estado de inicialização;
 - Extração da *RIR* do sistema – consiste em efectuar a transferência da *RIR* para o computador do endereço especificado dos bancos de memória *p*;
- Adaptar/Construir o código de *VHDL* para outras topologias, em particular a topologia tetraédrica, uma vez que nesta topologia a operação de divisão na unidade de *scattering* resume-se a um deslocamento de *bits* para a direita.

Bibliografia

- [1] J. O. Smith III, "Physical Modeling Using Waveguides," *Computer Music Journal*, vol. 16(4), Winter, pp. 74-91, 1992.
- [2] J. O. Smith III, "Principles of Digital Waveguide Models of Musical Instruments," em M Kahrs e K Brandenburg (edição), *Applications of Digital Signal Processing to Audio and Acoustics*, pp. 417-466. Editora Acadêmica Kluwer, 1998.
- [3] S. Van Duyne e J. O. Smith III, "Physical Modeling with the 2-D Digital Waveguide Mesh," na *Proc. Int. Computer Music Conf. (ICMC'93)*, pp. 40-47, Tóquio, Setembro, 1993.
- [4] F. Fontana e D. Rocchesso, "Physical Modelling of Membranes for Percussion Instruments," *Acustica – Acta Acustica*, vol. 84, Maio/Junho, pp. 529-542, 1998.
- [5] L. Savioja, M. Karjalainen e T. Takala, "DSP Formulation of a Finite Difference Method for Room Acoustics Simulation," no *Proc. IEEE Nordic Signal Processing Symp. (NORSIG'96)*, pp. 455-458, Espoo, Finlândia, Setembro 24-27, 1996.
- [6] G. Campos, "Three-dimensional Digital Waveguide Mesh Modelling for Room Acoustic Simulation," Tese de Doutorado, Universidade de York, 2003.
- [7] L. Beranek, *Concert and Opera Halls: How They Sound*. Woodbury, New York: Acoustical Society of America, 1996.
- [8] G. Campos e D. M. Howard, "On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation," *IEEE Trans. Speech Audio Process.*, vol. 13(5), pp. 1063-1072, Setembro 2005.
- [9] G. Campos e D. M. Howard, "On the Computation Time of Three-Dimensional Digital Waveguide Acoustic Models," na *Proc. 26th Euromicro Conf.*, vol. II, pp. 332-339, Maastricht, Holanda, Setembro. 5-7, 2000.
- [10] S. Van Duyne e J. O. Smith III "The 3-D Tetrahedral Digital Waveguide Mesh with Musical Applications," na *Proc. Int. Computer Music Conf. (ICMC'96)*, pp. 9-16, Hong-Kong, Agosto, 1996.
- [11] G. Campos e D. M. Howard, "A Parallel 3D Digital Waveguide Mesh Model with Tetrahedral Topology for Room Acoustic Simulation," na *Proc. COST G-6 Conf. on Digital Audio Effects (DAFx-00)*, pp. 73-78, Verona, Itália, 7-9 Dezembro, 2000.
- [12] D. T. Murphy e D. M. Howard, "2-D Digital Waveguide Mesh Topologies in room acoustic modelling", *COST G-6 Conference on Digital Audio Effects(DAFx-00)*, Verona, Itália, pp. 1-3, 7-9 Dezembro 2000.
- [13] S. Bilbao, "Wave and Scattering Methods for the Numerical Integration of Partial Differential Equations," Tese de Doutorado, Universidade de Stanford, 2001.
- [14] L. Savioja, *et al.*, "The interpolated 3-D Digital Waveguide Mesh method for room acoustic simulation and auralization," *Joint Baltic-Nordic Acoustical Meeting*, Lyngby, Dinamarca, pp. 1-2, Agosto 2002.

- [15] G. Campos e S. Barros, “The *Mesbotron*: a network of specialised hardware units for 3D Digital Waveguide Mesh acoustic model parallelisation”, na *128^a Conferência AES*, paper 296, Londres, Maio 22-25, 2010.
- [16] G. Campos e S. Barros, “The *Mesbotron*: a network of specialised hardware units for 3D Digital Waveguide Mesh acoustic model parallelisation”, na *128^a Conferência AES*, poster, Londres, Maio 22-25, 2010.
- [17] J. Wakerly, *Digital Design – Principles and Practices*. 4^a Edição, Prentice-Hall, 2005.
- [18] Smith, Julius O, “Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects”, Edição de Dezembro de 2005, livro *online* no site: <http://ccrma.stanford.edu/~jos/pasp05/>, Centro de Investigação Computacional em Música e Acústica (CCRMA), Universidade de Stanford, Dezembro de 2005.
- [19] Xilinx, *Tutorial ISE*, URL: <http://www.xilinx.com> – Abril 2010. – 20 de Março 2011.
- [20] Xilinx, *ML505/ML506/ML507 Evaluation Platform User Guide*, URL: http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf – 20 de Março 2011.
- [21] Digital Waveguide Mesh, URL: <http://www.mattmontag.com/auralization/waveguide.php> – Fevereiro 2011.
- [22] Ray-tracing, URL: [http://en.wikipedia.org/wiki/Ray_tracing_\(physics\)](http://en.wikipedia.org/wiki/Ray_tracing_(physics)) – 10 de Abril 2011.
- [23] Resposta impulsional (*RIR*), URL: http://en.wikipedia.org/wiki/Impulse_response - 25 de Março 2011.
- [24] Método dos Elementos Finitos (*FEM*), URL: http://en.wikipedia.org/wiki/Finite_element_method - 25 de Março 2011.
- [25] Método dos elementos fronteira (*BEM*), URL: http://en.wikipedia.org/wiki/Boundary_Element_Method - 19 de Março 2011.
- [26] *Ray-tracing* e *Image source*, URL: http://www.akustikon.se/eng/software_e.html - 20 de Maio 2011.
- [27] Método da diferença finita no domínio do tempo (FDTD) <http://en.wikipedia.org/wiki/FDTD> - 28 de Abril 2011.
- [28] Modelação por Linha de Transmissão (*Transmission Line Modelling*), URL: http://en.wikipedia.org/wiki/Transmission_line_modelling - 20 de Maio 2011.

