Universidade Federal de Juiz de Fora

Graduação em Engenharia Elétrica

Habilitação em Robótica e Automação Industrial

Kinn Belarmino Batista Dias

Implementação em FPGA de Filtro FIR Utilizando Máquina de Estados Para Estimação Online de Energia no Calorímetro de Telhas

Juiz de fora

2021

Kinn Belarmino Batista Dias

Implementação em FPGA de Filtro FIR Utilizando Máquina de Estados Para Estimação Online de Energia no Calorímetro de Telhas

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica, área de concentração: Robótica e Automação Industrial, da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro Eletricista.

Orientador: Luciano Manhães de Andrade Filho, D.Sc.

Juiz de Fora

2021

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Dias, Kinn Belarmino Batista.
Implementação em FPGA de Filtro FIR Utilizando Máquina de Estados Para Estimação Online de Energia no Calorímetro de Telhas. / Kinn Belarmino Batista Dias. -- 2021.
79 p.

Orientador: Luciano Manhães de Andrade Filho
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2021.

1. Processamento online. 2. Calorimetria. 3. Reconstrução de energia. 4. FPGA. 5. Máquina de Estados. I. Andrade Filho, Luciano Manhães de, orient. II. Título.

Universidade Federal de Juiz de Fora
Faculdade de Engenharia
Engenharia Elétrica – Robótica e Automação Industrial

ATA DE APRESENTAÇÃO DE TRABALHO FINAL DE CURSO

DATA DA DEFESA: **10 / 09 / 2021**

CANDIDATO: **KINN BELARMINO BATISTA DIAS**

ORIENTADOR: **PROF. LUCIANO MANHÃES DE ANDRADE FILHO**

TÍTULO DO TRABALHO: **Implementação em FPGA de Filtro FIR Utilizando Máquina de Estados para Estimação Online de Energia no Calorímetro de Telhas**

BANCA EXAMINADORA/INSTITUIÇÃO:

PREDIDENTE: **PROF. LUCIANO MANHÃES DE ANDRADE FILHO / UFJF**

AVALIADOR 1: **PROF. EDER DE BARBOZA KAPISCH / UFJF**

AVALIADOR 2: **D.Sc. JOÃO BITTENCOURT DA SILVEIRA DUARTE / UFJF**

LOCAL: **POR WEBCONFERÊNCIA, CONFORME RESOLUÇÃO Nº 24/2020-CONSU**

Nesta data, em sessão pública, após exposição oral de **30** minutos, o candidato foi arguido pelos membros da banca. Em decorrência desta arguição, a banca considerou o candidato:

(X) APROVADO

( ) REPROVADO

Na forma regulamentar foi lavrada a presente Ata que é abaixo assinada pelos membros da banca na ordem determinada e pelo candidato:

PREDIDENTE: Luciano M. de A. Filho

AVALIADOR 1: Eder Barboza Kapisch

AVALIADOR 2: João Luiz Bittencourt S. Duarte

CANDIDATO: Kinn Belarmino Batista Dias

Dedico este trabalho à minha avó Amélia...

AGRADECIMENTOS

Agradeço, primeiramente, à minha família, pelo alicerce que me deram todos esses anos para que eu pudesse concluir meu curso sem outras preocupações e desafios maiores, apesar das dificuldades. Principalmente ao meu pai que sempre esteve comigo, me apoiando e me empurrando quando eu achava que não conseguiria, e minha prima Marcela que foi minha inspiração e meu porto seguro, desde sempre e ainda mais depois da partida da nossa avó.

Agradeço à minha avó Amélia que, apesar de não estar mais aqui entre nós, sempre me incentivou em todos os meus sonhos e almejava estar presente nesse momento. Infelizmente, não foi possível tê-la aqui, mas sei que ela está sorrindo e comemorando em algum lugar deste vasto e infinito universo.

Aos amigos que fiz no decorrer deste percurso, sem os quais sei que tudo teria sido muito mais difícil. Citar todos não caberia aqui, mas um agradecimento especial ao Átila, que me acompanhou nas estradas e empreitadas da música, ao Thiago, que esteve comigo dentro e fora das salas de aula, me aconselhando e sofrendo comigo as adversidades, à Carla, que sempre fez planos para o meu futuro, mesmo eu não tendo muitas perspectivas para ele, e ao Matheus, ao Wilker, ao Neim, o Milo, o Celim e todos os demais que estiveram comigo, me perdoem se esqueci de alguém, mas são todos muito especiais.

Agradeço ao João que, apesar das dificuldades que eu mesmo criei, sempre teve paciência para me ajudar e me empurrar para frente, além de me auxiliar no que fosse preciso, mesmo quando ele estava ocupado ou tendo que lidar com seus próprios problemas. Esse trabalho só foi possível graças a ele.

Ao professor Luciano, que nunca desistiu de mim quando tantos outros já haviam desistido, que me orientou de forma primorosa e me inseriu nesse grupo de pessoas incríveis que é o NIPS CERN e que me fez aprender mais sobre esse incrível ambiente que é a física de altas energias e o CERN.

“Desejos morrem diante das perdas e contrariedades, sonhos criam raízes nas dificuldades.”

Augusto Cury

RESUMO

Este trabalho apresenta um estudo sobre formas de implementação de filtros FIR para a reconstrução *online* de energia no primeiro nível de *trigger* do calorímetro hadrônico (TileCal) do ATLAS. Com a perspectiva das atualizações previstas para acontecerem nos próximos anos no colisor de partículas LHC, que é atualmente o maior acelerador de partículas do mundo, haverá o aumento da energia das colisões, visando o aumento da probabilidade de haverem eventos cada vez mais raros. Com isso, as chances de ocorrerem sobreposição das informações de energia geradas devido à ocorrência de colisões adjacentes são enormes, o que promoverá o efeito de empilhamento de sinais, ou *pile-up*. Muitos métodos iterativos de deconvolução de sinais baseados em filtros FIR vêm sendo propostos, apresentando resultados satisfatórios. No entanto, as colisões geram um número de dados exorbitante, o que traz um desafio para os algoritmos atuais de reconstrução de energia: utilizar a menor quantidade de recursos de *hardware* possível, sem perder o desempenho do método e com o menor erro RMS possível, gerando o mesmo resultado final. Esses métodos necessitam de uma implementação *online*. Os dispositivos lógicos programáveis, ou FPGAs, são os escolhidos para implementação desses métodos, por se tratarem de instrumentos reconfiguráveis e de alta velocidade. Assim, neste trabalho se propõe um método de implementação para o filtro FIR baseado em uma máquina de estados, a qual atuará realizando os cálculos das informações de energia das partículas e reduzirá os recursos computacionais empregados no processo.

Palavras-chave: Processamento *online*, Calorimetria, Reconstrução de energia, FPGA, Máquina de Estados.

ABSTRACT

This work presents a study about forms of implementing FIR filters to be used for the online energy reconstruction in the first trigger level of the hadronic calorimeter (TileCal) at the ATLAS. With the prospect of planned upgrades happening in the next few years at the LHC collider, which is currently the largest collider in the world, the energy of the collisions will increase, aiming to have more probability of rare events occurring. Thereby, the chances of having energy information overlap, because of the adjacent collisions, are big, causing the pile-up effect. Many iterative methods based on signal deconvolution with FIR filters have been proposed, showing satisfying results. However, the collisions create a huge amount of data, which brings a challenge to current energy reconstruction algorithms: use as little hardware resources as possible, without losing the method performance and with the least amount of RMS error possible, giving the same final result. These methods require an online implementation. The programmable logic devices, or FPGAs, are the chosen integrated circuits to the implementation of these methods, because they are reconfigurable and have high speed. Thus, in this work will be proposed an implementation form to FIR filters based on state machines, which will focus on performing the particle energy information calculations and reduce the computational resources that are being used on the process.

Keywords: Online processing, Calorimetry, Energy reconstruction, FPGA, State Machine.

LISTA DE FIGURAS

Figura 1 - Mapa mostrando a localização do CERN..... 22

Figura 2 - Visão Geral do LHC..... 24

Figura 3 - O complexo do LHC com todos os seus detectores e subdetectores…. 25

Figura 4 - O detector ATLAS e seus subsistemas..... 26

Figura 5 - Sistema de calorimetria do ATLAS..... 27

Figura 6 - Interação das partículas com os subdetectores do ATLAS..... 28

Figura 7 - Esquema de um módulo do TileCal..... 29

Figura 8 - Formato de um pulso característico do TileCal..... 30

Figura 9 - Níveis de filtragem do sistema de trigger do ATLAS..... 32

Figura 10 - Linha do tempo do programa de atualizações do LHC..... 33

Figura 11 - Efeito *pile-up* no TileCal..... 34

Figura 12 - Comparação entre um sinal ideal e um com efeito *pile-up*..... 35

Figura 13 - Exemplo de kit de desenvolvimento de lógica programável FPGA..... 37

Figura 14 - Diagrama do Processador SAPHO..... 40

Figura 15 - Diagrama de uma Máquina de Estados de Moore..... 41

Figura 16 - Diagrama de exemplo de uma Máquina de Estados de Moore..... 41

Figura 17 - Diagrama de uma Máquina de Estados de Mealy..... 43

Figura 18 - Diagrama de exemplo de uma Máquina de Estados de Mealy…..... 44

Figura 19 - Estruturas de representação de um filtro FIR..... 48

Figura 20 - Estrutura de representação de um filtro FIR em cascata..... 48

Figura 21 - Representação em blocos do calorímetro como um sistema linear..... 49

Figura 22 - Modelagem Inversa de Filtros Adaptativos..... 51

Figura 23 - Primeiras linhas de código e inicialização dos coeficientes do filtro FIR..... 54

Figura 24 - Primeiro estágio da máquina de estados..... 55

Figura 25 - Ilustração de um shift register de 4 bits..... 56

Figura 26 - Segundo estágio da máquina de estados..... 56

Figura 27 - Terceiro estágio da máquina de estados..... 57

Figura 28 - Último estágio da máquina de estados..... 58
Figura 29 - *Top Level* demonstrativo do filtro FIR por Máquina de Estados..... 59
Figura 30 - Gráfico do erro RMS variando com o número de bits..... 62
Figura 31 - Gráfico da relação entre número de bits, o erro RMS e recursos lógicos..... 63
Figura 32 - Inicialização dos coeficientes do filtro FIR com 16 bits..... 65
Figura 33 - Relatório de compilação da implementação por Máquina de Estados. 65
Figura 34 - Código de implementação de filtro FIR na forma direta..... 66
Figura 35 - Relatório de compilação da implementação na forma direta..... 66
Figura 36 - Código de implementação de filtro FIR na forma transposta..... 67
Figura 37 - Relatório de compilação da implementação na forma transposta..... 68

LISTA DE TABELAS

Tabela 1 - Tabela verdade do exemplo da Máquina de Moore..... 42

Tabela 2 - Tabela verdade do exemplo da Máquina de Mealy..... 44

Tabela 3 - Análise do número de bits e elementos lógicos com 2^{16}..... 61

Tabela 4 - Relação entre número de bits, o erro RMS e os recursos utilizados..... 63

LISTA DE ABREVIATURAS E SIGLAS

ALICE	*A Large Ion Collider Experiment*
ATLAS	*A Toroidal LHC ApparatuS*
BC	Cruzamento de Feixes, (do inglês, *Bunch Crossing*)
CERN	(do francês, *Conseil Européen pour la Recherche Nucléaire*)
CMS	*Compact Muon Solenoid*
CTP	Processador Central de Filtragem, (do inglês, *Central Trigger Processor*)
DSP	Processador Digital de Sinais, (do inglês, *Digital Signal Processor*)
EF	Filtro de eventos, (do inglês *Event Filter*)
EMEC	*LAr Electromagnetic End-cap*
FCal	*LAr Forward Calorimeter*
FIR	Resposta ao Impulso Finita, (do inglês, *Finite Impulse Response*)
FPGA	*Field Programmable Gate Array*
FPL	*Field-Programmable Logic*
HEC	*LAr Hadronic End-cap*
HL-LHC	*High-Luminosity Large Hadron Collider*
HLT	Trigger de alto nível, (do inglês, *High Level Trigger*)
ID	Detector de Traços (do inglês, *Inner Detector*)
IEEE	*The Institute of Electrical and Electronics Engineers*
IIR	*Infinite Impulse Response*
L1	Nível 1, (do inglês, *Level* 1)
L2	Nível 2, (do inglês, *Level* 2)
LArg	Calorímetro Eletromagnético de Argônio Líquido, (do inglês *Liquid Argon Calorimeter*)
LHC	Grande Colisionador de Hádrons, (do inglês, *Large Hadron Collider*)
LHCb	*Large Hadron Collider beauty experiment for precision measurements of CP-violation and rare decay*
LHCf	*Large Hadron Collider forward*
LTI	Linear e Invariante no Tempo, (do inglês, *Linear Time- Invariant*)
MATLAB	*Matrix Laboratory*
MF	Filtro Casado, (do inglês, *Matched Filter*)

NIPS	Núcleo de Instrumentação e Processamento de Sinais
PD	*Pixel Detector*
PMT	Tubo Foto-multiplicador, (do inglês, *Photo Multiplier Tube*)
RMS	Raiz quadrada média, (do inglês, *Root Mean Square*)
RoI	Regiões de Interesse, (do inglês, *Regions of Interest*)
SAPHO	*Scalable Architecture Processor for Hardware Optimization*
SISO	*Single-Input Single-Output*
SPS	*Synchrotron Super Proton*
ST	*Semiconductor Tracker*
TileCal	Calorímetro Hadrônico de Telhas, (do inglês, *Tile Calorimeter*)
TOTEM	*Total Elastic and Diffractive Cross Section Measurement*
TRT	*Transition Radiation Tracker*
UFJF	Universidade Federal de Juiz de Fora
VHDL	*Verilog Hardware Description Language*

SUMÁRIO

1	**INTRODUÇÃO**	17
1.1	CONTEXTUALIZAÇÃO E MOTIVAÇÃO	17
1.2	OBJETIVOS	19
1.3	METODOLOGIA	20
1.4	ESTRUTURA DO TRABALHO	20
2	**O CERN, ACELERADOR LHC E O EXPERIMENTO ATLAS**	22
2.1	O CERN	22
2.2	O LHC	23
2.3	O EXPERIMENTO ATLAS	25
2.3.1	**O Calorímetro Hadrônico**	28
2.3.2	**O Sistema de *Trigger* do ATLAS**	30
2.4	A ATUALIZAÇÃO DO ATLAS	32
3	**REVISÃO BIBLIOGRÁFICA**	36
3.1	O FPGA	36
3.2	O PROCESSADOR SAPHO	39
3.3	MÁQUINAS DE ESTADOS	40
3.2.1	**O modelo de Moore**	40
3.2.2	**O modelo de Mealy**	43
3.4	FILTROS FIR E IIR	45
4	**TÉCNICAS PARA ESTIMAÇÃO DE ENERGIA NO L1**	47
4.1	IMPLEMENTAÇÃO DE FILTROS FIR	47
4.1.1	**Filtros FIR para sinais estocásticos**	50
4.1.1.1	Modelagem Inversa	50
4.1.1.2	Estimação utilizando o método dos Mínimos Quadrados	51

5 IMPLEMENTAÇÃO..... 54

6 RESULTADOS..... 60
6.1 SIMULAÇÃO E ANÁLISE DO NÚMERO DE BITS..... 61
6.1.1 Análise da implementação da máquina por estados..... 64
6.1.2 Análise da implementação na forma direta..... 65
6.1.3 Análise da implementação na forma transposta..... 67
6.2 DISCUSSÃO..... 68

7 CONSIDERAÇÕES FINAIS..... 70
7.1 CONCLUSÕES..... 70
7.2 TRABALHOS FUTUROS..... 70

REFERÊNCIAS..... 71

APÊNDICE A - ALGORITMOS IMPLEMENTADOS NO QUARTUS.... 76

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Atualmente, tem sido observada uma evolução considerável no campo da tecnologia, mais especificamente no ramo da instrumentação eletrônica, onde, cada vez mais, sensores geram quantidades abundantes de dados para processamento e tratamento, conforme os mesmos vão sendo desenvolvidos e aperfeiçoados. Aliado a esse contexto, há a necessidade de processar esses sinais de forma quase instantânea, ou seja, *online* como se é comumente chamado.

Uma aplicação desafiadora desses avanços se dá no campo da Física de Altas Energias, que estuda as interações das menores partes constituintes da matéria. Por se tratarem de elementos e fenômenos de dimensões mínimas e velocidades altíssimas, tais experimentos demandam grande número de sensores, operando com elevada quantidade de eventos para serem processados.

No contexto supracitado, destaca-se o trabalho executado no LHC (***L**arge **H**adron **C**ollider*), o colisor de partículas localizado na fronteira entre a França e a Suíça, o qual faz parte do complexo de aceleradores da Organização Europeia para a Pesquisa Nuclear, o CERN (***C**onseil **E**uropéen pour la **R**echerche **N**ucléaire*), fundado em 1954 [1]. Esse colisor é considerado a maior máquina já construída pela humanidade e nele ocorrem os principais experimentos na área de Física de Altas Energias. As colisões de partículas em máquinas como essa geram uma enorme quantidade de energia, permitindo observar fenômenos nunca antes vistos pela humanidade.

Colisores, ou aceleradores de partículas, são máquinas projetadas, como o próprio nome sugere, para acelerar partículas, através de campos elétricos e magnéticos, e posteriormente colidi-las, orientando e focalizando feixes em altíssima velocidade, resultando na produção de novas partículas [2]. Estas, para serem observadas experimentalmente, precisam ter suas amplitudes de energia reconstruídas a partir de sensores específicos. Estes instrumentos são denominados calorímetros. Estes são altamente segmentados estruturalmente, na ordem das

milhares de partes, a fim de proporcionar resolução espacial da deposição da energia. Devido a essa segmentação e a velocidade em que ocorrem os eventos físicos, as colisões geram números exorbitantes de dados, pois a elevada quantidade de sensores deve ser lida de forma simultânea. O processamento e filtragem desses dados será feito de duas formas: *online* e *offline*, através de um sistema chamado de *trigger* [3]. A primeira forma escolhe as informações no momento em que elas ocorrem, de forma quase instantânea, respeitando o atraso do sistema. Já a segunda, analisa dados armazenados de colisões passadas [4].

No contexto do CERN e do LHC, destaca-se o experimento ATLAS (***A*** *toroidal* ***L****HC* ***Apparatus***), que possui subdetectores em camadas cilíndricas em torno do ponto de colisão. Dentre esses subdetectores existem os chamados calorímetros, os quais podem ser de dois tipos: o eletromagnético, mais interno, também conhecido como Calorímetro de Argônio Líquido LAr (***L****iquid* ***Arg****onic Calorimeter*), e o hadrônico ou calorímetro de telhas TileCal (***Tile Cal****orimeter*) [5]. O primeiro funciona medindo a energia de partículas oriundas de colisões que interagem eletromagneticamente como fótons e elétrons, por exemplo. O segundo mede a energia de hádrons, partículas que mantém sua coesão interna devido às interações fortes, que atravessam o LAr, por este ser menos denso, por isso ele é chamado de calorímetro hadrônico.

No *TileCal*, que é o foco principal do presente trabalho, ocorre a interação das partículas resultantes das colisões com as células do calorímetro, visando a reconstrução das informações sobre a energia das partículas captadas, a qual é uma característica muito importante a ser mensurada [5]. No entanto, pelo fato da quantidade de informações ser volumosa e gerada em um instante de tempo muito pequeno, pode surgir o efeito de empilhamento de sinais ou *pile-up*. Visando a resolução desse problema, foram propostos métodos baseados na deconvolução desses sinais a fim de recuperar as informações de amplitude da energia [4, 6].

O efeito *pile-up* se dá pelo fato da taxa em que as colisões ocorrem ser menor do que o tempo de resposta dos calorímetros, havendo assim, uma sobreposição dos sinais, provocando um aumento do erro na estimação de amplitude de energia [4, 7].

A implementação das técnicas de deconvolução para aplicação *online*, que requer o processamento ininterrupto (do inglês *free-running*), pode ser realizada de diversas formas, como por exemplo, por meio de algoritmos janelados com o auxílio de conversores paralelo e serial, através de filtros IIR (***I**nfinite **I**mpulse **R**esponse*) ou mediante ao uso de filtros FIR (***F**inite **I**mpulse **R**esponse*) [8]. Os métodos que utilizam implementações com filtros digitais simples do tipo FIR têm sido comumente utilizados na reconstrução *online* dos sinais em calorimetria com bastante eficácia, pela sua simplicidade e ausência de realimentações, que podem provocar instabilidade em implementações realizadas em ponto fixo.

Normalmente implementados em FPGA (***F**ield **P**rogrammable **G**ate **A**rray*), nos experimentos de física de altas energias, os filtros FIR são usualmente estruturados em suas formas direta ou transposta. Na segunda, quebram-se as operações em circuitos combinacionais menos extensos, tornando-a a mais recomendada para implementação em FPGA. Contudo, com a crescente demanda dos sensores e as atualizações que ocorrerão no LHC, ideias cada vez menos custosas, em termos de recursos lógicos para o processamento envolvido nas operações do filtro FIR, necessitam ser exploradas.

1.2 OBJETIVOS

O objetivo do presente trabalho é apresentar um método de implementação de um algoritmo de deconvolução para reconstrução de energia, baseado em filtros FIR, de modo a se obter uma resposta que apresente melhor otimização na utilização de recursos de *hardware*, se comparados aos métodos tradicionalmente aplicados atualmente. Mais especificamente em FPGA, implementar uma máquina de estados como núcleo para realizar operações simples desse filtro, de modo a investigar a possibilidade de se economizar recursos lógicos ao longo de todo o processo.

1.3 METODOLOGIA

A metodologia de trabalho consiste basicamente de utilizar-se das estruturas já desenvolvidas para filtros FIR, as quais já são utilizadas no cálculo da reconstrução de energia no TileCal e possuem um bom desempenho, e aplicá-las com um processador em FPGA, utilizando-se uma implementação alternativa, baseada na teoria de máquina de estados. Sendo assim, busca-se melhorar ainda mais esse desempenho, tornando o processo menos custoso e abrindo a possibilidade de se utilizar mais sensores em uma única FPGA e, ao mesmo tempo, despendendo menos elementos lógicos.

Com esse intuito, foi desenvolvido um código, em *verilog*, que cria uma máquina de estados capaz de coletar os dados provenientes do TileCal e, com as informações do filtro em mãos, realizar os cálculos internos que resultarão na estimação da energia das partículas.

Em seguida, serão simuladas as implementações de filtros FIR de três formas: na direta, na transposta e na por máquina de estados aqui projetada. Assim, os resultados obtidos serão analisados e se investigará a viabilidade da implementação aqui proposta no que diz respeito à quantidade de recursos lógicos utilizados.

1.4 ESTRUTURA DO TRABALHO

A presente monografia está organizada da seguinte forma: no Capítulo 1 são apresentados a contextualização, motivação, objetivos e a estrutura do trabalho.

O ambiente deste trabalho é introduzido no Capítulo 2, com destaque para o experimento ATLAS e seus sistemas de calorimetria e de filtragem *online* e, além da descrição do problema de empilhamento de sinais proveniente da atualização do colisor LHC.

O Capítulo 3 foi dedicado à revisão bibliográfica deste trabalho, ou seja, são apresentadas algumas teorias necessárias para a compreensão do mesmo, teorias essas que foram aplicadas e utilizadas nesta monografia.

No Capítulo 4 são apresentadas as teorias de implementação de filtros FIR nas formas direta e transposta. Além disso, é apresentada a técnica de estimação baseada na modelagem inversa para obtenção dos coeficientes do filtro FIR, fazendo o uso do método dos Mínimos Quadrados para minimizar o erro da sua função custo.

A implementação do algoritmo da máquina de estados em *verilog* e sua explicação detalhada estão presentes no Capítulo 5. Todo o funcionamento e o código da estrutura são lá descritos e explicados.

Estão apresentadas no Capítulo 6 as comparações com relação às simulações realizadas para as estruturas de filtros FIR na forma direta e transposta, bem como a pela forma de Máquina de Estados proposta neste trabalho. Também são feitas as comparações entre elas utilizando um banco de dados lá descrito. Ao final do capítulo é apresentada uma discussão acerca dos resultados obtidos.

No Capítulo 7 são discutidos os resultados obtidos e as conclusões gerais do trabalho, e também são apresentadas as considerações finais acerca do mesmo e de seus objetivos previamente propostos. Adicionalmente, são feitos apontamentos com relação a possíveis trabalhos futuros a título de aprimoramentos.

O Apêndice A contém a descrição dos algoritmos implementados em *verilog* para as formas por máquina de estados, direta e transposta de Filtros FIR.

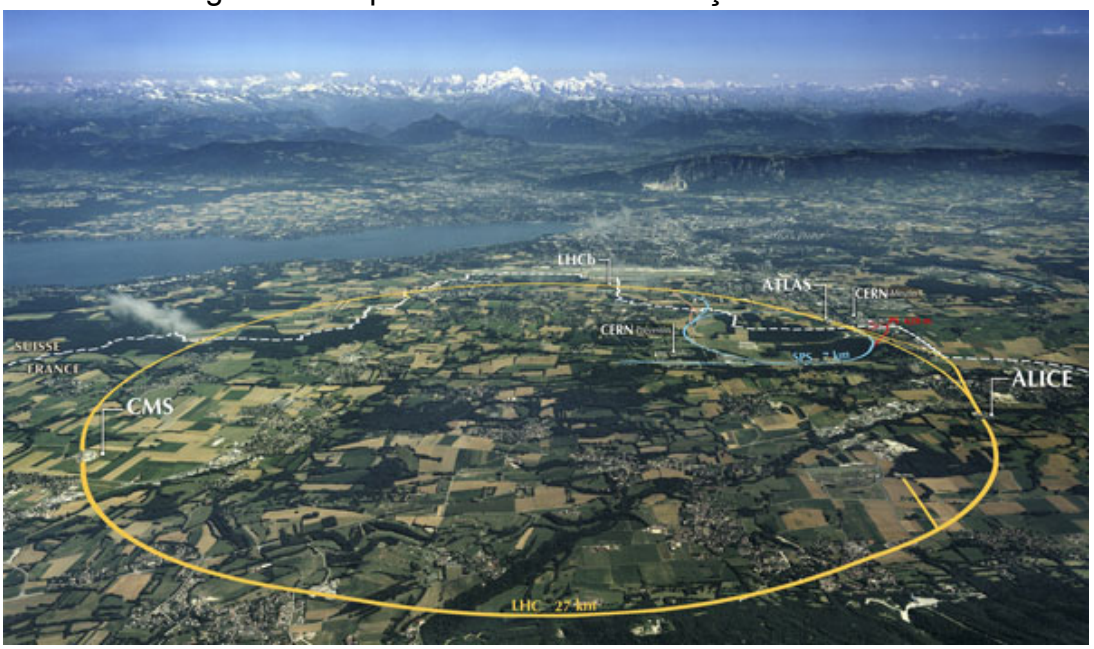
2. O CERN, ACELERADOR LHC E O EXPERIMENTO ATLAS

Neste capítulo é apresentada a contextualização do experimento LHC com foco no ATLAS, que é o local onde o trabalho vem sendo realizado, bem como a ambientalização no CERN de forma geral.

2.1 O CERN

O CERN, ou Organização Europeia para Pesquisa Nuclear, tratava-se, inicialmente, de um órgão mundial que tinha por objetivo estabelecer uma organização de pesquisa sobre física fundamental na Europa. Em 1954, o CERN fundou seu laboratório com sede na fronteira Franco-Suíça próximo a Genebra, sua localização pode ser vista na Figura 1. Ele é formado por 23 estados membros [9].

Figura 1 - Mapa mostrando a localização do CERN.



Fonte: Site da CoEPP (***C**entre of **E**xcellence for **P**article **P**hysics*) [10].

O Brasil se inclui no contexto como país com acordo de colaboração e vem contribuindo de forma direta para o desenvolvimento e pesquisa em áreas de interesse mútuo, sendo que existe a possibilidade, ainda em análise, de o país se tornar membro oficial da instituição.

O CERN tem um efetivo aproximado de 2400 funcionários, além de colaboradores externos os quais são cerca de 11000 pessoas, somando-se cientistas e engenheiros, representando diversas universidades de várias partes do mundo [11].

Dentre os colisores de partículas presentes no CERN, destaca-se o Grande Colisor de Hádrons, ou Large Hadron Collider, ou ainda LHC, que é o foco deste trabalho.

2.2 O LHC

É atualmente o maior e mais poderoso acelerador de partículas do mundo, além de ser aquele de maior energia. Seu principal objetivo é obter dados sobre colisões de feixes de partículas.

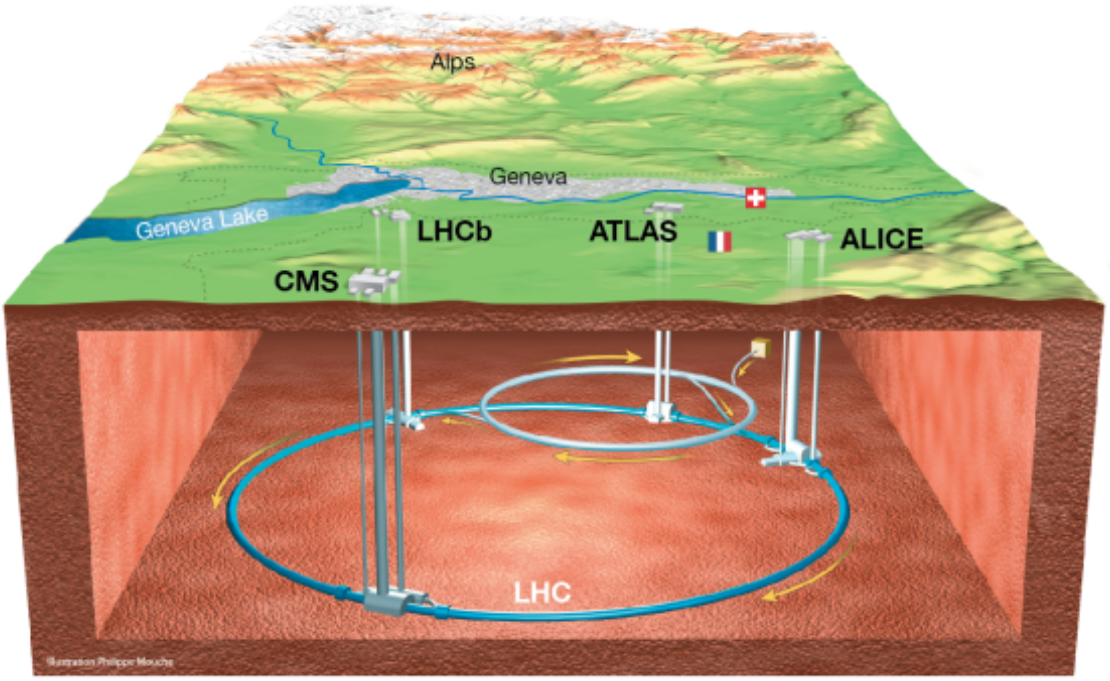
O laboratório encontra-se em um túnel de 27 km de circunferência, localizado a 175 metros abaixo do nível do solo. Ele começou a ser construído em 1998, com a colaboração de mais de 100 países, e está operando desde setembro de 2008.

Em funcionamento, ele acelera dois feixes de prótons em sentidos opostos dentro de tubos mantidos a ultra vácuo [12], os quais atingem, através de eletroímãs, uma velocidade próxima à da luz. O complexo do LHC, mostrado nas Figuras 2 e 3, possui seis detectores em pontos de colisão ao longo de sua circunferência, de modo que as colisões ocorridas possam ser reconstruídas. Tais detectores são denominados: ATLAS (***A*** ***T****oroidal* ***L****HC* ***A****pparatu****S***), CMS (***C****ompact* ***M****uon* ***S****olenoid*), ALICE (***A*** ***L****arge* ***I****on* ***C****ollider* ***E****xperiment*), LHCb (***L****arge* ***H****adron* ***C****ollider* ***b****eauty*), TOTEM (***TOT****al* ***E****lastic and diffractive cross section* ***M****easurement*) e o LHCf (***L****arge* ***H****adron* ***C****ollider* ***f****orward*).

Os dois maiores experimentos de propósito geral do LHC são o ATLAS e o CMS, sendo esses dois detectores construídos para estudar a colisão entre prótons e entre íons pesados.

O ALICE é um detector dedicado à física de íons pesados, ou seja, é destinado ao estudo da física de matéria com fortes interações e de extrema densidade de energia, que é onde uma fase da matéria se forma: o plasma de *quarks* e *glúons* [14].

Figura 2 - Visão Geral do LHC.



Fonte: Foto de Philippe Mouche retirada do Servidor de Documentos do CERN [13].

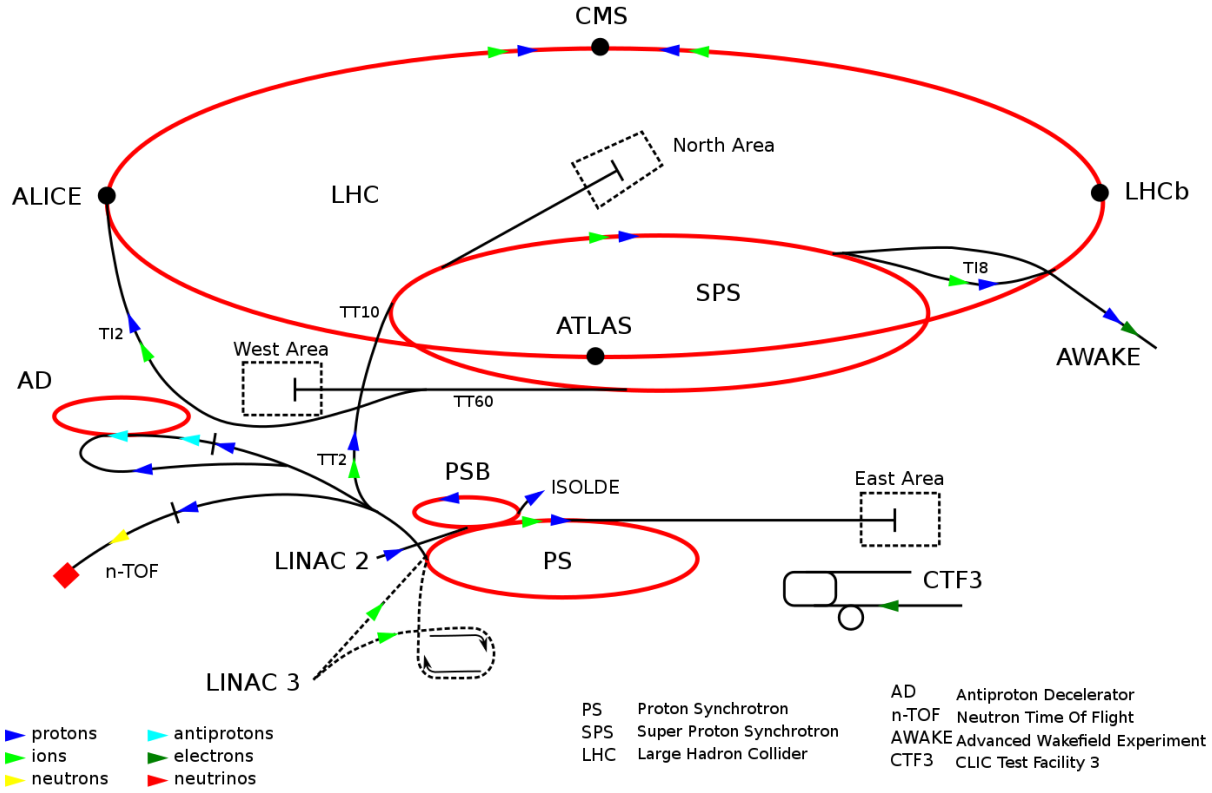
O CMS é um experimento de propósito geral para o estudo do modelo atômico elementar, incluindo o *Bóson de Higgs*, em busca de dimensões extras e partículas que podem originar matéria escura [15].

O LHCb é um experimento especializado em investigar as diferenças minuciosas entre matéria e antimatéria através do estudo de uma partícula chamada *beauty quark* ou *b quark*.

O ATLAS é, também,um experimento de propósito geral do LHC. Ele investiga uma variada gama de fenômenos físicos provenientes de uma colisão proton-proton. Apesar de ter os mesmo objetivos científicos do experimento CMS já citado acima, esse usa diferentes soluções técnicas e diferente design de sistema magnético [16].

Os experimentos TOTEM e LHCf são de pequeno porte e se dedicam à física projetiva (*forward*) de prótons e íons pesados.

Figura 3 - O complexo do LHC com todos os seus detectores e subdetectores.



Fonte: Foto de Maximilien Brice retirada do Servidor de Documentos do CERN [17].

O ATLAS é o ambiente no qual este trabalho foi desenvolvido e será melhor explicado e detalhado a seguir.

2.3 O EXPERIMENTO ATLAS

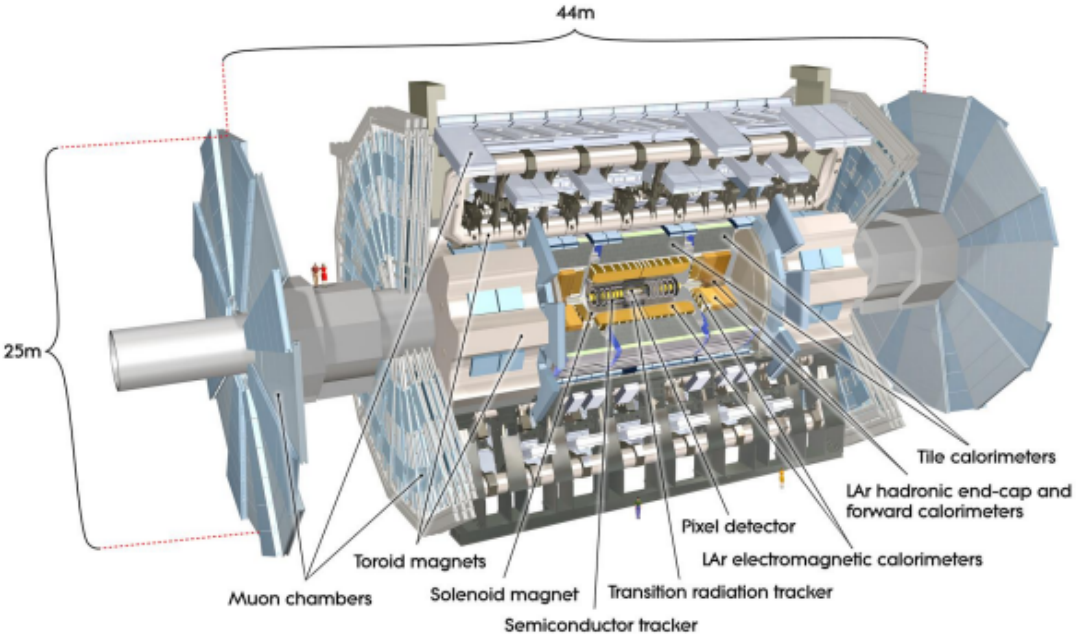
Trata-se de um dos principais experimentos do LHC, onde trabalham mais de 3000 cientistas de 174 instituições espalhadas por 38 países. Ele tem por finalidade investigar diversos fenômenos físicos, fazendo a medição e detecção de resultados vindos das colisões *próton-próton* no LHC [18].