Apêndice A *Fifo dual-clock*

A Fig. 70, a Fig. 71 e a Fig. 72 ilustram o interface gráfico de configuração da *FIFO dual-clock* fornecida pelo compilador da *Xilinx* e a configuração do respectivo elemento.

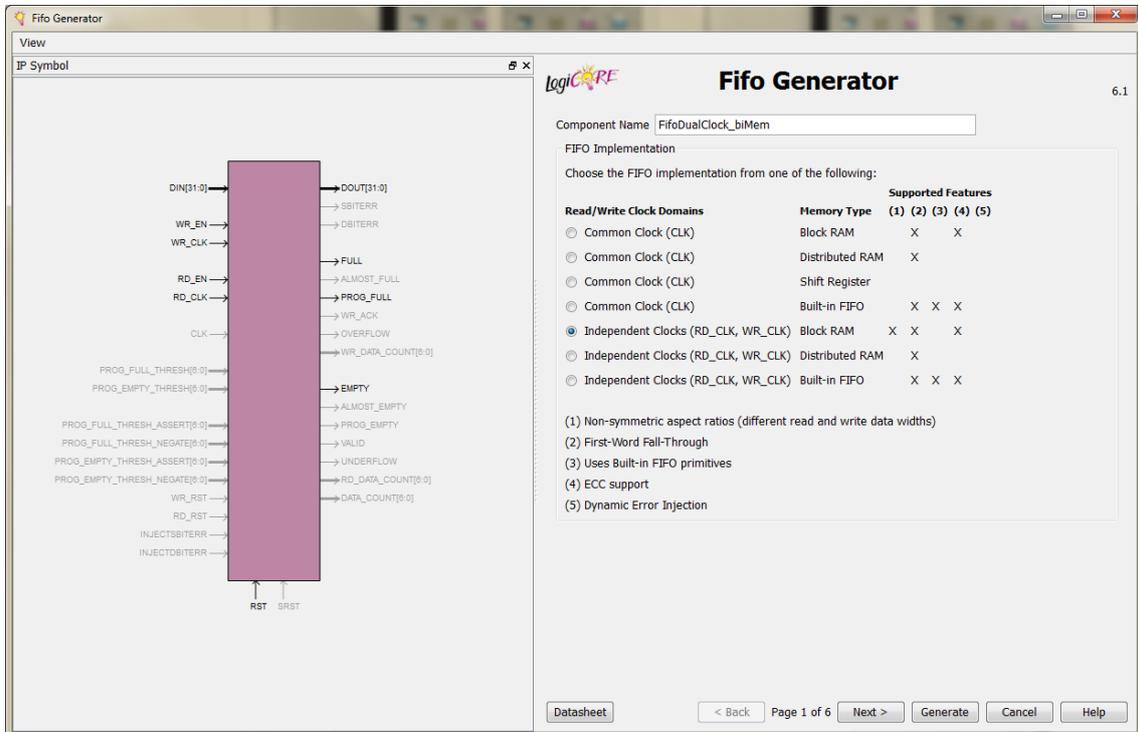


Fig. 70 – Janela de configuração do número de sinais de relógio da *FIFO*.