O ATLAS, ilustrado na Figura 4, pesa cerca de 7.000 toneladas, possui 44 metros de comprimento, 25 metros de altura e 25 metros de largura e é o maior detector de partículas já construído. Ele possui formato cilíndrico e foi projetado para cobrir um ângulo sólido próximo a 4π, ao redor da região de colisão das partículas. Além de magnetos responsáveis pela geração de intensos campos magnéticos que auxiliam na medida do momento das partículas carregadas, o ATLAS também possui

3 subdetectores: o detector de trajetórias, o detector de *múons* e os calorímetros eletromagnético e hadrônico, conforme é mostrado na Figura 4.

O detector de trajetórias ID (***I**nner **D**etector*), que se encontra na camada mais interna do ATLAS, consiste de três diferentes sistemas de sensores: o PD (***P**ixel **D**etector*), o ST (***S**emiconductor **T**racker*) e o TRT (***T**ransition **R**adiation **T**racker*), imersos em um campo magnético paralelo ao eixo de feixes. Ele mede a direção, o momento, e a carga das partículas produzidas em cada colisão *próton-próton*.

Figura 4 - O detector ATLAS e seus subsistemas.



Fonte: Foto de João Pequenão retirada do Servidor de Documentos do CERN [19].

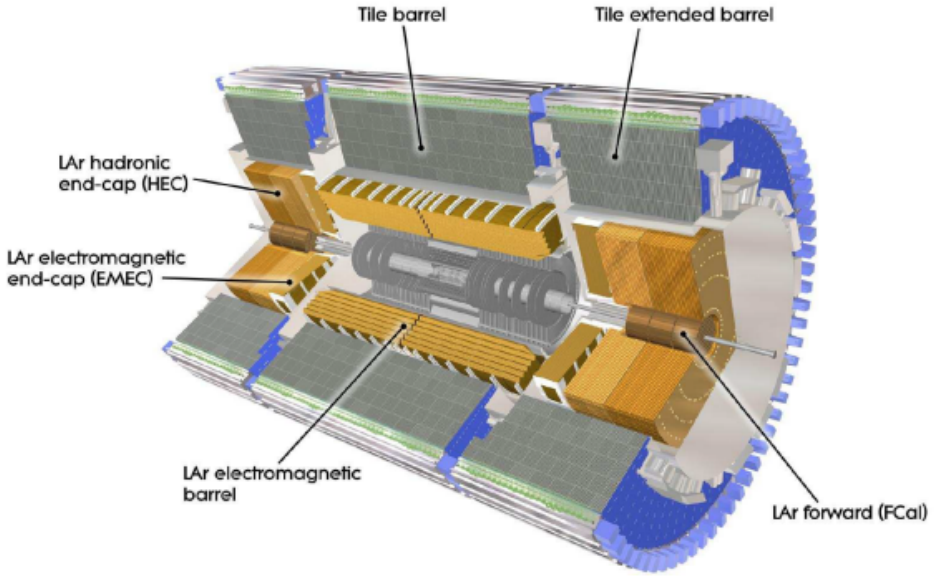
O Espectrômetro de Múons, do inglês *Muon Spectrometer*, é um subdetector desenvolvido para identificar, filtrar e medir o momento dos múons, partícula capaz de atravessar todo o material do detector, sendo as únicas que alcançam distâncias tão grandes, além do ponto de colisão [20].

No sistema de calorimetria do ATLAS, ilustrado na Figura 5, temos diversos calorímetros especializados: o LAr (***L**iquid **Ar**gon Calorimeter*) [31], o TileCal (***Tile Cal**orimeter*), o EMEC (*LAr **E**lectro**m**agnetic **E**nd-**C**ap*), o HEC (*LAr **H**adronic **E**nd-**C**ap*) [32] e o FCal (*LAr **F**orward **Cal**orimeter*). Dentre eles destacamos o LAr, projetado para medir a energia de partículas que interagem de forma eletromagnética com seu material constituinte, ou seja, elétrons e fótons, e o

TileCal, também conhecido como Calorímetro Hadrônico ou ainda Calorímetro de Telhas [21]. Este último foi projetado para medir a energia de partículas que interagem de forma hadrônica com seu material [22].

Na prática, os calorímetros têm a função de absorver, amostrar e medir a energia das partículas em que neles incidem. Estas partículas, ao entrarem em contato com o material dos calorímetros, geram um chuveiro de partículas, onde parte de sua energia é depositada, coletada e medida, o que é possível devido aos calorímetros serem compostos por um material denso, formando barreiras que permitem que as partículas sejam absorvidas por completo. Os *múons*, de alta energia, não são absorvidos pelo experimento, depositando apenas uma pequena parte de sua energia nos calorímetros. Quando o processo de chuveiro é iniciado, as partículas sofrem decaimentos, produzindo partículas de menor energia, e este processo se dá até a absorção total da energia da partícula pelo calorímetro [24].

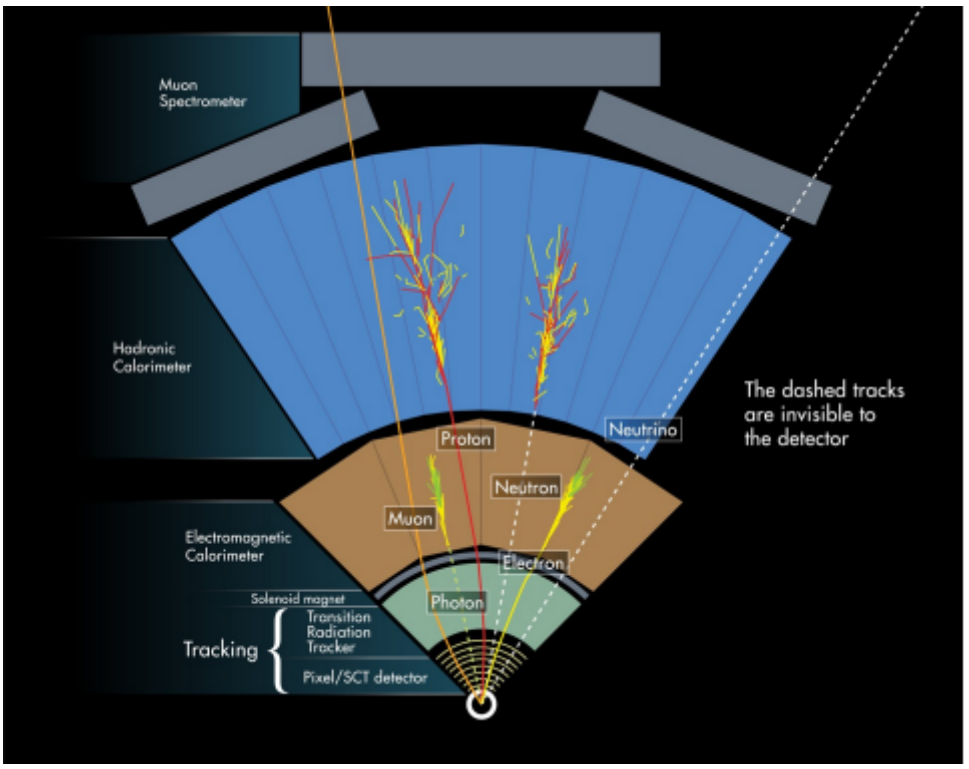
Figura 5 - Sistema de calorimetria do ATLAS.



Fonte: Foto de João Pequenão retirada do Servidor de Documentos do CERN [23].

Pode-se observar na Figura 6, um esquema de como diversas partículas interagem com os subdetectores do ATLAS após uma colisão, sendo possível mensurá-las e caracterizá-las à medida que as mesmas atravessam as camadas de detecção.

Figura 6 - Interação das partículas com os subdetectores do ATLAS.



Fonte: Foto de João Pequenão e Paul Schaffner retirada do Servidor de Documentos do CERN [25].

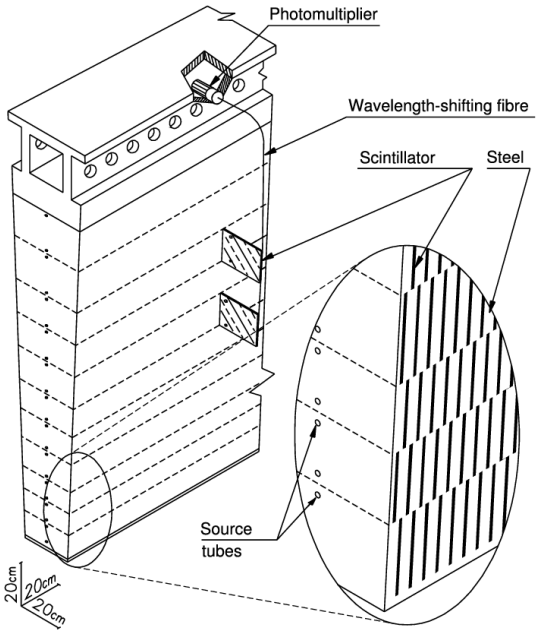
2.3.1 O Calorímetro Hadrônico

O TileCal foi projetado para absorver maior parte das partículas provenientes de uma colisão, forçando-as a depositarem toda sua energia dentro do detector. Ele mede a energia de *hadrons*, partículas constituídas de *quarks* e *gluons*, através de placas cintiladoras, ou telhas cintiladoras, que funcionam como material amostrador de energia, e placas de aço ativo como material absorvedor.

As telhas cintiladoras são excitadas quando partículas carregadas as atravessam, ocorrendo a produção de *fótons,* havendo assim a conversão de luz em sinal elétrico através de células fotomultiplicadoras, ou PMT (***P**hoto**M**ultiplier **T**ube*). Isso permite, também, a percepção de partículas que não interagem com o material do TileCal e passam direto, mas têm suas presenças notadas por ele, de forma indireta. Após a conversão de luz em sinal elétrico, o segundo é coletado por fibras óticas, as quais estão nas duas extremidades da telha, conferindo redundância e, com isso, maior confiabilidade nas leituras.

A Figura 7 ilustra um módulo do TileCal tridimensionalmente, onde pode ser observado a disposição perpendicular das telhas em relação ao feixe de partículas.

Figura 7 - Esquema de um módulo do TileCal.

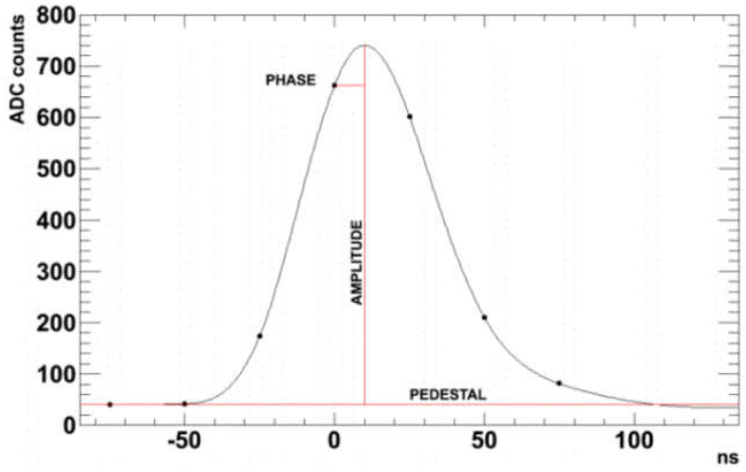


Fonte: Foto retirada do artigo The ATLAS Experiment at the CERN Large Hadron Collider [26].

Este sinal é enviado a um circuito de amplificação e condicionamento, resultando em um pulso estável de forma fixa, o qual possui uma energia proporcional àquela depositada nas células [24]. Cada célula é lida por duas PMTs e cada PMT corresponde a um canal de leitura resultando, no total, em cerca de 10.000 canais de leitura que, juntos, geram 60 TB/s de informação. Este pulso característico, ilustrado na Figura 8, é composto por 7 amostras espaçadas de 25 ns, digitalizado a uma taxa de 40 MHz, sincronizada com a taxa de colisão [27].

Resumindo, o objetivo principal do TileCal é contribuir na reconstrução da energia dos feixes de partículas provenientes das interações *proton-proton* e auxiliar nos cálculos de momento transverso. Sua eletrônica inclui circuitos de front-end e digitalizadores de sinais, os quais são projetados de acordo com as características de alta velocidade e baixo ruído de suas PMTs. Assim, os sinais digitalizados correspondentes aos eventos no calorímetros são transferidos para buffers por meio das fibras óticas [24].

Figura 8 - Formato de um pulso característico do TileCal.



Fonte: Foto retirada do artigo *The TileCal Energy Reconstruction for Collision Data Using the Matched Filter* [27].

2.3.2 O Sistema de *Trigger* do ATLAS

O armazenamento dos dados e os recursos para análises offline não seriam capazes de lidar com a quantidade de dados gerados na frequência de operação nominal do LHC. Contudo, apenas uma fração desta informação é considerada de interesse para a descrição da física desejada. Assim, a fim de registrar quaisquer informações relevantes e descartar as demais, o ATLAS possui um sistema de filtragem *online*, sistema de *trigger*, o qual reduz a taxa de informações a fim de focarmos apenas no que interessa para a reconstrução da informação [28].

As interações nos subdetectores do ATLAS geram um enorme fluxo de dados. Para processar os mesmos, o ATLAS utiliza um sistema avançado de seleção de eventos, com o objetivo principal de selecionar quais eventos podem ser armazenados e quais devem ser ignorados [18].

Geralmente, em ambientes que operam com taxas altas de eventos, existem restrições temporais e, eventualmente, os eventos gerados podem exigir uma elevada quantidade de memoria, necessitando, assim, de uma discriminação online de alta velocidade, tornando o processo ainda mais complexo [30].

Esse sistema é muito importante visto que, a cada *bunch* de *prótons* se cruzando, o chamado BC (***B**unch **C**rossing*) está, inicialmente, a uma taxa de 40 MHz no momento em que colidem. No entanto, mesmo que tenhamos

100 bilhões de prótons por *bunch* no ponto de interação, só ocorre cerca de 20 colisões por cruzamento e acaba-se tendo 800 milhões de colisões por segundo, isso ocorre pois a maioria dos prótons se erram no momento da colisão e continuam seu caminho ao redor do anel. A taxa de eventos selecionados deve ser reduzida para 200 Hz, visando o armazenamento para análise posteriores dos dados [33].

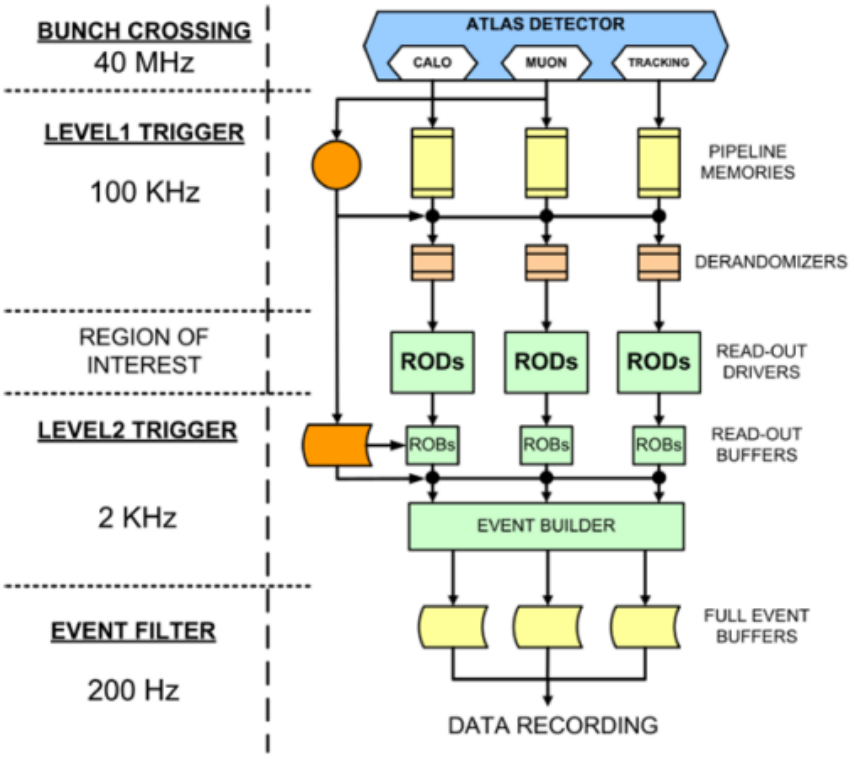
O sistema de *trigger* é composto por três níveis conectados em cascata, os quais operam *online* e, em cada um deles, o critério de seleção é refinado [34]. Eles podem ser visualizados na Figura 9 e serão citados em ordem de complexidade e tempo de processamento:

- O primeiro nível de *trigger*, também chamado de L1 (*level* 1), é implementado em *hardware* e se subdivide em *trigger* de calorimetria e trigger de muons. Ele foi projetado para operar numa frequência de 40 MHz. Os sinais provenientes da leitura de cada subdetector chegam ao processador central de filtragem, ou CTP (***C**entral **T**rigger **P**rocessor*) onde se identifica as assinaturas básicas da física de interesse. Trabalha-se olhando uma segmentação menor das células do calorímetro hadrônico, o que é feito juntando-se informações de células vizinhas em um único elemento, reduzindo a quantidade de informação a ser analisada. Assim é reduzida a taxa de eventos de entrada para 100 KHz. É neste nível que está inserido o presente trabalho [35];

- O segundo nível de filtragem, chamado de L2 (*level* 2), é implementado em *software* e recebe as informações oriundas das Regiões de Interesses, ou RoI (***R**egion **o**f **I**nterest*), definidas pelo L1 e geradas pelo CTP com uma latência de 10 μs, refinando a filtragem realizada pelo mesmo e reduzindo ainda mais o número de eventos selecionados, de 100 KHz para 2 KHz. Ele faz isso através de uma rede de computadores que processa os algoritmos de busca especializados nos diversos subdetectores do ATLAS, visando encontrar elementos que representam os possíveis decaimentos referentes à física de interesse [36]; e

- O terceiro nível de filtragem, chamado de Filtro de Eventos, ou EF (***E**vent **F**ilter*), combina toda a informação selecionada pelo L2 e reduz a taxa de 1 KHz para 200 Hz e, neste nível, os dados estão em memórias temporárias e o EF indica os dados que, de fato, serão armazenados, permitindo, assim, posteriores análises que podem ser executadas de forma offline [36].

Figura 9 - Níveis de filtragem do sistema de trigger do ATLAS.



Fonte: *Implementation of the ROD Crate DAQ Software for the ATLAS Tile Calorimeter and a Search for a MSSM Higgs Boson decaying into Tau pairs* [37].

Os segundo e terceiro níveis de filtragem são denominados Trigger de Alto Nível, ou HLT (***H**igh **L**evel **T**rigger*) [33]. Quando o processo de filtragem chega ao fim, a taxa de eventos diminui para uma frequência de 200 Hz.

2.4 A ATUALIZAÇÃO DO ATLAS

De tempos em tempos, o LHC passa por um processo chamado de Programa de Atualização, sua cronologia e o planejamento futuro da mesma podem ser vistos

na Figura 10, visando a sua preparação para um aumento da densidade dos feixes de prótons, fenômeno esse que, no âmbito da física de altas energias, é denominado de aumento de luminosidade [38, 39]. Isso significa que diversos componentes de seus sistemas serão substituídos. Contudo, uma das principais mudanças será no sistema de interface com o primeiro nível de *trigger*, tendo por finalidade, lidar com os novos requisitos de alta luminosidade.

Figura 10 - Linha do tempo do programa de atualizações do LHC.