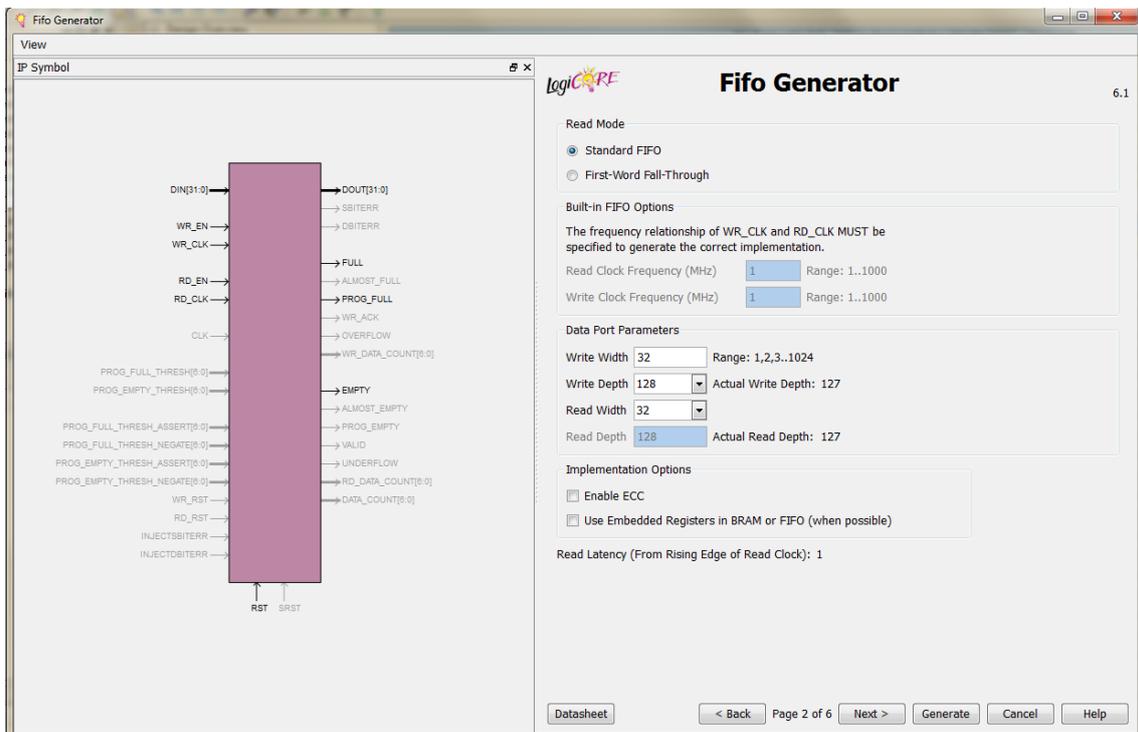


Fig. 71 – Janela de configuração do número de bits dos parâmetros de dados da *FIFO*.

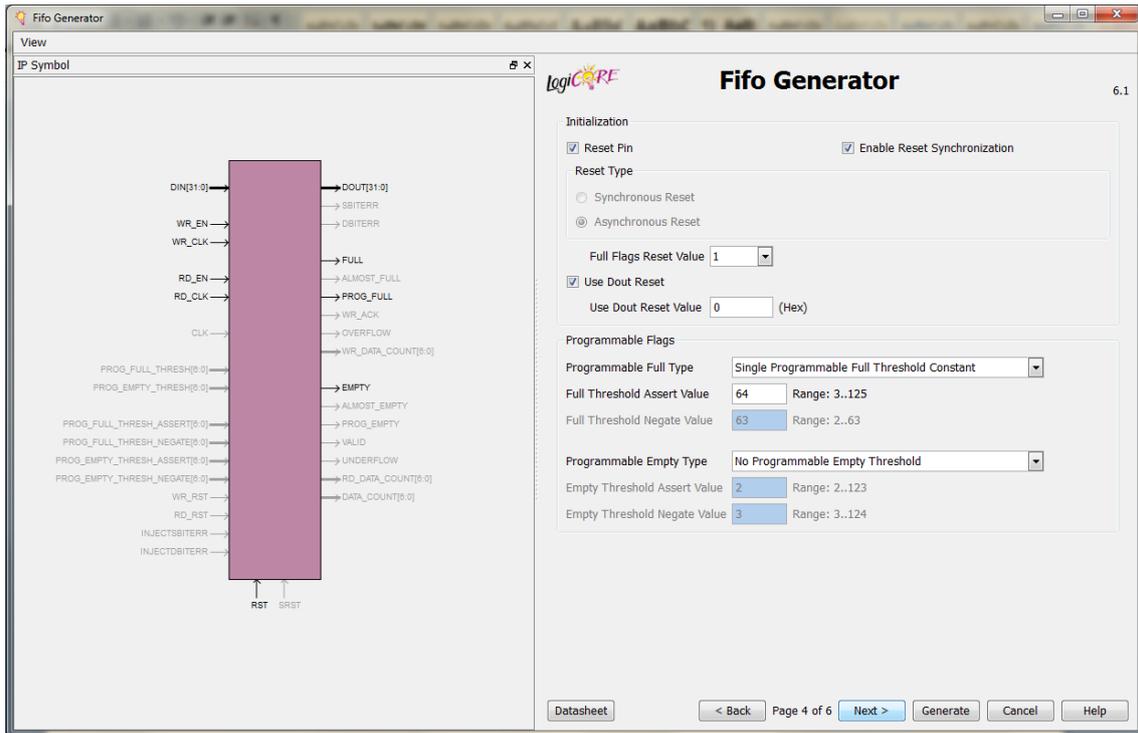


Fig. 72 – Janela de configuração dos sinais de controlo da FIFO.

Apêndice B Divisor por 3

B.1. Soluções possíveis

Este problema tem várias soluções distintas, uma vez que esta temática tem sido bastante estudada, de modo a possibilitar a implementação de algoritmos de divisão cada vez mais eficientes nos processadores e respectivas unidades aritméticas lógicas. De entre as soluções encontradas, optou-se pelas soluções mais simples, uma vez que no caso da aplicação em causa a precisão não era um dos factores preponderantes. Logo, o nosso leque de soluções cingiu-se a três métodos distintos, que são os seguintes: multiplicação do valor pretendido por 1/3 em binário em vírgula fixa, utilização de um divisor fornecido pelo compilador da *Xilinx* e utilização de um divisor baseado em lógica combinatória.

B.1.1. 1º Método – “Multiplicação por 1/3”

O primeiro método baseia-se na multiplicação do valor pretendido por 1/3. Todavia, como a representação em binário deste número racional é impossível, tem de se recorrer à utilização de um valor aproximado $0.328125 = 0.01010101010101010101$, o que vai introduzir um erro ao nível do resultado final desta operação. Neste método, depois de se efectuar o produtório do operando pela respectiva aproximação em binário de 1/3, extrai-se do resultado os *m bits* menos significativos, que correspondem à parte decimal do resultado. Os *bits* que sobram, correspondem ao resultado inteiro da multiplicação. Recorreu-se a este processo, uma vez que geralmente a operação de multiplicação é mais simples e utiliza menos recursos que a operação de divisão. Neste caso, recorre-se à utilização de alguns dos multiplicadores presentes na *FPGA*. O método de cálculo encontra-se exemplificado na Fig. 73.

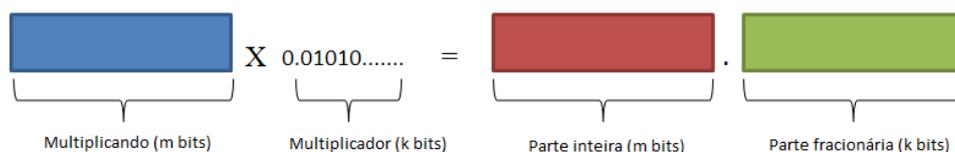


Fig. 73 – Demonstração da operação de multiplicação.

B.1.2. 2º Método – “Bloco de divisão da *Xilinx*”

O segundo método baseia-se na utilização de uma das ferramentas do programa de compilação da *Xilinx*, *IP – Core Generator & Architecture Wizard*, que permite criar um bloco de divisão de números de *n-bits* (vide Fig. 74), que pode ser interpretado pelo utilizador como uma caixa preta (a sua estrutura interna é desconhecida para o utilizador). Isto é, são lhe fornecidos os respectivos parâmetros de entrada e este disponibiliza o respectivo resultado final em valor inteiro. Neste caso, definiram-se os parâmetros de entrada como números *signed* (tanto o divisor como o dividendo) e em termos temporais um ciclo de

relógio por divisão. Para manter condições similares entre os diferentes blocos de divisão, o divisor neste método é fornecido como uma constante.

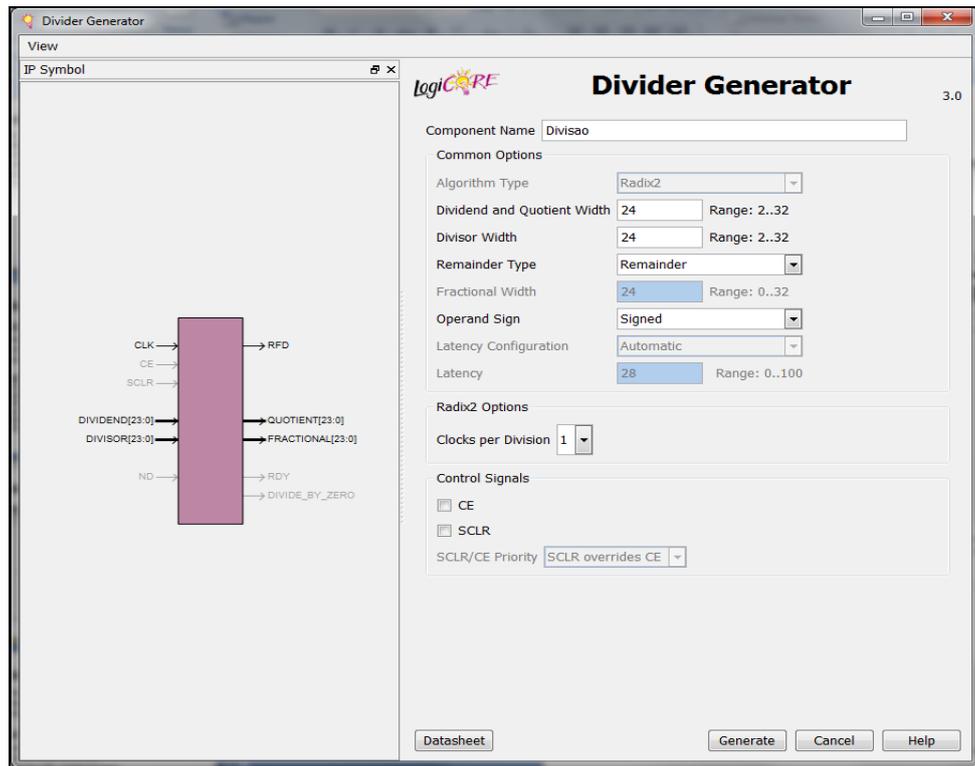


Fig. 74 – Interface gráfico de configuração do bloco de divisão da Xilinx.

B.1.3. 3º Método – “Bloco de divisão com lógica combinatória”

O terceiro método baseia-se na utilização de um algoritmo de divisão, que permite obter o valor inteiro do quociente e o respectivo resto da operação através de lógica combinatória. Este algoritmo tem como princípio base considerar a estrutura modular de um circuito combinatório, que permite a divisão semelhante a um somador completo ou um incrementador, em que o respectivo *carry* se propaga da esquerda para a direita e se encontra limitado pelos valores adquiridos pelo divisor. Este método estabelece a seguinte relação entre os respectivos parâmetros de entrada e saída:

$$2^b \times c_b + a = \beta \times s + c_0, \text{ com } 2^k \geq \beta. \quad \text{Eq. 37}$$

Pode ser exemplificada pela Fig. 75:

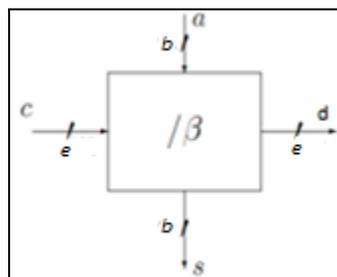


Fig. 75 – Bloco elementar de um divisor.

Dado que o objectivo seria a construção de um módulo elementar de um divisor por 3, consideraram-se os seguintes valores: $b=1$, $k=2$ e $\beta=3$, e procedeu-se à construção das respectivas tabela de verdade (vide Fig. 76) e funções booleanas (Eq. 38, Eq. 39 e Eq. 40), tendo sempre em conta a relação existente entre as respectivas variáveis.

$$s = c_1 + a \times c_0 \quad \text{Eq. 38}$$

$$d_1 = \bar{a} \times c_0 + a \times c_1 \quad \text{Eq. 39}$$

$$d_0 = \bar{a} \times c_1 + a \times \bar{c}_1 \times \bar{c}_0 \quad \text{Eq. 40}$$

Entradas			Saídas		
c_1	c_0	a	s	d_1	d_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0			
1	1	1			

Fig. 76 – Tabela de verdade.