Fonte: *Introduction to the HL-LHC Project* do site *High Luminosity LHC Project*[40].

A luminosidade corresponde à medida do número de colisões que podem ser produzidas, em um detector, por cm² e por segundo, podendo ser calculada através da Equação 2.1 [41].

$$Luminosidade \approx \frac{P^2}{t.A} \tag{2.1}$$

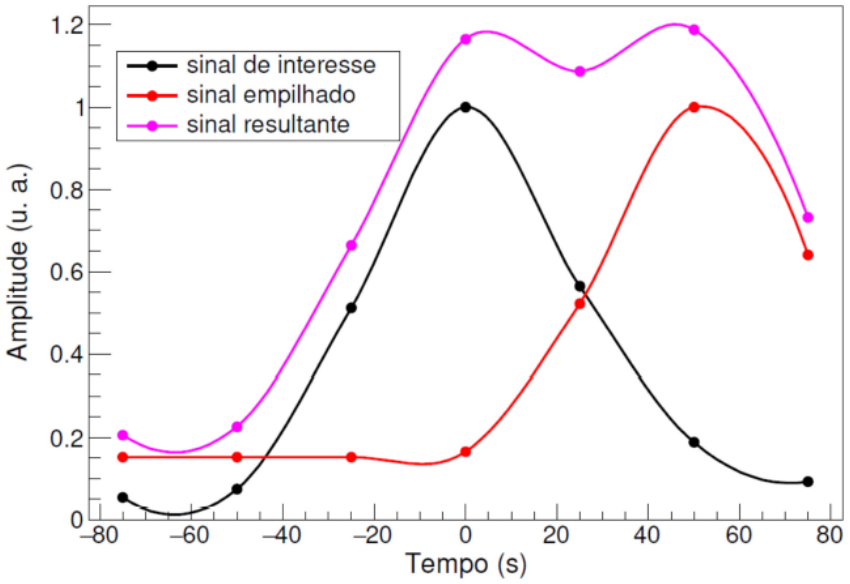
Nesta equação tem-se que P^2 corresponde ao número de prótons ao quadrado, t é o tempo entre as colisões e A a área da seção transversal do feixe [41].

Como o tempo de resposta da eletrônica de leitura nos calorímetros é maior que o período entre as colisões, o aumento na taxa de eventos intervirá, principalmente, na sobreposição dos sinais provenientes de eventos subsequentes,

resultando em um sinal recebido deformado, o que dificulta a equalização do canal. A esse efeito é dado o nome de *pile-up* [4, 7, 42].

A Figura 11 ilustra esse efeito, onde o sinal resultante é modificado, pois a célula do TileCal foi sensibilizada novamente enquanto o sinal de interesse ainda estava sendo processado, ou seja, houve uma segunda colisão num período de tempo pequeno demais para que o sistema entendesse se tratar de sinais diferentes. Isso gera um sinal resultante que é o efeito do empilhamento dos sinais [4, 7].

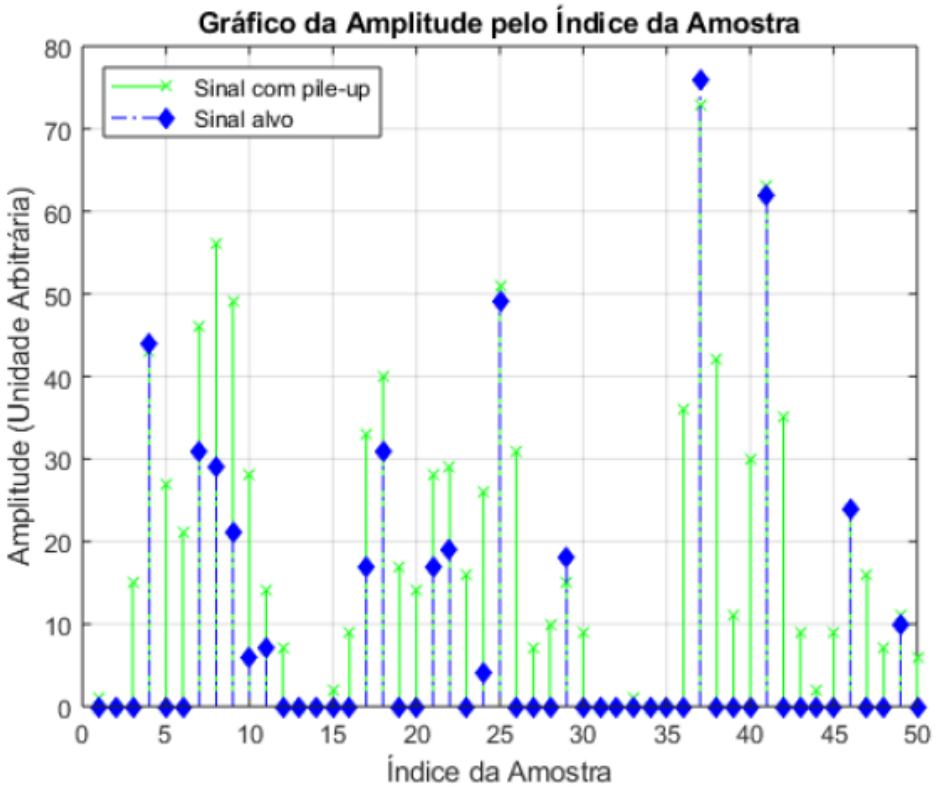
Figura 11 - Efeito *pile-up* no TileCal.



Fonte: Foto retirada do artigo *Calorimeter Response Deconvolution for Energy Estimation in High-Luminosity Conditions* publicado na revista *IEEE Transactions on Nuclear Science* [4].

A Figura 12 apresenta um gráfico contendo as características de empilhamento realizadas no *software* MATLAB [43]. A amplitude do sinal em azul representa a energia ideal que desejamos mensurar no canal de leitura do TileCal, sem os ruídos inclusos. Tal sinal de energia acaba sendo deformado devido ao efeito de pile-up, representado na cor verde [30].

Figura 12 - Comparação entre um sinal ideal e um com efeito *pile-up*.



Fonte: Foto retirada do trabalho Processamento Embarcado para Implementação de um Método Iterativo de Deconvolução Visando a Reconstrução de Energia em Calorímetros de Altas Energias [30].

A sobreposição de sinais é um problema recorrente quando os dados são enviados através de um canal de informações a uma taxa superior ao tempo de resposta deste canal, dificultando a seleção de eventos de interesse, feita pelo sistema de *trigger* [34].

3. REVISÃO BIBLIOGRÁFICA

A seleção de eventos realizada no primeiro nível de *trigger*, até então, podia ser resumida em uma sequência de dois algoritmos: um Filtro Casado ou MF (***M**atched **F**ilte*r), o qual auxilia atenuando ruídos provenientes da eletrônica do calorímetro, e um circuito detector de pico que extrai a informação de onde ocorreu a deposição de energia [44].

Em ambientes de alta luminosidade, pelo fato do efeito *pile-up* se intensificar e a forma do pulso característico ser modificada, o desempenho do Filtro Casado fica comprometido, uma vez que este depende do conhecimento prévio da forma do pulso característico, resposta impulsiva do calorímetro [7]. Sendo assim, novas técnicas de estimação foram propostas e testadas a fim de minimizar o efeito que o empilhamento de sinais produz na estimação de energia no L1.

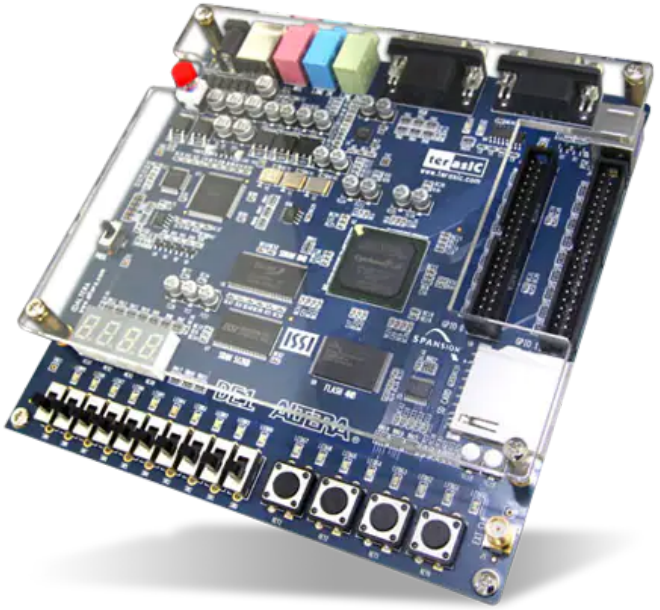
Os filtros FIR possuem simples implementação em FPGA e são largamente indicados para a reconstrução da energia das colisões do LHC e de outros experimentos de e calorimetria. Neste trabalho, o dispositivo utilizado para implementação dos métodos citados é o FPGA, sigla que significa *Field Programmable Gate Arrays*, e é um circuito integrado pertencente a uma classe de dispositivos chamada de FPL (***F**ield-**P**rogrammable **L**ogic*) ou, em português, Dispositivo Lógico Programável [45].

3.1 O FPGA

Trata-se de um dispositivo composto de blocos repetidos de elementos lógicos programáveis de forma a produzir um *hardware* digital dinamicamente configurável [46]. Resumidamente, são chips de silício que utilizam blocos lógicos programáveis pré-construídos e têm recursos de roteamento e a possibilidade de serem configurados para se implementar funcionalidades personalizadas de eletrônica digital. Na Figura 13 tem-se um exemplo de um sistema desses, sem a necessidade de se entrar em contato com uma placa de montagem, apenas carregando, em seu *software*, um arquivo contendo as informações sobre como os componentes devem ser ligados entre si.

Este arquivo é carregado em uma memória de configuração interna do FPGA e pode ser alterado durante o tempo de execução [44].

Figura 13 - Exemplo de kit de desenvolvimento de lógica programável FPGA.



Fonte: Foto retirada do site da empresa Terasic Technologies [56].

Pode-se destacar algumas vantagens em relação ao uso dos FPGAs no que tange o projeto de funcionalidades da eletrônica digital, dentre elas:

- Habilidade de se poder reconfigurar ou customizar a lógica digital, ou seja, sua estrutura interna, constituída de uma arquitetura altamente granularizada, permite uma grande flexibilidade, abrangendo todo e qualquer tipo de circuito digital que se deseja implementar;

- Alta densidade de lógica, ou seja, alta granularidade, existindo chips de FPGA contendo milhares e até milhões de elementos lógicos num mesmo encapsulamento [47, 48]. Isso faz com que o esse dispositivo admita implementações mais elaboradas e complexas;

- Alta velocidade de operação, suportando a síntese de circuitos síncronos que podem atingir frequências da ordem de *Gigahertz* [49];

- Capacidade de implementação de circuitos digitais independentes sendo utilizados simultaneamente, com diferentes taxas de clock, se desejado; e

- Possibilidade de se utilizar linguagens de alto nível de abstração para se desenvolver projetos de descrição de *hardware*. Dentre essas linguagens destacam-se o VHDL e o *Verilog*, sendo esta última a linguagem utilizada para o projeto deste trabalho.

Porém, existem também desvantagens que merecem ser destacadas quando comparamos os FPGAs a outros processadores como os DSPs (***D**igital **S**ignal **P**rocessor*), por exemplo, os quais são microprocessadores especializados em processamento digital de sinais. São elas, dentre outras:

- Alto custo pois, apesar de os chips e kits de desenvolvimento estarem sendo fabricados e disponibilizados com um preço cada vez mais acessível, eles ainda são considerados caros para serem adquiridos; e

- Complexidade de implementação, já que se gastaria menos tempo se implementando um algoritmo em um processador se comparado a fazer o mesmo em um FPGA, isso ocorre devido à possibilidade de projeto em linguagem de *software* e de realização de processo de *debug*.

A programação empregada no FPGA é feita através de *Verilog*, que é uma linguagem de descrição de *hardware* usada para modelar sistemas eletrônicos ao nível de circuito, permitindo a implementação de processos analógicos, digitais e híbridos em vários níveis de abstração. Essa possui uma sintaxe bem parecida com a da linguagem C e possui uma estruturação baseada em blocos, se caracterizando em uma linguagem modular.

Técnicas de deconvolução para estimação de energia *online* e *offline*, baseadas em filtros FIR, vêm sendo propostas com o intuito específico de reduzir o efeito de empilhamento de sinais [4, 6]. Tais técnicas possuem uma implementação simples em FPGA, o que torna a utilização desse elemento bastante viável.

Além disso, trabalhos anteriores demonstram o potencial de um processador dedicado para implementação de técnicas iterativas de estimação de energia em FPGA [30] e [58]. Sendo que este permite a viabilidade na utilização, em *hardware*, de algoritmos complexos e é detalhado a seguir.

3.2 O PROCESSADOR SAPHO

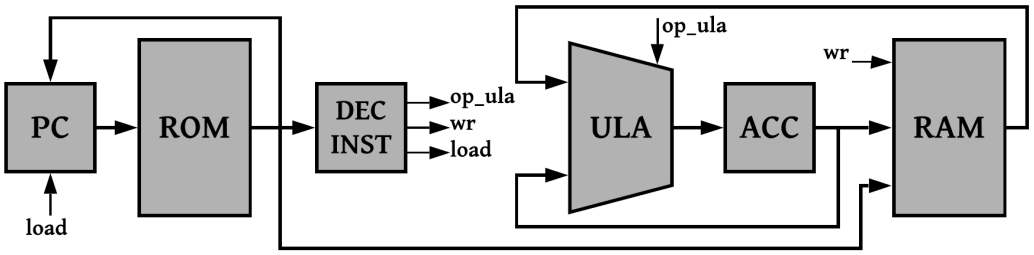
Recentemente, vem-se estudando a implementação de um processador dedicado, em FPGA, com uma arquitetura multi-core, o qual é capaz de executar algoritmos iterativos, destaca-se os baseados em Representação Esparsa de Dados, visando, também, a reconstrução *online* de energia em Calorímetros de Altas Energias, em cenários de empilhamento de sinais [30].

Para a implementação do método iterativo de deconvolução utiliza-se o Processador *Soft-Core* SAPHO (***S**calable **A**rchitecture **P**rocessor for **H**ardware **O**ptimization*), mais especificamente um processador soft-core parametrizável e de código aberto, o qual é capaz de realizar operações aritméticas em ponto fixo e ponto flutuante [51].

Ao contrário dos processadores *soft-core* existentes, o SAPHO, que tem o diagrama de sua estrutura apresentado na Figura 14, não possui uma arquitetura fixa, ou seja, os recursos são automaticamente alocados, fazendo com que apenas os elementos lógicos necessários venham a ser utilizados, tornando o projeto, como um todo, menos dispendioso. Ele realiza uma conversão paralelo-serial de uma janela de amostras e, através de métodos iterativos, essas amostras são disponibilizadas numa velocidade maior que o clock de requisição, de forma free-running e sequencial.

Este processador foi desenvolvido no NIPS (**N**úcleo de **I**nstrumentação e **P**rocessamento de **S**inais) da UFJF (**U**niversidade **F**ederal de **J**uiz de **F**ora) e já vem sendo usado como parte integrante de diversos sistemas já consolidados, nos quais pode-se citar [52], [53], [54] e [55]. Nestes trabalhos encontram-se detalhes mais aprofundados a respeito da estrutura e funcionamento do SAPHO.

Figura 14 - Diagrama do Processador SAPHO.



Fonte: *A gapless waveform recorder for monitoring Smart Grids* apresentado na conferência *17th International Conference on Harmonics and Quality of Pow*er (ICHQP) [29]**.**

3.3 MÁQUINAS DE ESTADOS

Máquinas de estados são circuitos que sequenciam um conjunto de estados predeterminados controlados por um *clock* e outros sinais de entrada. Baseada em seu estado atual e dada uma entrada, a máquina de estados realiza a transição entre estados e produz saídas [64].

A construção básica dos blocos de uma máquina de estados é baseada em estados e transições. Um estado é uma característica do sistema que depende de uma entrada anterior e causa uma reação nas etapas seguintes do mesmo. Define-se um estado como o inicial da máquina, onde a execução da mesma se inicia. À medida que a máquina está em funcionamento seus estados irão mudar de um para outro, seguindo uma sequência e lógica predeterminada, gerando resultados que serão carregados, ou não, de um estado a outro. Deve-se notar que, ao se dizer entradas e saídas, está se referindo à ações ou símbolos, onde uma entrada pode ser o apertar de um botão, ou um sinal de controle que irá habilitar, ou não, um determinado estado da máquina, dependendo de seu valor lógico [64].

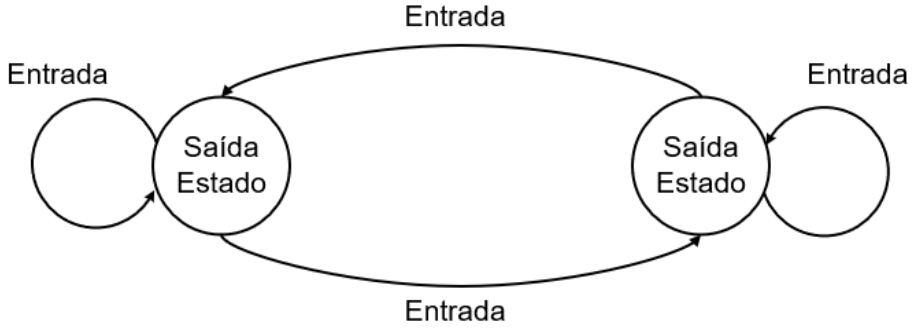
Destacam-se dois tipos de variações de máquinas de estados: o modelo de Mealy e o de Moore.

3.3.1 O Modelo de Moore

Os valores de saída são determinados, unicamente, pelo estado atual, ou seja, são incondicionais e não dependem diretamente dos sinais de entrada [64].

A Figura 15 exemplifica o modelo genérico e simples para esse tipo de máquina de estados, podendo haver mais estados dependendo da aplicação.

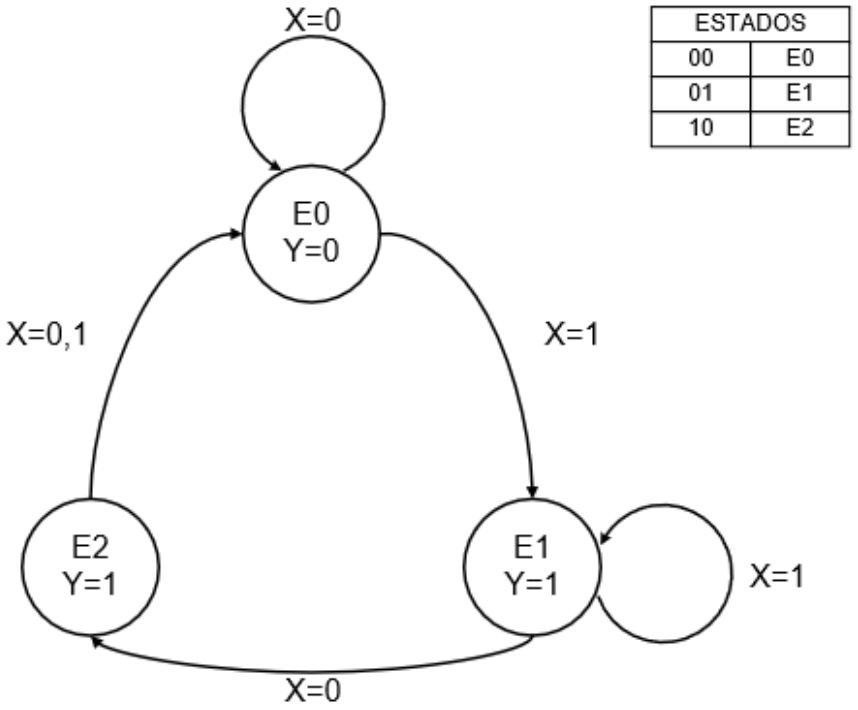
Figura 15 - Diagrama de uma Máquina de Estados de Moore.



Fonte: Do Autor.

Para exemplificar considera-se o diagrama da Figura 16 e a tabela de estados presente na mesma.

Figura 16 - Diagrama de exemplo de uma Máquina de Estados de Moore.



ESTADOS	
00	E0
01	E1
10	E2

Fonte: Do Autor

Como pode-se perceber pelo diagrama constante da Figura 16, quando o estado atual possui 00 sua saída será 0, se for 01 será 0, se for 10 será 1 e no caso de ser 11 não está definida saída nem haverá um próximo estado, sendo descrito, assim, por '–' na Tabela 1. Com isso pode-se escrever todas as saídas na tabela de acordo com o estado atual da máquina.

Em seguida se escreve o próximo estado da máquina de acordo com o estado atual. Se o estado atual for E_0 e X=0, se permanece no estado atual E_0. Se o estado é E_0 e X=1 o mesmo muda para E_1. No estado E_1 se X=1 se permanece em E_1, mas se X=0 o mesmo muda para E_2. Em E_2 se X=0 ou X=1 o estado irá para E_0, em ambos os casos. Assim a tabela ficará preenchida conforme consta da Tabela 1.

Tabela 1 - Tabela verdade do exemplo da Máquina de Moore.

ESTADO ATUAL(AB)		**ENTRADA(X)**	**PRÓXIMO ESTADO(A^+B^+)**		**SAÍDA(Y)**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	–	–	–
1	1	1	–	–	–

Fonte: Do Autor.

Com isso tem-se uma ideia do funcionamento de uma máquina de Moore. Através da Tabela 1 pode-se montar o circuito da máquina de estados do exemplo para os próximos estados, Equações 3.1 e 3.2, e para a saída, Equação 3.3, os quais ficam da seguinte forma:

$$A^+ = \overline{A}B\overline{X} \tag{3.1}$$

$$B^+ = \overline{A}X \tag{3.2}$$

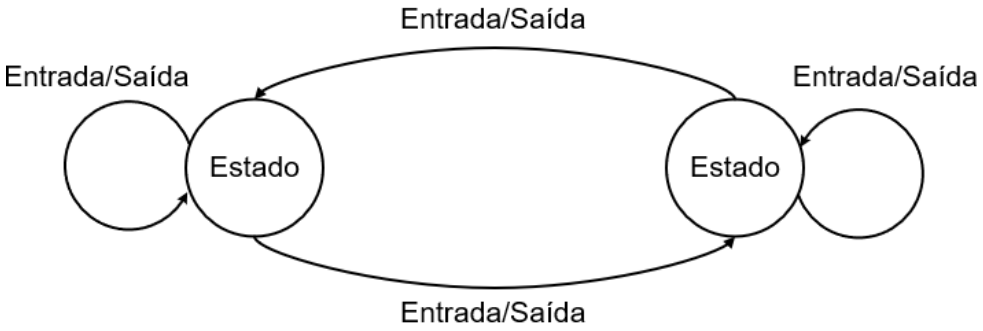
$$Y = A\overline{B} \tag{3.3}$$

Logo, percebe-se que a saída Y não depende dos valores de entrada X mas, apenas, dos valores dos estados atuais A e B.

3.3.2 O Modelo de Mealy

No modelo de Mealy, os sinais de saída são controlados por sinais de entrada adicionais, e a mudança de estado depende não somente dos anteriores, mas também do estado atual. Pode-se ver na Figura 17 o funcionamento desse modelo e suas peculiaridades, visto que a mudança de estados depende, não somente da entrada, mas também da saída. Com isso, esta máquina é mais difícil de modelar e requer mais *hardware* para ser implementada, em contrapartida apresenta resposta mais rápida e requer menor número de estados, se comparado ao modelo de Moore [64].

Figura 17 - Diagrama de uma Máquina de Estados de Mealy.



Fonte: Do Autor.

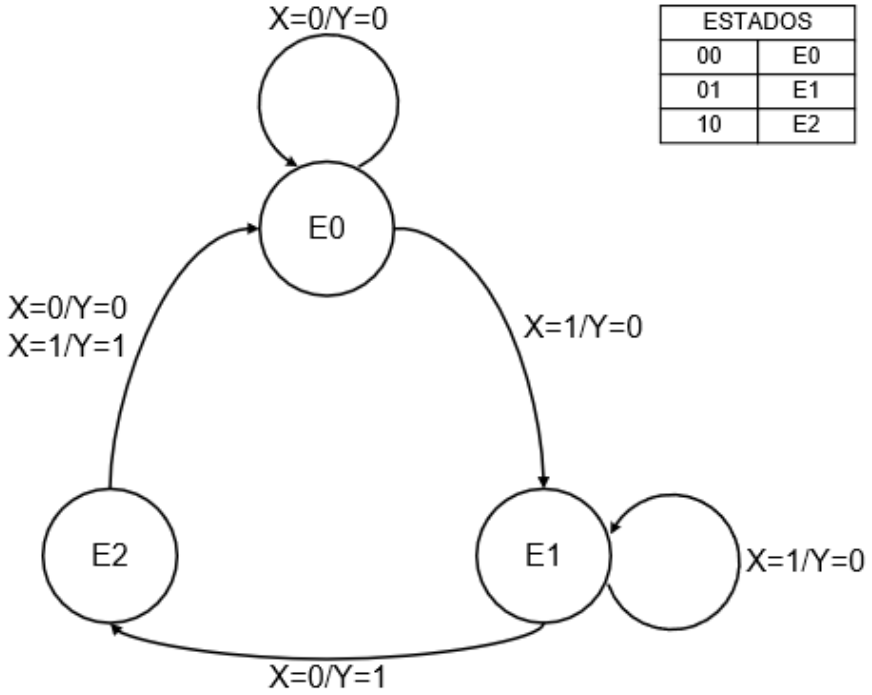
Para exemplificar considera-se o diagrama da Figura 18 e a tabela de estados presente na mesma.

Este caso funciona da seguinte forma: quando se está no estado E_0 e X=0 permanece no mesmo estado (00) a saída será Y=0. Se X=1 em E_0a saída será Y=0 e o estado mudará para E_1. Em E_1 se X=1, a saída será y=0 e o estado permanecerá em E_1. Se X=0 a saída será Y=1 e o estado irá para E_2. Em E_2 se X=0 a saída será Y=0 e o estado voltará para E_0. Se X=1 a saída será Y=1 e o estado também mudará para E_0 novamente.

Para os estados não definidos na máquina de Mealy, escreve-se *don't care* ('x'), como pode ser visto na Tabela 2, ao contrário da máquina de Moore que utiliza o sinal de não definido ('–').

Assim, escreve-se a Tabela 2 com os valores dos próximos estados e as saídas para o exemplo.

Figura 18 - Diagrama de exemplo de uma Máquina de Estados de Mealy.



Fonte: Do Autor.

Tabela 2 - Tabela verdade do exemplo da Máquina de Mealy.

ESTADO ATUAL(AB)		**ENTRADA(X)**	**PRÓXIMO ESTADO(A^+B^+)**		**SAÍDA(Y)**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	x	x	x
1	1	1	x	x	x

Fonte: Do Autor.

Com a tabela em mãos pode-se escrever os circuitos correspondentes ao caso acima para os próximos estados, Equações 3.4 e 3.5, e para a saída, Equação 3.6, ficando da seguinte forma:

$$A^{+}=B\overline{X} \tag{3.4}$$

$$B^{+}=X\overline{A} \tag{3.5}$$

$$\mathrm{Y} = B\overline{X} + XA \tag{3.6}$$

Pode-se notar que na máquina de Mealy a saída Y depende dos valores de entrada X ou de outros sinais adicionais, se necessário, e não unicamente do estado atual, como é o caso da máquina de Moore. Assim, a máquina de estados proposta neste trabalho é uma máquina de Mealy.

Com o embasamento teórico exposto até então, pretende-se, a partir deste ponto, implementar, de forma alternativa, o filtro FIR através de uma máquina de estados, visando a otimização da lógica de *hardware*. Tal proposta será explicada nos próximos capítulos, bem como o projeto e o raciocínio envolvidos.

Primeiramente, é preciso introduzir a teoria de filtros FIR e IIR, o que será descrito em seguida.

3.4 FILTROS FIR E IIR

Classifica-se uma função de transferência digital baseada em seu comprimento da sequência de sua resposta ao impulso por respostas ao impulso finitas FIR e respostas ao impulso infinitas IIR [50].

Para se entender melhor ambas as classificações precisa-se, primeiramente, entender um sistema SISO (***S**ingle-**I**nput **S**ingle-**O**utput*). Este é descrito por uma equação a diferenças no domínio do tempo, a qual tem seu formato caracterizado pela Equação 3.7 [57]:

$$y[n] = \sum_{j=0}^{M} b_j x[x - j] - \sum_{j=1}^{D} \mathrm{a}_j y[n - j] \tag{3.7}$$

onde $y[n]$ é a saída, $x[n]$ a entrada, b_j(j = 0,1,...,M) e a_j (j = 1,...D) são coeficientes do sistema.

Aplica-se a transformada Z na Equação 3.7, juntamente com a propriedade que estabelece a relação $x[n - n_0] \leftrightarrow z^{-n_0}X(z)$ e obtém-se a Equação 3.8:

$$Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z) - \sum_{j=1}^{D} \mathrm{a}_j z^{-k} Y(z) \tag{3.8}$$

onde a_0 normalmente é 1, o que nos permite reescrever a função de transferência $Q(z)$ do sistema através da Equação 3.9 [57]:

$$Q(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{D} \mathrm{a}_k z^{-k}} \tag{3.9}$$

Analisando-se a Equação 3.8 vê-se que o primeiro somatório não é recursivo, ou seja, ele depende apenas das amostras passadas do sinal de entrada $x[n]$ e está ligado às respostas ao impulso finitas (FIR). Já o segundo somatório contém elementos recursivos e amostras passadas do sinal $y[n]$ são necessárias para se obter a saída, o que representa um filtro recursivo. Estes filtros, devido a recursividade, produzem uma resposta ao impulso infinita (IIR) [44].

No geral, filtros FIR são preferidos em diversas aplicações se comparados aos filtros IIR, principalmente para a estimação de energia no TileCal. Isso se dá pois, os primeiros, são sempre estáveis e não possuem realimentação. Esta, pode aumentar o erro do sistema e diminuir consideravelmente seu desempenho [44].

4. TÉCNICAS PARA ESTIMAÇÃO DE ENERGIA NO L1

Neste trabalho se propõe uma técnica alternativa para implementação do filtro FIR além das direta ou transposta, as quais, apesar de apresentarem bom desempenho, no contexto de calorimetria, são mais custosos pelo fato de cada célula do calorímetro requisitar a implementação de uma estrutura do filtro, o que exige mais *hardware*. Por isso, se irá estudar um método sequencial utilizando a máquina de estados como núcleo de cálculo, e aliada ao processador SAPHO, como co-processador, executando os cálculos matriciais de produto interno das formas iterativas de Reconstrução de energia.

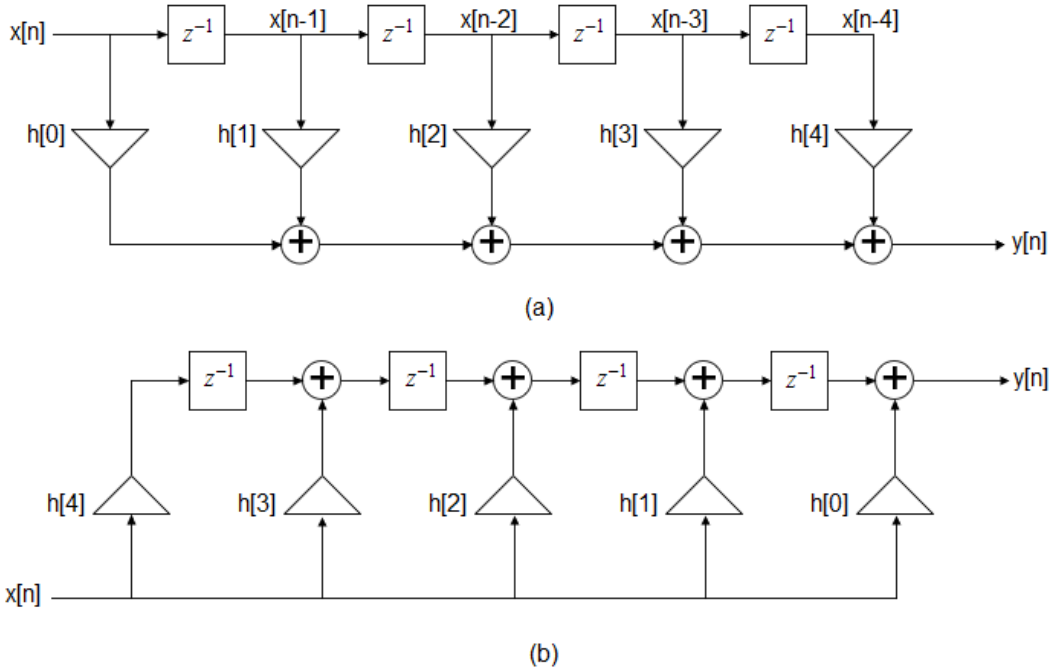
4.1 IMPLEMENTAÇÃO DE FILTROS FIR

Um filtro, por se tratar de um sistema LTI (***L**inear **T**ime-**I**nvariant system*, ou sistema linear invariante no tempo), pode ser representado por sua função de transferência escrita em formato de equações diferenças, como visto anteriormente. Com isso, reduzimos a Equação 3.7 para a Equação 4.1, que descreve o filtro FIR:

$$y[n] = \sum_{j=0}^{M} b_j x[x - j] \tag{4.1}$$

Existem algumas estruturas para implementação *online* da Equação 4.1, dentre elas pode-se citar: a direta, a em cascata e a por meio da decomposição polifásica [50]. A forma mais simples de implementação é a direta, a qual pode ser visualizada com o auxílio da Figura 19(a). Existe uma variação da estrutura direta de implementação, denominada Forma Transposta, a qual pode ser visualizada na Figura 19(b). Já a em cascata, mostrada na Figura 20, é normalmente utilizada em sistemas com altas ordens [50]. Existem outras estruturas, como a decomposição polifásica por exemplo, que é uma estruturação mais eficiente computacionalmente, se comparada à forma direta e, no geral, é aplicada quando se trabalha com processamentos multi-taxa de sinais.

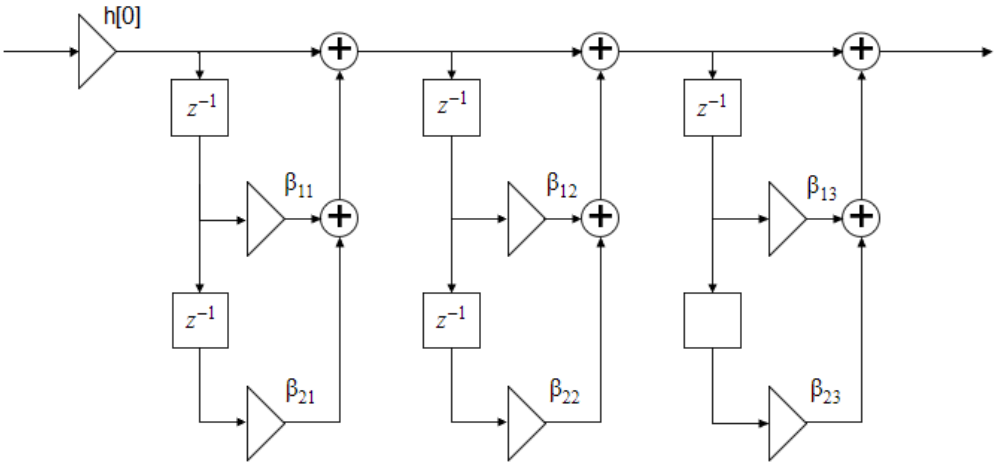
Figura 19 - Estruturas de representação de um filtro FIR.



Fonte: Do Autor.

A forma direta e a forma transposta são apenas diferentes maneiras de se computar a soma. Na teoria elas são idênticas mas, quando computadas com precisão finita, podem haver diferenças entre ambas. No que diz respeito ao ambiente FPGA, a forma transposta possui uma arquitetura com circuitos combinacionais menores o que diminui a utilização de recursos lógicos. Este trabalho irá focar apenas nestas duas formas de implementação.

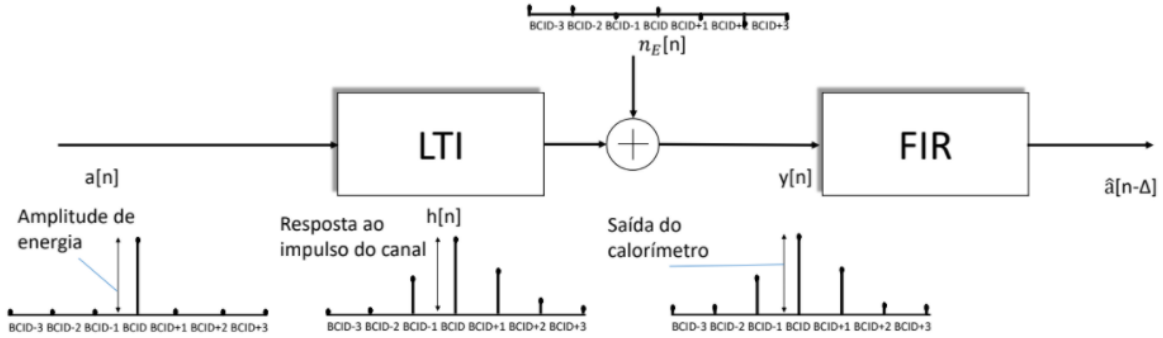
Figura 20 - Estrutura de representação de um filtro FIR em cascata.



Fonte: Do Autor.

Modelando-se toda a cadeia eletrônica de leitura do TileCal no tempo discreto como um LTI, pode-se criar um diagrama de blocos que que representa o sinal medido do mesmo, que pode ser visualizado como o resultado da convolução entre sua resposta impulsiva $h[n]$ com a combinação linear de versões atrasadas da sequência impulso unitário $\mathrm{a}[n]$, somados a um ruído branco gaussiano proveniente da sua eletrônica de leitura. Esse diagrama de blocos, bem como essa relação, podem ser visualizados na Figura 21. Um bloco representando um filtro FIR digital é adicionado em cascata ao modelo do calorímetro. Este, tem o propósito de equalizar, ou deconvoluir, o sinal $y[n]$, recuperando uma estimativa do sinal de entrada do sistema LTI, denominada por $\hat{\mathrm{a}}[n]$, sendo esta uma estimativa do valor de energia [44].

Figura 21 - Representação em blocos do calorímetro como um sistema linear.



Fonte: Imagem elaborada na tese de doutorado de João Paulo Bittencourt da Silveira Duarte [44].

Essas formas de implementação, ainda que que funcionem bem, ainda gastam muitos elementos lógicos e, quando se considera o tanto de sensores que se está trabalhando e o tanto de dados que necessitam ser processados de forma *online*, estratégias devem ser criadas para minimizar a utilização desses elementos.

A partir de agora, serão descritas algumas teorias de filtros FIR necessárias para entendimento das técnicas utilizadas em [44] para, com isso, se poder realizar comparações importantes no que diz respeito aos resultados. Uma delas é a aplicação de filtros FIR para sinais com natureza estocástica.