Após a construção do bloco elementar, colocaram-se vários blocos em cascata para possibilitar a construção do módulo de divisão por 3 de números de m bits (vide Fig. 77).

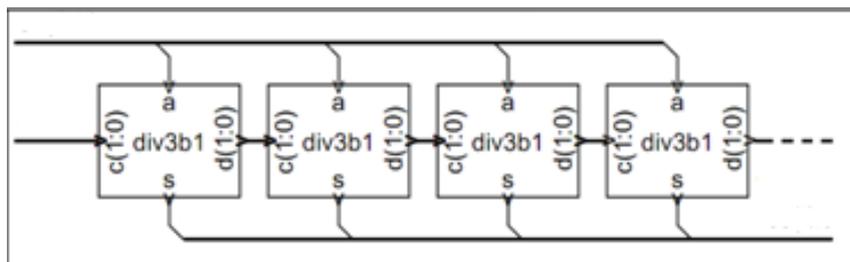


Fig. 77 – Blocos elementares em cascata.

B.2. Métricas a serem avaliadas

De entre os métodos propostos tem de se escolher o melhor para a aplicação em causa. Essa escolha é efectuada com base nos recursos utilizados (número de *LUTs*, *shift registers*, *flip-flops* e multiplicadores), erro relativo introduzido no resultado da divisão inteira e frequência máxima de funcionamento de cada um dos diferentes métodos testados, quando sujeitos a condições de funcionamento idênticas: registos à entrada e saída de cada um destes módulos, mesmo grau de exactidão em termos de resultado final, parâmetros de entrada e saída com o mesmo número de *bits* para os diferentes métodos em estudo, ...

Durante a execução de todos os testes considerou-se parâmetros de entrada e saída com $n = 24$ bits, para parâmetros de saída e entrada. Para além disso devido à simplicidade deste teste optou-se por utilizar uma *FPGA* de baixo custo – *Spartan3E*, *device XC3S500E* e *package FG320*.

B.3. Características básicas

Para avaliar os recursos utilizados pelos diferentes métodos em estudo, colocou-se, por uma questão de coerência, um registo à saída do primeiro e terceiro métodos, de modo a termos condições de teste semelhantes ao segundo método utilizado, uma vez que o bloco de divisão fornecido pela *Xilinx* possui um registo à saída. Além disso, tentou garantir-se que os parâmetros à saída e entrada dos blocos de divisão apresentassem o mesmo número de *bits*.

Relativamente ao cálculo do segundo parâmetro em avaliação (frequência máxima de funcionamento) procedeu-se à colocação de um registo à entrada de cada um dos módulos utilizados. Os resultados encontram-se apresentados na Tabela XXVI.

Tabela XXVI – Frequência máxima de funcionamento e recursos consumidos.

		B.M.	B.D.X.	B.D.C.
Recursos consumidos	Número de LUTs	149	751	183
	Número de <i>Slices</i> ocupadas	76	985	94
	Número de <i>Flip-Flops</i>	48	1883	26
	Número de <i>input/output buffers</i>	73	74	73
	Número de <i>BUFGMUXs</i>	1	1	1
	Número de <i>MULT18X18SIOs</i>	4		
Frequência máxima de funcionamento (MHz)		55,984	168,095	35,740

Legenda:

B.M. – *Bloco Multiplicador;*

B.D.X. – *Bloco de Divisão da Xilinx;*

B.D.C. – *Bloco de Divisão Combinatória.*

B.4. Erro relativo

De modo a que os diferentes métodos em estudo estivessem em igualdade de circunstâncias ao nível do grau de exactidão da operação de divisão, procedeu-se à utilização, ao nível do método do multiplicador e do divisor combinatório, de técnicas de arredondamento que permitissem reduzir o erro relativo introduzido por cada um destes métodos no resultado final. Os resultados obtidos derivam do erro relativo associado à divisão inteira por cada um dos métodos em estudo (vide Tabela XXVII).

Tabela XXVII – Erro relativo.

Dividendo	Quociente			Erro relativo (%)		
	B.M.	B.D.X.	B.D.C.	B.M.	B.D.X.	B.D.C.
-8388608	-2796204	-2796203	0	3,58E-5	0	100
-4194304	-1398103	-1398101	-1398101	0,014	0	0
-1047517	-349172	-349172	-349172	0	0	0
-983037	-327679	-327679	-327679	0	0	0
-3	-1	-1	-1	0	0	0
3	1	1	1	0	0	0
6	2	2	2	0	0	0
31	10	10	10	0	0	0
42	14	14	14	0	0	0
55	18	18	18	0	0	0
66	22	22	22	0	0	0
101	33	33	33	0	0	0
304	101	101	101	0	0	0
2572	857	857	857	0	0	0
36883	12294	12294	12294	0	0	0
4194304	1398101	1398101	1398101	0	0	0
8388607	2796202	2796202	0	0	0	100

Legenda:

B.M. – Bloco Multiplicador;

B.D.X. – Bloco de Divisão da Xilinx;

B.D.C. – Bloco de Divisão Combinatória.