4.1.1 Filtros FIR para sinais estocásticos

Na prática, a resposta ao impulso no TileCal não possui uma natureza determinística, ou seja, não se tem uma descrição física completamente conhecida para a mesma, visto que existem alguns aspectos aleatórios envolvidos, como: deformações no pulso característico com relação à amplitude, saturação, desvios de fase, adição de ruído, etc [44]. Por isso, uma abordagem estocástica baseada na teoria da filtragem adaptativa é utilizada. Neste caso, se está simulando sinais com características de empilhamento de sinais similares aos que podem ser encontrados no TileCal [61].

A teoria da filtragem adaptativa descreve filtros que têm seus coeficientes ajustados ao longo do tempo, de modo a acompanhar as mudanças que ocorrem no sistema. Pelo fato de a luminosidade variar com o tempo durante o período de aquisição em uma tomada de dados no LHC, tais filtros são necessários para estabelecer uma relação entre seus coeficientes e a luminosidade no instante da colisão. Mais precisamente, em [44], a aplicação estudada e a que será usada como comparação neste trabalho, foi a Modelagem Inversa.

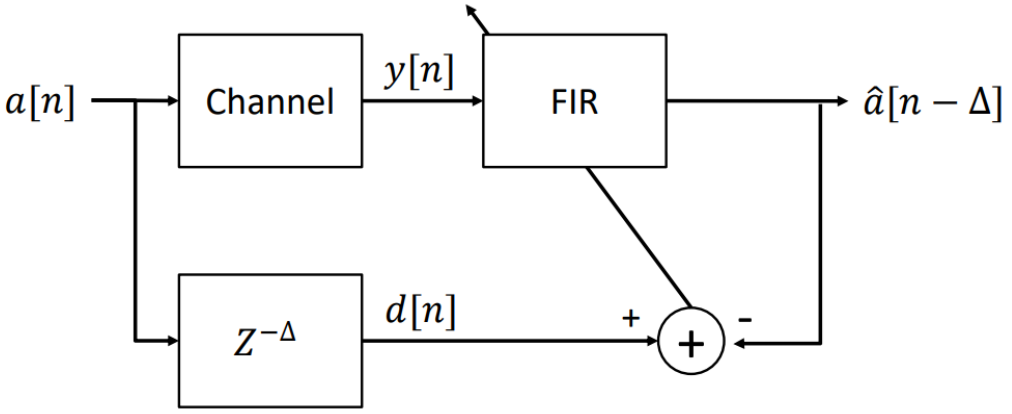
4.1.1.1 Modelagem Inversa

Nesta aplicação o filtro produz, para um sistema linear desconhecido, um modelo inverso, ou seja, ele possui uma função de transferência equivalente à do sistema inverso da planta desconhecida.

Na figura 22 tem-se a ilustração da modelagem inversa na qual, com o conhecimento prévio de uma entrada e do sinal observado na saída do sistema, é possível projetar um filtro FIR que se aproxima à deconvolução do canal [44]. Pela mesma figura percebe-se, também, que, o erro entre o sinal de entrada atrasado e a saída do filtro é utilizado no projeto dos coeficientes do mesmo.

De forma geral, filtros adaptativos são utilizados em aplicações *online*, ou seja, em tempo real, o que gera uma demanda de custo e elaboração de *hardware* excessivos para o projeto [63].

Figura 22 - Modelagem Inversa de Filtros Adaptativos.



Fonte: Imagem retirada do livro *Adaptive Filter Theory* [63].

Na prática, se desenvolve um filtro FIR que aproxima a resposta do sistema inverso utilizando-se algum critério de otimização. Normalmente, o método do *Least Squares*, ou dos Mínimos Quadrados, é aplicado, o qual minimiza o erro médio quadrático entre o valor desejado e a saída do filtro [62]. Essa aproximação é feita por este método pelo fato de, o mesmo, demandar apenas que se assuma um modelo de sinal. A próxima seção detalha melhor este processo.

4.1.1.2 Estimação utilizando o método dos Mínimos Quadrados

Esta estimação resume-se na escolha dos parâmetros $\hat{\mathbf{g}}$ que minimizam a Equação 4.2:

$$J(\hat{\mathbf{g}}) = \sum_{n=0}^{W-1} (\mathrm{a}[n] - \hat{\mathrm{a}}[n])^2 \tag{4.2}$$

A Equação 4.2 descreve a função custo J , a qual é o critério de erro adotado pelo estimador. Nela temos: W representando o número de amostras coletadas, $\mathrm{a}[n]$ as amostras consecutivas do sinal desejado e $\hat{\mathrm{a}}[n]$ depende linearmente de $y[n]$, vetor contendo valores da saída do sistema, e $\hat{\mathbf{g}}$. Essa função custo pode ser reescrita de forma matricial, em função de $\hat{\mathbf{g}}$ e, este parâmetro, definirá a ordem do filtro através de sua dimensão. Sendo assim, considerando-se que $\hat{\mathbf{g}}$ possua

dimensão $U \times 1$ e que $\mathbf{Y}$ seja uma matriz de observação $W \times U(W > U)$ formada pelo deslocamento das amostras $y[n]$. Desta forma, pode-se reescrever a função custo J na forma matricial, em função de $\hat{\mathbf{g}}$, e obtém-se a Equação 4.3:

$$J(\hat{\mathbf{g}}) = (\mathbf{a} - \mathbf{Y}\hat{\mathbf{g}})^T(\mathbf{a} - \mathbf{Y}\hat{\mathbf{g}}) \tag{4.3}$$

Manipulando-se a Equação 4.3, obtém-se a Equação 4.4 a seguir:

$$J(\hat{\mathbf{g}}) = \mathbf{a}^T\mathbf{a} - \mathbf{a}^T\mathbf{Y}\hat{\mathbf{g}} - \hat{\mathbf{g}}^T\mathbf{Y}^T\mathbf{a} + \hat{\mathbf{g}}^T\mathbf{Y}^T\mathbf{Y}\hat{\mathbf{g}}$$

$$J(\hat{\mathbf{g}}) = \mathbf{a}^T\mathbf{a} - 2\mathbf{a}^T\mathbf{Y}\hat{\mathbf{g}} + \hat{\mathbf{g}}^T\mathbf{Y}^T\mathbf{Y}\hat{\mathbf{g}} \tag{4.4}$$

Para minimizar a função custo precisa-se, primeiramente, da derivada de J em relação a $\hat{\mathbf{g}}$, obtendo-se a Equação 4.5:

$$\frac{\partial(\hat{\mathbf{g}})}{\partial\hat{\mathbf{g}}} = -2\mathbf{Y}^T\mathbf{a} + 2\mathbf{Y}^T\mathbf{Y}\hat{\mathbf{g}} \tag{4.5}$$

fazendo-se a Equação 4.5 igual a zero e isolando-se $\hat{\mathbf{g}}$, assumindo que $\mathbf{Y}^T\mathbf{Y}$ garanta a invertibilidade, obtém-se a Equação 4.6:

$$\hat{\mathbf{g}} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{a} \tag{4.6}$$

A Equação 4.6 mostra que a relação entre os pesos de $\hat{\mathbf{g}}$ e o sinal de entrada é dada pela pseudo inversa da matriz de observação $\mathbf{Y}$ [62]. O vetor de parâmetros $\hat{\mathbf{g}}$ estimado corresponde à própria resposta ao impulso de um sistema inverso ao sistema desconhecido. Logo, $\hat{\mathbf{g}}$ pode ser interpretado como os coeficientes de um filtro FIR de ordem $U - 1$ que deconvolui o sistema desconhecido [44].

Considerando-se $y[n]$ como um vetor de sinal recebido do TileCal e $\mathrm{a}[n]$ como um vetor de energias por BC de uma célula do TileCal, $\hat{\mathbf{g}}$ se torna um filtro FIR que, ao ser implementado de maneira *online*, deconvolui o sinal recebido e realiza a estimação da energia da partícula incidente.

As técnicas apresentadas até agora, neste capítulo, já foram estudadas e analisadas no trabalho [44]. O objetivo agora é apresentar o código da máquina de estados desenvolvida e comparar os resultados lá obtidos, nas formas direta e transposta de implementação de filtros FIR, e em seguida comparar os resultados.

O código em verilog da máquina será mostrado e explicado no capítulo a seguir.

5. IMPLEMENTAÇÃO

A máquina de estados proposta neste trabalho possui quatro estados representados pela letra *q*, e esta recebe os dados provenientes das colisões e realiza, de forma *online*, o cálculo da reconstrução da energia da partícula incidente, mudando os estados de acordo com variáveis que são definidas no código verilog escrito no *software* Quartus da Intel.

O conceito é, basicamente, efetuar as contas do filtro FIR em um dos estados da máquina projetada, utilizando menos lógica e, assim, possibilitando a utilização de mais sensores simultaneamente.

O cruzamento de feixes de prótons ocorre, no TileCal, a cada 25 ns, ou seja, as colisões ocorrem com um *clock* de 40 MHz [27]. Essa é a taxa em que as informações estão sendo enviadas para o filtro FIR e, o mesmo, terá que efetuar os cálculos internos na máquina de estados para gerar os valores de estimação de energia. Sendo assim, o *clock* da máquina deverá ser maior que os 40 MHz com os quais o LHC opera. Com 15 amostras se teria 600 MHz de *clock* mínimo para se operar de forma *free-running* o que, para os FPGAs do CERN é um valor razoável, visto a capacidade dos mesmos.

De antemão precisa-se definir todos os coeficientes do filtro FIR, os quais são constantes e foram obtidos do trabalho [60]. Na Figura 23 tem-se os mesmos sendo inicializados e armazenados em quinze registradores *w*, que é o número de coeficientes utilizados neste trabalho.

Figura 23 - Primeiras linhas de código e inicialização dos coeficientes do filtro FIR.

```
module SM
(
    input                       clk, rst, ready_in,
    input  signed [9:0]         in,
    output     reg              req_in,
    output reg signed [15:0] out,
    output reg                  ena_out
);

reg [1:0] q;
reg [3:0] cnt;

reg signed [15:0] sr [14:0];
reg signed [15:0] w  [14:0];


initial begin

    w[00] = -64'd860;
    w[01] = 64'd2821;
    w[02] = -64'd6389;
    w[03] = 64'd12744;
    w[04] = -64'd24204;
    w[05] = 64'd44714;
    w[06] = -64'd80839;
    w[07] = 64'd137908;
    w[08] = -64'd79619;
    w[09] = 64'd32798;
    w[10] = -64'd15077;
    w[11] = 64'd8229;
    w[12] = -64'd4192;
    w[13] = 64'd1784;
    w[14] = -64'd539;
end
```

Fonte: Do Autor.

Primeiramente, é criada uma variável denominada *ready_in* que tem a função de informar para o programa quando um dado novo (variável *in*) está disponível para ser processado pelo algoritmo. O dado, então, é requisitado pelo programa com a variável *req_in*. Esta variável tem o intuito de frear os dados à medida que eles chegam, de modo que o programa execute o processamento apenas da amostra atual.

Pode-se, portanto, criar o primeiro estado da máquina de estados, que pode ser visto na Figura 24, que é o programa checando se há algum dado novo e requisitando o mesmo para ser processado.

Percebe-se, pelo código da Figura 24, que, enquanto *ready_in* não tiver seu estado mudado, ou seja, não mudar de nível lógico, a variável *q*, que corresponde ao estado atual, permanecerá inalterada. Assim que *ready_in* sinalizar que houve uma colisão e o TileCal captou este dado, *req_in* irá requisitar o dado e, o mesmo, será armazenado em um *shift register*. A variável de estado *q* receberá o valor correspondente ao próximo estado. Na prática, enquanto estiverem ocorrendo colisões e o LHC estiver em operação, a variável *ready_in* terá nível lógico 1. No caso de *ready_in* ser 0, colisor desligado, *req_in* também será 0 pois não há a necessidade de se requisitar nada, visto que não há informação alguma no sistema.

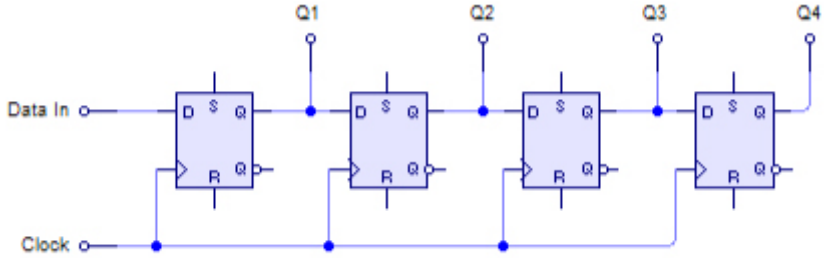
Figura 24 - Primeiro estágio da máquina de estados.

```
36    always @ (posedge clk)
37  begin
38      if (rst)
39          q <= 2'd0;
40      else
41          begin
42              case (q)
43              2'd0: begin
44                        ena_out <= 1'b0;
45                        if (ready_in)
46                            begin
47                                req_in <= 1'b1;
48                                q      <= 2'd1;
49                            end
50                        else
51                            begin
52                                req_in <= 1'b0;
53                                q      <= q;
54                            end
55                    end
```

Fonte: Do Autor.

Os *Shift registers* são circuitos digitais que usam *flip-flops* ligados em cascata onde a saída de um é a entrada do próximo. Neste código eles têm a função de atrasar a informação no tempo, de modo a realizar as operações na ordem correta. A representação lógica deles pode ser vista na Figura 25.

Figura 25 - Ilustração de um *shift register* de 4 bits.



Fonte: Imagem retirada do site Embarcados [59].

Considerando-se que *ready_in* tem nível lógico 1, sabe-se que há dados novos a serem processados e faz-se a requisição dos mesmos, *req_in* passa a ter nível lógico 1, e a máquina muda de estado, *q* passa para o valor do estado 2. No segundo estado, *req_in* retorna para 0 já que precisamos, primeiramente, trabalhar com a informação atual antes de requisitar a próxima. O dado será armazenado em um *shift register*, mantendo as informações anteriores armazenadas. Isso pode ser visto na Figura 26, onde o registrador $sr[01]$ recebe o valor de $sr[00]$, $sr[02]$ de $sr[01]$ e assim sucessivamente, atrasando a informação de entrada no tempo.

Figura 26 - Segundo estágio da máquina de estados.

```
56          2'd1: begin
57                    req_in <= 1'b0;
58                    sr[14] <= sr[13];
59                    sr[13] <= sr[12];
60                    sr[12] <= sr[11];
61                    sr[11] <= sr[10];
62                    sr[10] <= sr[09];
63                    sr[09] <= sr[08];
64                    sr[08] <= sr[07];
65                    sr[07] <= sr[06];
66                    sr[06] <= sr[05];
67                    sr[05] <= sr[04];
68                    sr[04] <= sr[03];
69                    sr[03] <= sr[02];
70                    sr[02] <= sr[01];
71                    sr[01] <= sr[00];
72                    sr[00] <= in;
73                    q <= 2'd2;
74                    out <= 20'd0;
75                    cnt <= 4'd0;
76                end
```

Fonte: Do Autor.

Ainda no segundo estado, inicia-se uma variável *out* com valor zero, a qual, futuramente, receberá o valor dos cálculos realizados pelo filtro FIR de forma acumulativa, e um contador *cnt* que contará os *shift registers* e será responsável por indicar qual registrador está sendo usado no momento para realizar o cálculo e, também, encerrar o cálculo e, depois de todas as operações feitas, mudar para o estado final da máquina de estados.

No terceiro estado da máquina é que serão realizados os cálculos propriamente ditos para a saída do filtro. Esse cálculo não poderá ser realizado em apenas um ciclo de clock, pois o mesmo problema de utilização de recursos lógicos das aplicações usuais de filtro FIR ocorrerá. Assim, cria-se um loop para realização da operação, sequencialmente, com o intuito de criar apenas um circuito de multiplicação e ir acumulando os valores.

Na primeira iteração, como pode ser visto na Figura 27, coleta-se a entrada, a qual está armazenada no *shift register* $sr[00]$, o multiplicado pelo primeiro coeficiente do filtro FIR $w[00]$, e atribui o resultado ao acumulador.

Figura 27 - Terceiro estágio da máquina de estados.

```
77          2'd2: begin
78                    out <= out + sr[cnt] * w[cnt];
79                    cnt <= cnt + 4'd1;
80                    if (cnt == 4'd14)
81                          q <= 2'd3;
82                    else
83
84                          q <= q;
85
86                end
```

Fonte: Do Autor.

Este circuito é o mais custoso do projeto, então ele só poderá aparecer nesse estado da máquina, para que a utilização de recursos de *hardware* seja a menor possível. Se ele for replicado em algum outro lugar do código, a função da proposta passa a não ter sentido, visto que este é justamente o problema das implementações usuais de filtros FIR, a criação de um circuito de soma e multiplicação para cada iteração.

O último estado da máquina é responsável por validar a saída para ser analisada através da variável *ena_out* que, quando está em valor lógico 1, sinaliza que o valor calculado no estado anterior está disponível e pronto para ser utilizado e

que todas as operações se encerraram, pode-se ver isso na Figura 28. Além disso, o valor *q* é mudado para voltar para o valor correspondente ao primeiro estado da máquina, onde *ena_out* recebe o valor 0 novamente, sinalizando que uma nova rodada de dados será processada.

Figura 28 - Último estágio da máquina de estados.

```
87          2'd3: begin
88                    ena_out <= 1'b1;
89                    q <= 2'd0;
90                end
91            endcase
92        end
93    end
94    endmodule
```

Fonte: Do Autor.

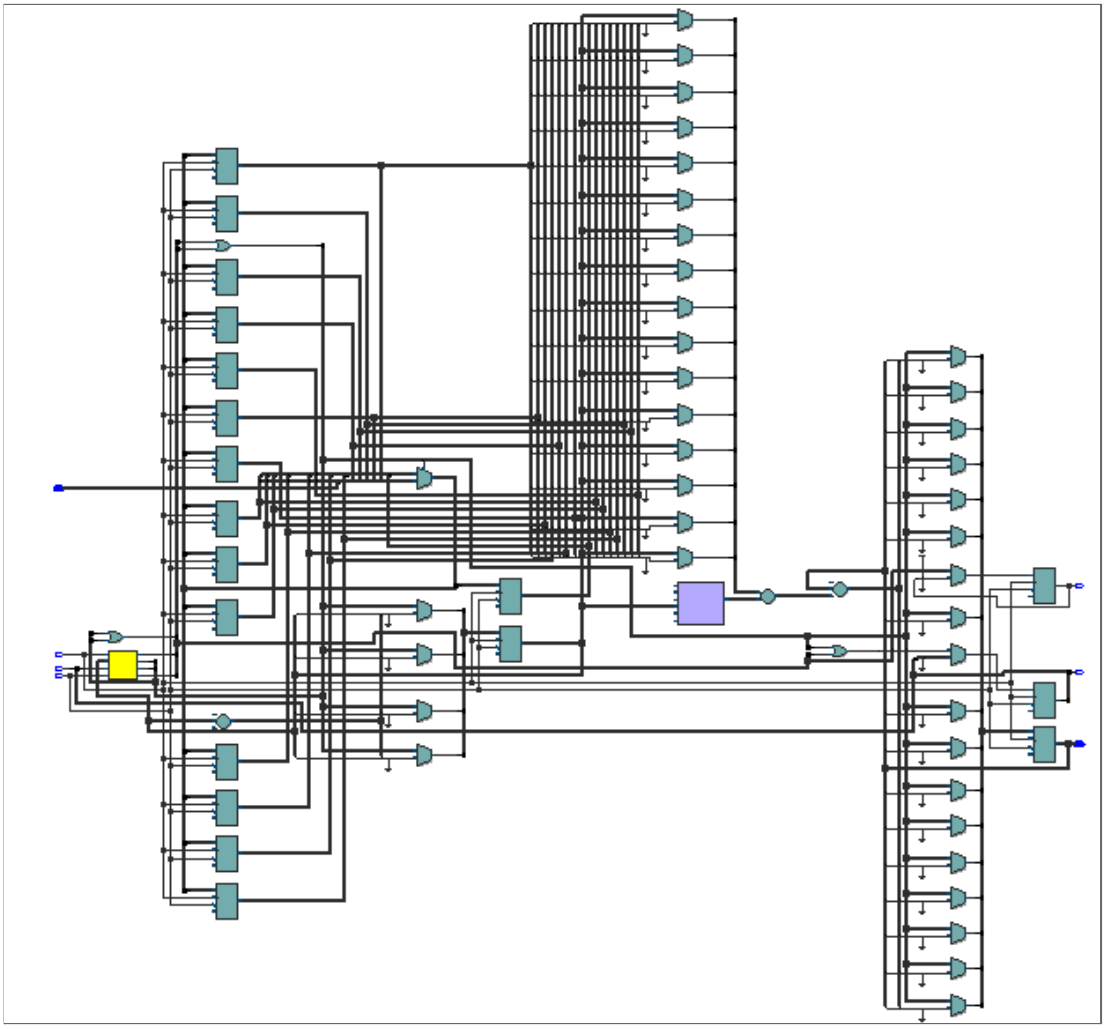
Com isso, o valor da variável *out* contém a estimação da energia da partícula proveniente da colisão. Claramente, os cálculos estão sendo realizados em cada sensor do TileCal e, muitas vezes, para cada partícula que incide no mesmo, dentro da taxa de *clock* de 40 MHz com que ocorrem as colisões. Aqui foi explicado de forma simples o que está ocorrendo em cada sensor, no caso de uma só partícula incidir no mesmo.