B.5. Vantagens e Desvantagens

Através da análise dos resultados obtidos, é possível concluir que o primeiro método, multiplicação por $1/3$, apresenta uma frequência máxima de funcionamento razoável (cerca de 55,984 MHz, que é mais elevada do que a existente no terceiro método) e sob o ponto de vista de recursos, não é o mais dispendioso (1 *BUFGMUXs*, 73 *input/output buffers*, 48 *flip-flops* e 149 *LUTs* de quatro entradas). Contudo, este método implica a utilização de um recurso que condiciona a sua utilização: os multiplicadores — 4 multiplicadores que não são utilizados a 100 % por parte da *FPGA*. A utilização deste recurso condiciona o número de divisores deste tipo que pode existir no nosso projecto (a *FPGA* está condicionada a 20 multiplicadores). Este método possui ainda uma segunda desvantagem, que advém do facto de necessitar de um algoritmo de arredondamento, de modo a diminuir o erro relativo proveniente da aproximação de $1/3$ utilizada.

O segundo método, bloco de divisão fornecido pelo compilador da *Xilinx*, é aquele que não necessitou de nenhum *software* adicional, de modo a melhorar o erro relativo associado à operação de divisão. Além disso, foi o que apresentou a maior frequência máxima de funcionamento. No entanto, esta alternativa implica a utilização de uma quantidade de recursos (ao nível de *LUTs* de quatro entradas, *flip-flops* e *input/output buffers*) muito maior que os outros dois métodos, o que limita muito a sua utilização.

Finalmente, o terceiro método, que consiste no bloco de divisão combinatória, sob o ponto de vista de recursos, é um dos menos exigentes (183 *LUTs* de quatro entradas, 1 *BUFGMUXs*, 73 *input/output buffers* e 26 *flip-flops*). Este método é meramente combinatório e apenas requer a utilização de *LUTs* para efectuar a operação de divisão. Mas, sob o ponto de vista de frequência máxima de funcionamento, este método é o menos fiável dos três apresentados. Além disso, devido à própria arquitectura interna do bloco de divisão combinacional, este encontra-se limitado a uma gama de valores de funcionamento. Isto é, como o quociente internamente possui menos 2 *bits* que o dividendo, logo, para números elevados, a divisão deixa de ser efectuada correctamente, resultando um erro relativo

elevado. Outra limitação deste método consiste no facto de, em relação aos restantes, apenas admitir operações com números *unsigned*, o que implicou a utilização de um algoritmo que entrasse em consideração com este ponto.

B.6. Bloco de divisão seleccionado

Considerando os prós e contras de cada um dos métodos, conclui-se que o método que apresenta o melhor desempenho, sob o ponto de vista dos diferentes parâmetros avaliados, é o primeiro método (Multiplicação por $1/3$), ainda que seja aquele que implique a utilização de multiplicadores da *FPGA*. No entanto, a escolha do melhor método para a aplicação em causa vai depender necessariamente do facto desta implicar uma frequência de funcionamento alta ou baixa.

Apêndice C Código em *VHDL*

Este apêndice descreve o conjunto de descrições criadas em *VHDL* que permitiram a construção da unidade do Meshotron na topologia seleccionada. A Tabela XXVIII, Tabela XXIX, a Tabela XXX, a Tabela XXXI e a Tabela XXXII identificam os ficheiros de código que constituem e constituíram o Meshotron e as suas respectivas descrições.

Tabela XXVIII – Ficheiros de código presentes no topo da hierarquia.

Ficheiro	Descrição do código
MeshotronUnit_biMem.vhd	Unidade do Meshotron.
MeshotronModule_biMem.vhd	Neste ficheiro são descritos os diversos modelos testados e simulados para validação do Meshotron.

Tabela XXIX – Ficheiros de código presentes na unidade do Meshotron.

Ficheiro	Descrição do código
AddressSelect_memP.vhd	Elemento que efectua a selecção do endereço de memória onde se efectua a inicialização.
DriverAddress.vhd	Elemento que efectua o endereçamento dos diversos estados e fornece os respectivos sinais de controlo dos diferentes elementos de armazenamento do Meshotron.
InitializeData.vhd	Elemento que coloca o valor de inicialização nas memórias definidas como entrada.
FifoDualClock_biMem.xco	<i>FIFO dual-clock</i> descrita na secção 3.2.1.3.
memRAM.vhd	Banco de memória descrito na secção 3.2.1.2.
Mux21.vhd	Elemento que controla o fluxo de dados para os bancos de memória, de acordo com o estado em execução.
ScatteringModule.vhd	Módulo de <i>scattering</i> descrito na secção 3.2.1.4.
ShiftRegisterN.vhd	<i>Shift-register</i> genérico que gera os Clock Enables (vide secção 3.2.4).
StateMachineEnable.vhd	Máquina de estados que garante que o sinal de <i>overflow</i> apenas activa quando surgir o primeiro estado S.
StateMachineInOut.vhd	Máquina de estados que comuta entre o valor 0 e 1 para definição de qual a memória de entrada e saída em cada iteração actual (vide Fig. 36).
StateMachineValid.vhd	Máquina de estados que fixa o sinal de overflow a 1 quando este é detectado.

Tabela XXX – Ficheiros de código presentes no *driver* de endereçamento (*DriverAddress.vhd*).

Ficheiro	Descrição do código
CounterIteration.vhd	Elemento que conta o número de iterações a executar pelo Meshotron.
CounterN.vhd	Contador genérico de n bits presente em cada um dos módulos de endereçamento.
ControlUnitEntity.vhd	Unidade de controlo descrita na secção 3.4.4.
Initialize.vhd	Módulo de endereçamento da fase de inicialização (vide secção 3.2.2.2.1).
MemorySet1.vhd	Módulo de endereçamento do <i>delay pass 1</i> e <i>delay pass 3</i> (vide secção 3.2.2.2.3.1).
Mux31Optimize.vhd	Elemento que efectua o controlo do endereçamento de acordo com o estado em execução (vide secção 3.4.3)
ScatteringDelay12_address.vhd	Módulo de endereçamento do <i>scattering/delay pass 12</i> (vide secção 3.4.2.2).
StateMachine.vhd	Máquina de estados que efectua a sequência de controlo geral do Meshotron (vide secção 3.4.2.1)
StateMachine1Bit.vhd	Máquina de estados de um bit que fica activa quando o número de unidades de <i>scattering</i> existentes é atingido. Este elemento efectua o atraso de d unidades no contador mais significativo presente no módulo de endereçamento do estado <i>SD12</i> (vide secção 3.4.1.2).

Tabela XXXI – Ficheiros de código presentes no módulo de *scattering* (*Scattering Module.vhd*).

Ficheiro	Descrição do código
ScatteringUnit.vhd	Unidade de <i>scattering</i> descrita na secção 3.2.1.6.
RegistN_scat.vhd	Registo de <i>n bits</i> utilizado para registar a entrada e saída da unidade de <i>scattering</i> .
Adder.vhd	Somador de <i>n bits</i> .
Subtraction.vhd	Subtractor de <i>n bits</i> .
MultiplierBlock.vhd	Bloco de divisão descrito no Apêndice B.
DetectorOverflow.vhd	Detector de <i>overflow</i> descrito na secção 3.2.1.6.2.
ABS_number.vhd	Elemento que permite obter o valor do módulo do dividendo, de modo a facilitar a operação de divisão.
MultNew.vhd	Elemento que corresponde a um multiplicador por uma dada constante, respectivamente 1/3.
ErrorCorrection.vhd	Elemento que efectua o arredondamento do resultado da divisão com base nos três <i>bits</i> mais significativos da parte fraccionária resultante da multiplicação por 1/3.
SignedNumber.vhd	Elemento que restitui o sinal original do número após a operação de divisão.

Tabela XXXII – Ficheiros de código não utilizados.

Ficheiro	Descrição do código
DecoderN.vhd	Elemento que selecciona qual a unidade de <i>scattering</i> a ler ou a escrever (vide secção 3.2.1.4).
ScatteringPass_address.vhd	Módulo de endereçamento do <i>scattering pass</i> otimizado descrito na secção 3.3.2.
MemorySet2.vhd	Módulo de endereçamento do <i>delay pass 2</i> descrito na secção 3.2.2.3.2.

A Fig. 78 ilustra como se encontra organizada a estrutura hierárquica dos ficheiros de código criados em *VHDL* que compõem uma unidade do Meshotron.

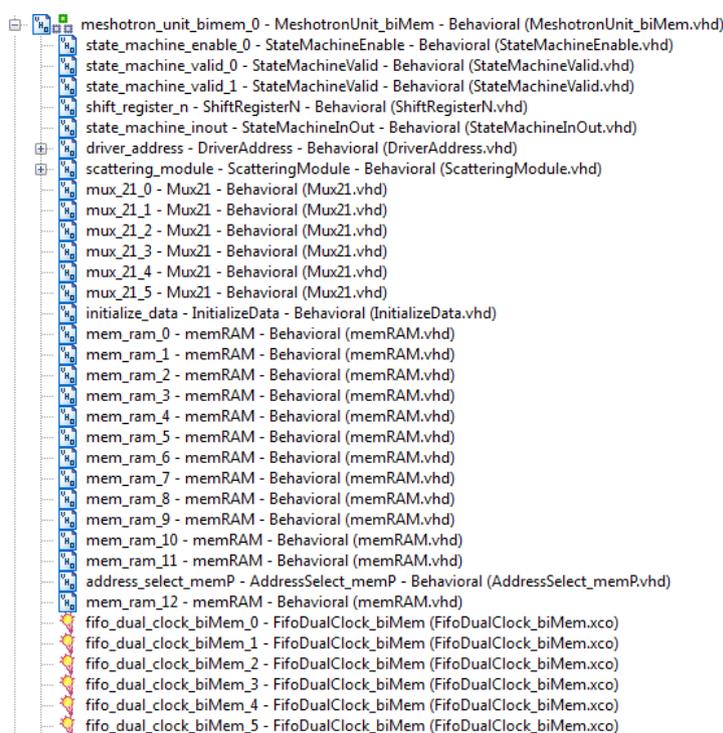


Fig. 78 – Organização da estrutura hierárquica dos ficheiros de código constituintes da unidade do Meshotron.

A Fig. 79 e a Fig. 80 ilustram como se encontra organizada a estrutura hierárquica dos ficheiros de código de maior relevância no Meshotron, respectivamente a *driver* de endereços e o módulo de *scattering*.