Dos 10.000 canais encontrados no TileCal, cerca de metade apresenta o efeito de empilhamento de sinais e a outra metade não. Para os primeiros, o filtro FIR terá ordem 14, ou seja, 15 amostras, enquanto para os segundos o filtro terá ordem 6, ou 7 amostras. Depois de feitos os cálculos, os resultados são disponibilizados serialmente, amostra por amostra, de forma free-running, a medida que entra um dado o resultado é habilitado.

A seguir, através de sinais simulados com as mesmas características de empilhamento de sinais encontradas no TileCal, a eficácia da implementação proposta é testada e se realiza a comparação com as estruturas direta e transposta de filtros FIR.

A representação do circuito *top level* do código implementado está representada na Figura 29. A mesma traz benefícios no que se refere à otimização do consumo de *hardware*, ou seja, quantidade de elementos lógicos despendidos na operação, esta afirmação será comprovada mais a frente.

Figura 29 - *Top Level* demonstrativo do filtro FIR por Máquina de Estados.



Fonte: Do Autor.

No capítulo seguinte, são comparados os conceitos estudados em [44] de modo a, através dos resultados lá obtidos, realizar simulações com o objetivo de analisar o desempenho da estruturação proposta neste trabalho.

De posse do código da máquina de estados projetada e com valores prévios de colisões do LHC é feita uma análise e comparação do funcionamento e da eficiência desta estruturação, com os resultados obtidos em [44], no que se refere a otimização dos recursos de *hardware* e ao erro RMS (***R**oot **M**ean **S**quare*).

O erro RMS é calculado pela raiz quadrada média do erro absoluto entre os valores de energia estimados pelo filtro e os valores reais de energia da simulação, este último é chamado de sinal verdadeiro.

6. RESULTADOS

Neste capítulo, são apresentados os resultados obtidos em [44] com as formas transposta e direta de filtros FIR para sinais estocásticos com estimação utilizando o método dos Mínimos Quadrados. Esses mesmos resultados são comparados com os obtidos pela forma descrita por máquina de estados, a qual é proposta neste trabalho, utilizando a mesma forma de estimação e os mesmo sinais.

Para realização das simulações foi utilizado uma *Toy Monte Carlo Simulation*, que se trata de uma técnica matemática usada para estimar o resultado de um evento incerto, ou seja, com características probabilísticas. Neste caso, estão sendo simulados sinais com características de empilhamento de sinais similares aos que podem ser encontrados no TileCal [61]. Os detalhes desta simulação estão explicados em detalhes em [44] e não serão o foco deste trabalho, pois estamos interessados apenas nas comparações entre os resultados lá obtidos, através dessa simulação, com os obtidos com a máquina de estados aqui implementada. Os parâmetros utilizados, para contextualização, são: N amostras de ruído branco gaussiano representando as amostras digitalizadas do TileCal, a razão entre sinal e ruído, chamada de ΄ϒ, a variação da ocupância, a qual baseia o sorteio de posições onde ocorreram as deposições de energia, e a média exponencial. Precisa-se, primeiramente, realizar uma análise para encontrar quantos bits mínimos são necessários para que o projeto utilize a menor quantidade possível de recursos lógicos e, depois, achar a relação ótima entre os bits, os elementos lógicos, o erro RMS e quantos bits serão destinados para a parte inteira e fracionária dos coeficientes.

Assim, de posse dos sinais simulados pode-se efetuar testes com o intuito de checar a máquina de estados projetada. Para isso, fixa-se os valores de ΄ϒ, da ocupância e da média exponencial em 4,33%, 30% e 30, respectivamente. Foram geradas 100.000 amostras com estes parâmetros, das quais 50.000 foram utilizadas para o projeto do filtro FIR, pelo método dos Mínimos Quadrados, e as 50.000 demais para testar a implementação da máquina de estados proposta. Tal análise é apresentada a seguir.

6.1 SIMULAÇÃO E ANÁLISE DO NÚMERO DE BITS

Neste capítulo é apresentado um estudo para se obter a quantidade de bits ótima para que o programa otimize a utilização dos recursos de *hardware* e possua o menor erro RMS possível.

Primeiramente fez-se uma extrapolação dos valores e foi-se diminuindo os bits da parte fracionária até que o erro RMS ficasse muito grande. Esses valores são:

- 16 bits para a parte fracionária dos coeficientes, multiplicando-se os mesmos por 2^{16} para deslocar a parte fracionária. Esse valor foi fixado nos testes; e

- 128 bits totais para os coeficientes. Esse valor foi diminuído para se encontrar o número de bits com o menor erro RMS e menor uso de recursos de *hardware*.

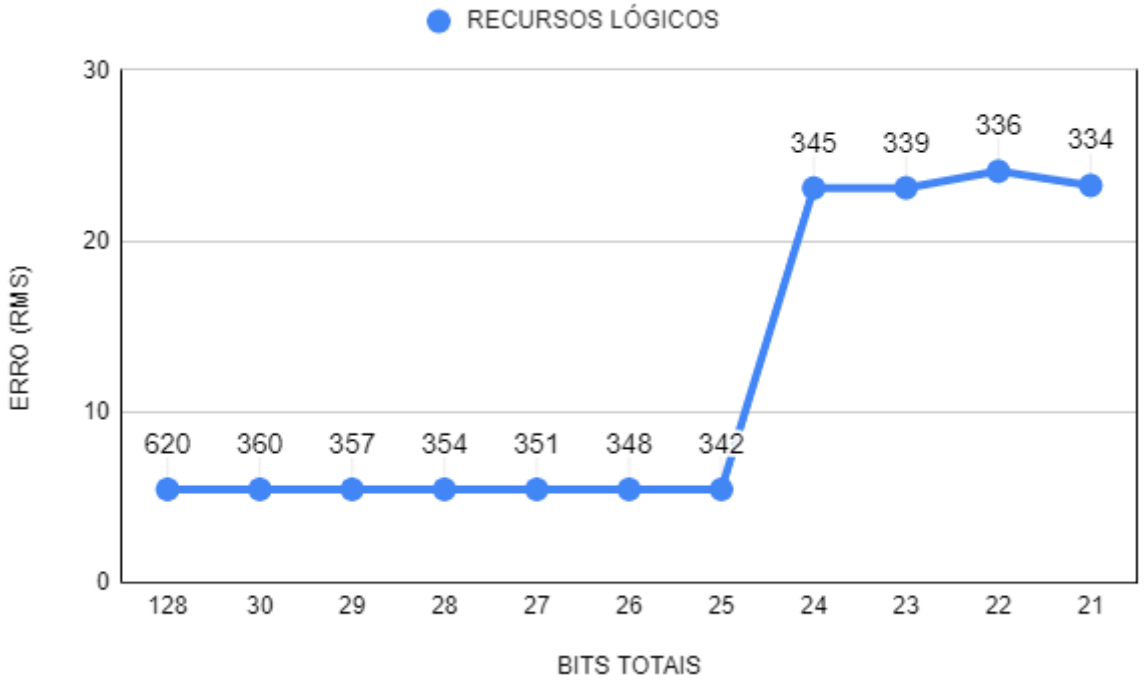
Os dados obtidos podem ser vistos na Tabela 3 e, de posse deles, pode-se fazer um gráfico, o qual está descrito na Figura 30.

Tabela 3 - Análise do número de bits e elementos lógicos com 2^{16}.

BITS TOTAIS	RECURSOS	ERRO RMS
128	620	5.4579
30	360	5.4579
29	357	5.4579
28	354	5.4579
27	351	5.4579
26	348	5.4579
25	342	5.4579
24	345	23.0928
23	339	23.1074
22	336	24.0768
21	334	23.2482

Fonte: Do Autor.

Figura 30 - Gráfico do erro RMS variando com o número de bits.



Fonte: Imagem do autor feita no Google Planilhas.

Analisando-se a Tabela 3 e o gráfico da Figura 30, percebe-se que 25 é o valor mínimo de bits totais em que o erro RMS e os elementos lógicos são pequenos.

A partir de agora, com base nesses valores, o número de bits da parte fracionária e os totais serão diminuídos até se encontrar um valor em que o erro RMS permaneça pequeno, bem como os elementos lógicos usados. Faz-se, então, a seguinte análise:

- Diminui-se os 16 bits da parte fracionária dos coeficientes, ou seja, diminui-se o expoente n do multiplicador dos coeficientes 2^n; e

- Paralelamente, varia-se o número de bits totais, até se chegar num valor em que o erro RMS e os recursos lógicos utilizados são satisfatórios.

Os dados desta análise se encontram na Tabela 4 e pode-se exibi-los no gráfico da Figura 31. Para esta análise utilizou-se mais casas decimais para o erro RMS, para que se possa perceber suas variações mais precisamente.

Tabela 4 - Relação entre número de bits, o erro RMS e os recursos utilizados.

BITS TOTAIS	BITS PARTE FRACIONÁRIA	ERRO RMS	RECURSOS
25	16	5.716629726265173	345
24	15	5.716267154014929	335
23	14	5.717628671680360	332
22	13	5.718225700686228	331
21	12	5.721438771888248	328
20	11	5.707693418652384	323
19	10	5.695967278209430	321
18	9	5.752260188463917	317
17	8	5.730785402938695	315
16	7	5.592900896316842	312
15	6	5.811354401310896	310
14	5	6.053529701222883	307
13	4	5.826456602108958	303
12	3	5.775855093342436	300
11	2	10.384827649359632	298
10	1	14.174547623161901	343
9	0	27.936090825619548	303

Fonte: Do Autor.

Figura 31 - Gráfico da relação entre número de bits, o erro RMS e recursos lógicos.

Fonte: Imagem do autor feita no Google Planilhas

Percebe-se que para 16 bits totais, com 7 destinados para a parte fracionária e os 9 restantes para a parte inteira dos coeficientes do filtro FIR, tem-se o menor erro RMS e a utilização de 312 recursos lógicos pelo FPGA. Como o padrão para escolha desses parâmetros é o menor erro RMS, esses valores serão fixados para que se possa realizar uma análise comparativa entre os resultados obtidos através das implementações tradicionais de filtros FIR, como as formas direta e transposta e a proposta por máquina de estados.

Deve-se lembrar que, quando se está trabalhando com variáveis do tipo *signed*, a codificação do valor é feita através de Complemento de 2, ou seja, quando se está utilizando 16 bits e se quer representar um número negativo, ele é representado dessa forma e, os 65.536 valores estão no intervalo de -32.768 e +32.768, ou seja, essa é a magnitude máxima que as operações podem chegar.

6.1.1 Análise da implementação por Máquina de Estados

Primeiramente, os coeficientes do filtro FIR são gerados computacionalmente através do método matricial dos Mínimos Quadrados, com a Equação 4.6, utilizando-se a *Toy Monte Carlo Simulation* já explicada. Obtém-se, então, um vetor de dados chamado “Hls”. Este vetor é multiplicado pelo valor 2^7, o que desloca a parte fracionária dos valores e, assim, podemos utilizá-los na máquina de estados nos registradores *w* da Figura 23, porém utilizando 16 bits totais no lugar dos 64 lá mostrados. Na Figura 32 está descrita a inicialização desses coeficientes nos registradores, já com os parâmetros fixados. No *software* Quartus, através do ModelSim, foi feita a simulação da máquina, utilizando o vetor “*input_data*” como entrada *in*, gerado pegando-se o sinal do TileCal e o colocando no formato de 10 bits. Esse sinal foi armazenado nos *shift registers* e rodou-se a simulação. A máquina de estados, então, realiza os cálculos e gera o vetor *“out”* que possui os valores de energia estimada das partículas. O relatório da compilação pode ser visto na Figura 33, o qual contém o dado elementos lógicos totais utilizados pelo processo. Este é o dado mais importante para este trabalho, visto que o objetivo é obter o menor uso de recursos de *hardware*, obtendo-se os mesmos resultados.

Figura 32 - Inicialização dos coeficientes do filtro FIR com 16 bits.

```
17    initial begin
18
19    /*1*/     w[00] = -16'd2;
20    /*2*/     w[01] = 16'd5;
21    /*3*/     w[02] = -16'd12;
22    /*4*/     w[03] = 16'd25;
23    /*5*/     w[04] = -16'd47;
24    /*6*/     w[05] = 16'd87;
25    /*7*/     w[06] = -16'd158;
26    /*8*/     w[07] = 16'd269;
27    /*9*/     w[08] = -16'd155;
28    /*10*/    w[09] = 16'd63;
29    /*11*/    w[10] = -16'd29;
30    /*12*/    w[11] = 16'd15;
31    /*13*/    w[12] = -16'd7;
32    /*14*/    w[13] = 16'd3;
33    /*15*/    w[14] = -16'd1;
34    end
```

Fonte: Do Autor.

Figura 33 - Relatório de compilação da implementação por Máquina de Estados.

Flow Status	Successful - Wed Sep 01 12:15:21 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	SM
Top-level Entity Name	SM
Family	Cyclone IV E
Total logic elements	312
Total registers	176
Total pins	31
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	2
Total PLLs	0

Fonte: Do Autor.

A partir de agora, esse dados serão comparados, no que se refere a utilização de recursos lógicos, com as implementações de filtros FIR nas formas direta e transposta.

6.1.2 Análise da implementação na forma direta

Os coeficientes obtidos foram utilizados em um código em *verilog* feito no Quartus, o qual foi projetado utilizando a forma direta de implementação de filtros FIR, com 16 bits totais, dos quais 7 foram destinados à parte fracionária. Este código e a forma de implementação podem ser vistas na Figura 34.

Figura 34 - Código de implementação de filtro FIR na forma direta.

```
module fir1
(
    input clk,
    input signed [9:0] x,
    output signed [15:0] y
);

reg signed [9:0] r [14:0];
reg signed [15:0] w  [14:0];


initial begin

/*1*/    w[00] = -16'd2;
/*2*/    w[01] = 16'd5;
/*3*/    w[02] = -16'd12;
/*4*/    w[03] = 16'd25;
/*5*/    w[04] = -16'd47;
/*6*/    w[05] = 16'd87;
/*7*/    w[06] = -16'd158;
/*8*/    w[07] = 16'd269;
/*9*/    w[08] = -16'd155;
/*10*/   w[09] = 16'd63;
/*11*/   w[10] = -16'd29;
/*12*/   w[11] = 16'd15;
/*13*/   w[12] = -16'd7;
/*14*/   w[13] = 16'd3;
/*15*/   w[14] = -16'd1;
end


always @ (posedge clk) begin
    r[0] <= x;
    r[1] <= r[0];
    r[2] <= r[1];
    r[3] <= r[2];
    r[4] <= r[3];
    r[5] <= r[4];
    r[6] <= r[5];
    r[7] <= r[6];
    r[8] <= r[7];
    r[9] <= r[8];
    r[10] <= r[9];
    r[11] <= r[10];
    r[12] <= r[11];
    r[13] <= r[12];
    r[14] <= r[13];
end

assign y = x*w[00]
+ r[0]*w[01] + r[1]*w[02]
+ r[2]*w[03] + r[3]*w[04]
+ r[4]*w[05] + r[5]*w[06]
+ r[6]*w[07] + r[7]*w[08]
+ r[8]*w[09] + r[9]*w[10]
+ r[10]*w[11] + r[11]*w[12]
+ r[12]*w[13] + r[13]*w[14];
endmodule
```

Fonte: Do Autor.

Como se pode perceber o circuito de soma do filtro está sendo gerado toda vez que ocorre uma colisão e novos dados chegam no filtro, o que consome um volume muito grande de recursos lógicos, como pode ser visto na Figura 35 no relatório de compilação da implementação direta do filtro.

Figura 35 - Relatório de compilação da implementação na forma direta.

Flow Status	Successful - Wed Sep 01 11:53:12 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	fir1
Top-level Entity Name	fir1
Family	Cyclone IV E
Total logic elements	818
Total registers	140
Total pins	27
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Fonte: Do Autor

Nota-se que, apesar de possuir um código simples e de fácil implementação, esta forma de implementação utilizou-se de 818 recursos lógicos para realizar as operações do filtro e fazer a reconstrução da energia da partícula. A saída que se obteve no vetor “out” pôde ser comparada para ambas as implementações e observou-se os mesmos valores de amplitude.

6.1.3 Análise da implementação na forma transposta

A forma transposta de implementação do filtro FIR foi implementada também utilizando-se 16 bits, sendo 7 para a parte fracionária dos coeficientes. O código no Quartus encontra-se na Figura 36.

Figura 36 - Código de implementação de filtro FIR na forma transposta.

```
1   module fir_trans
2   (
3      input clk,
4      input signed [9:0] x,
5      output signed [15:0] y
6   );
7   reg signed [15:0] r[14:0];
8   reg signed [15:0] w[14:0];
9
10
11  initial begin
12
13  /*1*/    w[00] = -16'd2;
14  /*2*/    w[01] = 16'd5;
15  /*3*/    w[02] = -16'd12;
16  /*4*/    w[03] = 16'd25;
17  /*5*/    w[04] = -16'd47;
18  /*6*/    w[05] = 16'd87;
19  /*7*/    w[06] = -16'd158;
20  /*8*/    w[07] = 16'd269;
21  /*9*/    w[08] = -16'd155;
22  /*10*/   w[09] = 16'd63;
23  /*11*/   w[10] = -16'd29;
24  /*12*/   w[11] = 16'd15;
25  /*13*/   w[12] = -16'd7;
26  /*14*/   w[13] = 16'd3;
27  /*15*/   w[14] = -16'd1;
28  end
29
30
```

```
31  always @ (posedge clk) begin
32
33  r[0]<=x*w[14];
34  r[1]<=r[0]+x*w[13];
35  r[2]<=r[1]+x*w[12];
36  r[3]<=r[2]+x*w[11];
37  r[4]<=r[3]+x*w[10];
38  r[5]<=r[4]+x*w[09];
39  r[6]<=r[5]+x*w[08];
40  r[7]<=r[6]+x*w[07];
41  r[8]<=r[7]+x*w[06];
42  r[9]<=r[8]+x*w[05];
43  r[10]<=r[9]+x*w[04];
44  r[11]<=r[10]+x*w[03];
45  r[12]<=r[11]+x*w[02];
46  r[13]<=r[12]+x*w[01];
47  end
48  assign y = r[13]+x*w[00];
49
50  endmodule
```

Fonte: Do Autor.