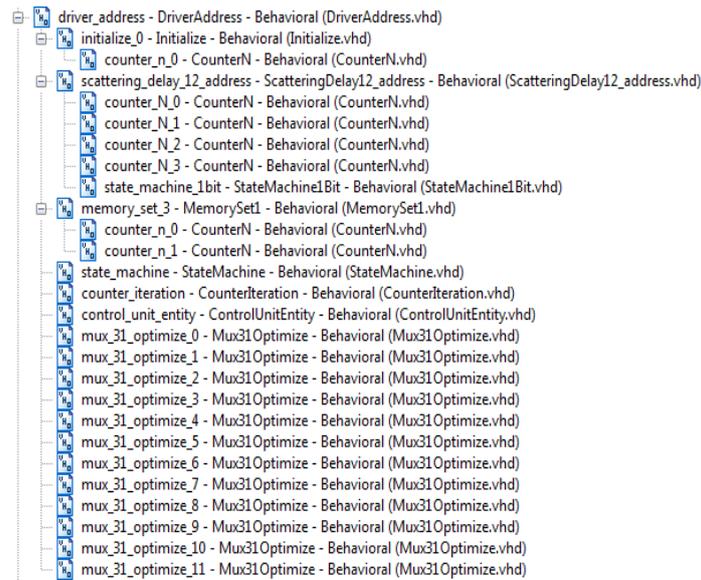


Fig. 79 – Organização da estrutura hierárquica dos ficheiros de código constituintes da *driver* de endereços.

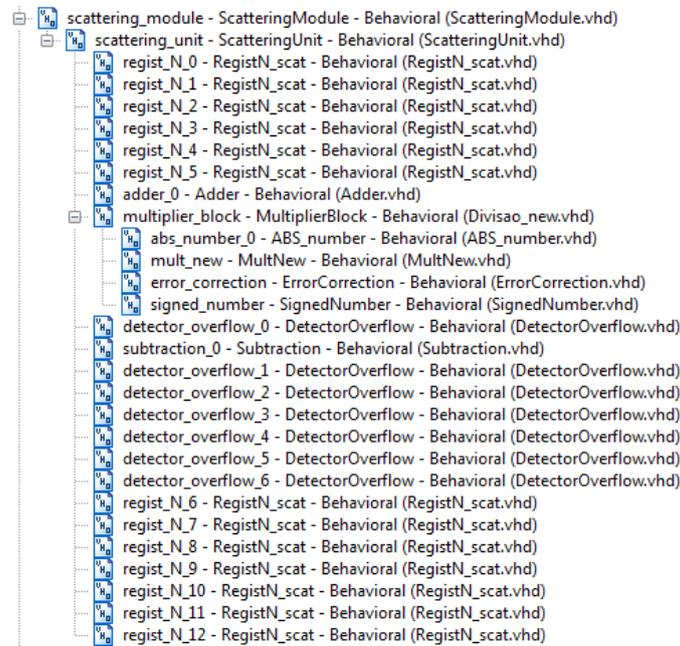


Fig. 80 - Organização da estrutura hierárquica dos ficheiros de código constituintes da unidade de *scattering*.

Para mais detalhes sobre o código criado em *VHDL* deverá aceder ao directório *Código de VHDL/Meshotron - Versão actual no CD* de anexo.

Apêndice D Código em *Matlab*

Este apêndice descreve o conjunto de programas criados em *Matlab* que permitiram a especificação e validação da unidade do Meshotron. A Tabela XXXIII identifica os principais programas criados em *Matlab* com a sua respectiva descrição.

Tabela XXXIII – Lista de programas criados em *Matlab*.

Programa	Descrição
DWMmodel_oneUnit.m	Programa de modelação <i>DWM-3D</i> em inteiros de 32 <i>bits</i> com uma unidade do Meshotron utilizado para validação do modelo em teste na secção 4.2.2.1 e 4.2.5.
DWMmodel_twoUnits.m	Programa de modelação <i>DWM-3D</i> em inteiros de 32 <i>bits</i> com duas unidades do Meshotron utilizado para validação do modelo em teste na secção 4.2.3.
DWMmodel_fourUnits.m	Programa de modelação <i>DWM-3D</i> em inteiros de 32 <i>bits</i> com quatro unidades do Meshotron utilizado para validação do modelo em teste na secção 4.2.4.
Model_1_test.m	Programa que efectua a comparação entre as <i>RIRs</i> obtidas pelo programa de modelação <i>DWM-3D</i> e pelo <i>Chipscope</i> para o modelo com uma unidade do Meshotron (vide secção 4.4).
Model_2_test1.m	Programa que efectua a comparação entre as <i>RIRs</i> obtidas pelo programa de modelação <i>DWM-3D</i> e pelo <i>Chipscope</i> para o modelo com duas unidades do Meshotron, quando o sinal de relógio de ambas as unidades é sujeito a uma variação (vide secção 4.4.3.1).
Model_2_test2.m	Programa que efectua a comparação entre as <i>RIRs</i> obtidas pelo programa de modelação <i>DWM-3D</i> e pelo <i>Chipscope</i> para o modelo com duas unidades do Meshotron, quando é variada a coordenada do nó emissor (vide secção 4.4.3.2).
RIR_calc.m – <i>function</i> DWMdouble.m - <i>function</i>	Funções baseadas no programa de modelação <i>DWM-3D</i> em <i>double</i> com uma unidade do Meshotron utilizado para validação do modelo em teste na secção 4.4.2.
RIR_calc2unit.m - <i>function</i>	Função baseada no programa de modelação <i>DWM-3D</i> em <i>double</i> com duas unidades do Meshotron utilizado para validação do modelo em teste na secção 4.4.3.
DWM32.m - <i>function</i>	Função baseada no programa de modelação <i>DWM-3D</i> em inteiros de 32 <i>bits</i> com uma unidade do Meshotron.
DWM16.m - <i>function</i>	Função baseada no programa de modelação <i>DWM-3D</i> em inteiros de 16 <i>bits</i> com uma unidade do Meshotron.
DWMmain.m	Programa de especificação do número de <i>bits</i> a utilizar no barramento de dados do Meshotron (vide secção 4.2.1).

Para mais detalhes sobre o código criado em *Matlab* dever-se-á consultar a pasta Código em Matlab no CD em anexo.