A forma transposta não possui circuitos computacionais muito extensos. Nesta implementação, todos os registradores mudam ao mesmo tempo na chegada de um dado novo, o que consome menos elementos lógicos, se comparado a forma direta de implementação. Na Figura 37 se encontra o relatório de compilação desta estruturação.

Figura 37 - Relatório de compilação da implementação na forma transposta.

Flow Status	Successful - Wed Sep 01 11:49:43 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	fir_trans
Top-level Entity Name	fir_trans
Family	Cyclone IV E
Total logic elements	569
Total registers	219
Total pins	27
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Fonte: Do Autor.

Em seguida, são apresentadas uma discussão e comparação sobre os resultados obtidos, de forma a investigar a utilidade, o funcionamento e as vantagens da implementação proposta, se comparada às usuais.

6.2 DISCUSSÃO

O objetivo, desde o princípio deste trabalho, foi propor uma estruturação de filtros FIR, em FPGA, que fosse mais vantajosa na utilização de *hardware*, para a implementação *online* na reconstrução da energia das partículas no TileCal.

Em [44] foram estudadas as formas direta e transposta de filtros FIR para sinais estocásticos, que também apresentam um erro RMS pequeno, utilizando a estimação de parâmetros do filtro pelo método matricial do algoritmo dos Mínimos Quadrados.

Estes códigos foram implementados em *verilog* e simulados no *software* Quartus utilizando sinais obtidos através de T*oy Monte Carlo Simulation* para serem comparados. Utilizaram-se 16 bits totais para os coeficientes, além de 7 bits para a parte fracionária, o que significa que os valores do vetor “Hls” foram multiplicados por 2^7.

Com todos os valores estabelecidos e o mesmo banco de dados para todas as simulações, pode realizar a comparação dos resultados de forma mais precisa, já

que espera-se a mesma saída “out” contendo os valores de energia, para todos os códigos implementados.

Para a implementação por máquina de estados, percebe-se que o código de implementação em *verilog* é bastante extenso o que, num primeiro olhar, pode dar a impressão de que esta é a mais trabalhosa entre as três formas.

Após realizada síntese e análise no Quartus constatou-se que um total de 312 elementos lógicos foram utilizados durante a operação, como pode ser visto na Figura 33.

Para a forma direta do filtro FIR, tem-se um código em *verilog* razoavelmente menor porém, no que diz respeito a uso de recursos de *hardware*, percebe-se que foram utilizados bem mais que o dobro, 818 como pode ser visualizado na Figura 35, o que confirma a investigação de que a forma proposta é mais vantajosa nesse quesito, se ambas forem comparadas. Isso se dá devido a presença de circuitos combinacionais mais extensos nessa estrutura, devido a presença de uma soma muito grande em sua saída.

Em [44] foi dito que a forma transposta de implementação é a que possui o circuito combinacional *top level* menos extenso. Isto acontece pelo fato desta forma de estruturação possuir várias pequenas somas separadas por elementos de atraso. Nota-se, pela Figura 37, que 569 elementos lógicos totais foram utilizados no processo. Este valor é bem menor do que o obtido pela forma direta de implementação.

Portanto, a forma de implementação de filtros FIR proposta neste trabalho foi, com bastante folga, a que otimizou melhor a utilização de recursos de *hardware*, dentre as testadas, o que confirma a usabilidade da mesma como a implementação sugerida para uso na reconstrução *online* de energia no TileCal.

Para a implementação do filtro FIR utilizando máquina de estados, o erro RMS entre os resultados obtidos e o sinal verdadeiro continuou baixo, o que confirma a viabilidade da aplicação proposta visto que, não apenas a utilização dos recursos de *hardware* foi otimizada, mas também os resultados são próximos aos esperados.

7. CONSIDERAÇÕES FINAIS

7.1 CONCLUSÕES

Com base nas teorias apresentadas na revisão bibliográfica e nas simulações realizadas, resta concluir que a forma de implementação de filtros FIR utilizando máquina de estados apresentou resultados muito satisfatórios no que diz respeito à utilização de recursos lógicos.

O ponto ótimo da implementação foi obtido ao se utilizar 16 bits totais para a saída e os coeficientes, sendo os estes multiplicados pelo fator 2^7 e possuindo 7 bits para sua parte fracionária. Com esses valores se obteve o menor erro RMS e minimizou-se a utilização de recursos de *hardware*.

Em comparação com as implementações de filtros FIR nas formas direta e transposta, utilizando o ponto ótimo para a quantidade de bits e o erro RMS, a máquina de estados apresentou o melhor resultado de utilização de recursos lógicos em comparação às outras estruturas.

7.2 TRABALHOS FUTUROS

Os resultados obtidos nas implementações foram bastante satisfatórios no que diz respeito a recursos de *hardware* utilizados. A grande vantagem de implementar a máquina de estados em FPGA é que este dispositivo nos permite trabalhar com clocks da ordem dos *Gigahertz*. No entanto, quando se começa a utilizar mais canais se começa a ter limitações, já que a ideia é processar mais canais ao mesmo tempo. Para trabalhos futuros seria interessante estudar a implementação de mais blocos desses filtros FIR por máquinas de estado ligados em paralelo, de modo a possibilitar que mais canais sejam utilizados no processamento e reduzindo, ainda mais, a utilização de recursos de *hardware* do processo.

REFERÊNCIAS

[1] ANTHONY, K. Celebrating the First of a Kind. Sep 2014.

[2] CERN. About CERN. CERN Document Server, 2012.

[3] NAKAHAMA, Y. The ATLAS Trigger System: Ready for Run-2. CERN Document Server, ATL-DAQ-PROC-2015-006, Geneva, 2015.

[4] ANDRADE FILHO, L. M.; PERALVA, B. S.; SEIXAS, J. M.; CERQUEIRA, A. S. Calorimeter Response Deconvolution for Energy Estimation in High-Luminosity Conditions. IEEE Transactions on Nuclear Science, 62(6):3265–3273, Dec 2015.

[5] WIGMANS, R. Calorimetry. Oxford University Press, 2018.

[6] DUARTE, J. P. B. S. Técnicas de Deconvolução Aplicadas à Estimação de Energia Online em Calorimetria de Altas Energias em Condições de Alta Taxa de Eventos. Anais do XX Congresso Brasileiro de Automática, pp. 1-6, 2016.

[7] CLELAND, W. E.; STERN, E. G. Signal Processing Considerations for Liquid Ionization Calorimeters in a High Rate Environment. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 338, pp. 467–497, Jan 1994.

[8] DUARTE, J. P. B. S. et al. Online energy reconstruction for calorimeters under high pile-up conditions using deconvolutional techniques. Journal of Instrumentation, v. 14, n. 12, p. P12017, 2019.

[9] CERN, C. About CERN. Acesso em Agosto de 2021. Disponível em: <http://home.web.cern.ch/about>.

[10] COEPP. ARC Centre of Excellence for Particle Physics at the Terascale COEPP. Acesso em Agosto de 2021. Disponível em: <http://www.coepp.org.au/files/coepp/images/cern-aerial2.jpg>.

[11] CERN, C. ATLAS. Acesso em Agosto de 2021. Disponível em: <http://home.web.cern.ch/about/experiments/atlas>.

[12] CERN, C. About CERN. Acesso em Agosto de 2021. Disponível em: <https://home.cern/science/physics/standard-model>.

[13] MOUCHE, P. Overall View of the LHC. CERN Document Server, OPEN-PHOACCEL- 2014-001, Jun, 2014.

[14] ADOLPHI, R. et al. The CMS experiment at the cern LHC. Jinst, v. 803, p. S08004–2008, 2008.

[15] EVANS, L.; BRYANT, P. LHC machine. Journal of Instrumentation, IOP Publishing, v. 3, n. 08, p. S08001, 2008.

[16] AAD, G. et al. The ATLAS experiment at the cern large hadron collider. Journal of Instrumentation, IOP Publishing, v. 3, n. 08, p. S08003, 2008.

[17] BRICE, M. LHC Restart Run 2. CERN Document Server, April, 2015.

[18] CERN, C. About CERN. Acesso em 16 Agosto de 2021. Disponível em: <http://home.web.cern.ch/about>.

[19] PEQUENAO, J. Computer Generated Image of the Whole ATLAS Detector. CERN Document Server, CERN-GE-0803012, Geneva, 2008.

[20] PALESTINI, S. The Muon Spectrometer of the ATLAS Experiment. Nuclear Physics B Proceedings Supplements, vol. 125, pp. 237-345, 2003.

[21] ALEKSA, M.; CLELAND, W.; ENARI, Y. ATLAS Liquid Argon Calorimeter Phase-I Upgrade Technical Design Report. CERN Document Server, CERN-LHCC-2013-017 and ATLAS-TDR-022, Geneva, 2013.

[22] CARRIO, F.; KIM, H. Y.; MORENO, P.; REED, R.; SANDROK, C. Design of an FPGA-Based Embedded System for the ATLAS Tile Calorimeter Front-End Electronics Test-Bench. CERN Document Server, ATL-TILECAL-PROC-2013-017, Geneva, 2013.

[23] PEQUENAO, J. Computer Generated Image of the ATLAS Calorimeter. CERN Document Server, CERN-GE-0803015, 2008.

[24] ANDRADE FILHO, L. M. Detecção e Reconstrução de Raios Cósmicos Usando Calorimetria de Altas Energias. Tese de Doutorado, COPPE/UFRJ, Março, 2009.

[25] PEQUENAO, J.; SCHAFFNER, P. How ATLAS Detects Particles: Diagram of Particle Paths in the Detector. CERN Document Server, CERN-EX-1301009, Geneva, Jan 2013.

[26] AAD, G. The ATLAS Experiment at the CERN Large Hadron Collider. JINST, vol. 3, pp. S08003, 2008.

[27] PERALVA, B. S. M.; ANDRADE FILHO, L. M. A.; CERQUEIRA, A. S. The TileCal Energy Reconstruction for Collision Data Using the Matched Filter. CERN Document Server, ATL-TILECAL-PROC-2013-023, Geneva, 2013.

[28] BECK, H. P. et al. The base-line data flow system of the atlas trigger and daq. IEEE Transactions on Nuclear Science, v. 51, n. 3, p. 470-475, June 2004. ISSN 0018-9499.

[29] AGUIAR, M. S.; RESENDE, M. O.; TEIXEIRA, T.; ANDRADE FILHO, L. M; SEIXAS, J. M. Implementação em FPGA de um Método Iterativo de Deconvolução para Operar no Sistema de Trigger do Experimento ATLAS. XXII Encontro Nacional de Modelagem Computacional e X Encontro de Ciências e Tecnologia de Materiais, Juiz de Fora, MG, 2018.

[30] AGUIAR, Melissa Santos; DE ANDRADE FILHO, Luciano Manhães; DE SEIXAS, José Manoel. Processamento Embarcado para Implementação de um Método Iterativo de Deconvolução Visando a Reconstrução de Energia em Calorímetros de Altas Energias. 2019.

[31] ATLAS, C. ATLAS liquid argon calorimeter: Technical design report. 1996. Disponível em: <https://inspirehep.net/record/432394/files/CERN-LHCC-96-41.pdf>.

[32] BAN, J. et al. Cold electronics for the liquid argon hadronic end-cap calorimeter of atlas. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Elsevier, v. 556, n. 1, p. 158-168, 2006.

[33] THE ATLAS HLT/DAQ/DCS GROUP. ATLAS High-Level Trigger, Data Acquisition and Controls. ATLAS Technical Report, TDR-016, CERN, 2001.

[34] WATTS, G. Review of Triggering Nuclear Science Symposium Conference Record, IEEE, v. 1, pp. 282-287, 2003.

[35] HADLEY, D. R. Digital Filtering Performance in the ATLAS Level-1 Calorimeter Trigger. Real Time Conference (RT), 7th IEEE-NPSS, pp. 1-6. IEEE, 2010.

[36] ARMSTRONG, S.; ANJOS, A.; BAINES, J. Implementation and Performance of the High Level Trigger Electron and Photon Selection for the ATLAS Experiment at the LHC. Nuclear Science Symposium Conference Record, IEEE, vol. 4, pp. 2038-2042, oct, 2004.

[37] SANCHEZ, C. A. S. Implementation of the ROD Crate DAQ Software for the ATLAS Tile Calorimeter and a Search for a MSSM Higgs Boson Decaying int Tau Pairs. PhD Thesis, Universitat de Valencia - CSIC, Valencia, Spain, 2010.

[38] RUGGIERO, F. LHC Accelerator R&D and Upgrade Scenarios. The European Physical Journal C - Particles and Fields, v. 34, pp. 433-442, 2004.

[39] CEPEDA, M.; GORI, S.; ILTEN, P. Report from Working Group 2: Higgs Physics at the HL-LHC and HE-LHC. CERN Yellow Rep. Monogr., v. 7, pp. 221-584. 364 p., 2018.

[40] HILUMI, P. “HL-LHC Industry: Industry Relations and Procurement Website for the HL-LHC project”, 2019.

[41] CLEMENT, C.; KLIMEK, P. Identification of pile-up using the quality factor of pulse shapes in the ATLAS tile calorimeter. In: IEEE. Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE. [S.l.], 2011. p. 1188–1193.

[42] POLUSHKIN, Vladimir. Nuclear Electronics: Superconducting Detectors and Processing Techniques. John Wiley & Sons, 2004.

[43] GILAT, A. Matlab com Aplicações em Engenharia Bookman Editora, 2009.

[44] DUARTE, J. P. B. S. Estudo de Técnicas de Deconvolução para Reconstrução de Energia Online no Calorímetro Hadrônico do ATLAS. Tese de Mestrado, UFJF, Juiz de Fora, MG, 2015.

[45] MEYER-BAESE, U. Digital Signal Processing with Field Programmable Gate Arrays. 3rd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2007. ISBN 3540726128, 9783540726128.

[46] BHANDARI, Sheetal et al. Methodology for on the fly partial reconfiguration for computation intensive applications on FPGA. In: 2010 International Conference on Computer Applications and Industrial Electronics. IEEE, 2010. p. 597-601.

[47] CYCLONE, I. Device Handbook. San Jose: ALTERA. Acesso em Agosto de 2021. Disponível em: <http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf>.

[48] STRATIX, V. Device Handbook. San Jose: ALTERA. Acesso em Agosto de 2021. Disponível em: <http://www.altera.com/literature/hb/stratix-v/stratix5 handbook.pdf>.

[49] VIRTEX-6. Family Overview. Acesso em Agosto de 2021. Disponível em: <http://www.xilinx.com/support/documentation/data sheets/ds150.pdf>.

[50] MITRA, S. K. Digital Signal Processing: A Computer Based Approach. [S.l.]: McGraw-Hill, 1998.

[51] SANTOS, V. A. M.; SILVA, L. R. M.; ANDRADE FILHO, L. M. Implementação de Circuitos Aritméticos em Ponto Flutuante Utilizando Formato com Número de bits Configurável. Congresso Brasileiro de Automática, pp. 1-2, João Pessoa, 2018.

[52] KAPISCH, E.B., SILVA, L.R.M., MARTINS, C.H.N., BARBOSA, A.S., DUQUE, C.A., ANDRADE FILHO, L.M., AND CERQUEIRA, A.S. (2014). An Electrical Signal Disturbance Detector and Compressor Based on FPGA Platform. 16th International Conference on Harmonics and Quality of Power (ICHQP), 278-282.

[53] KAPISCH, E.B., SILVA, L.R.M., CERQUEIRA, A.S., ANDRADE FILHO, L.M., DUQUE, C.A., AND RIBEIRO, P.F. (2016). A Gapless Waveform Recorder for Monitoring Smart Grids. 17th International Conference on Harmonics and Quality of Power (ICHQP), 130-136.

[54] MARTINS, C.H., MONTEIRO, H.L.M., M., O.M., SILVA, L.R.M., DUQUE, C.A., AND RIBEIRO, P.F. (2016). A Virtual Instrument for Time Varying Harmonic Analysis. IEEE Power and Energy Society General Meeting (PESGM), 1-5.

[55] OLIVEIRA, M.M., SILVA, L.R.M., DUQUE, C.A., ANDRADE FILHO, L.M., AND RIBEIRO, P.F. (2018). Implementation of an Electrical Signal Compression System Using Sparse Representation. 18th International Conference on Harmonics and Quality of Power (ICHQP), 1-5.

[56] TERASIC TECHNOLOGIES. Terasic Technologies FPGA Dev Kits for Altera Cyclone® II, III, & IV. Acesso em Agosto de 2021. Disponível em: <https://br.mouser.com/new/terasic-technologies/terasic-fpga-dev-cyclone-kits/>.

[57] RIBEIRO, P. F. et al. Power systems signal processing for smart grids. [S.l.]: John Wiley & Sons, 2013.

[58] AGUIAR, Melissa Santos et al. Arquitetura Multi-Core de Processadores Reconfiguráveis para Reconstrução Online de Energia no Calorímetro Hadrônico do ATLAS. In: Congresso Brasileiro de Automática-CBA. 2020.

[59] CASTELAN, J.P. Implementação de um Shift Register em VHDL. Acesso em Agosto de 2021. Disponível em: <https://www.embarcados.com.br/implementacao-de-um-shift-register-em-vhdl/>.

[60] DUARTE, J. P. B. S. Deconvolução Free-running aplicada à estimação Online de Energia na Calorimetria do ATLAS. Tese de Doutorado, UFJF, Juiz de Fora, MG, 2019.

[61] CHAPMAN, J. ATLAS simulation computing performance and pile-up simulation in ATLAS. LPCC, 2011.

[62] KAY, S. M. Fundamentals of statistical signal processing: Detection theory, vol. 1. [S.l.]: Prentice Hall Upper Saddle River, NJ, USA:, 1998.

[63] HAYKIN, S. Adaptive Filter Theory (3rd Ed.). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-322760-X.

[64] TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. Digital systems. 2007.

APÊNDICE A - ALGORITMOS IMPLEMENTADOS NO QUARTUS.

Neste apêndice são descritos os algoritmos *top level* que contém os passos seguidos, de forma simplificada, para cada implementação que foi realizada no QUARTUS, em linguagem *verilog*.

Algoritmo 1: Implementação por Máquina de Estados.

```
 1  module SM
 2  (
 3          input                           clk, rst, ready in,
 4          input          signed [9:0]     in,
 5          output   reg                    req_in,
 6          output   reg signed [15:0]  out,
 7          output   reg                    ena_out
 8  );
 9
10  reg [1:0] q;
11  reg [3:0] cnt;
12
13  reg signed [15:0] sr [14:0];
14  reg signed [15:0] w  [14:0];
15
16
17  initial begin
18
19  w[00] = -16'd2;
20  w[01] = 16'd5;
21  w[02] = -16'd12;
22  w[03] = 16'd25;
23  w[04] = -16'd47;
24  w[05] = 16'd87;
25  w[06] = -16'd158;
26  w[07] = 16'd269;
27  w[08] = -16'd155;
28  w[09] = 16'd63;
29  w[10] = -16'd29;
30  w[11] = 16'd15;
31  w[12] = -16'd7;
32  w[13] = 16'd3;
33  w[14] = -16'd1;
34  end
35
36  always @ (posedge clk)
37  begin
38          if (rst)
39                  q <= 2'd0;
40          else
41                  begin
42                          case (q)
```

```
43                      2'd0:   begin
44                                      ena_out <= 1'b0;
45                                      if (ready_in)
46                                              begin
47                                                      req_in <= 1'b1;
48                                                      q      <= 2'd1;
49                                              end
50                                      else
51                                              begin
52                                                      req_in <= 1'b0;
53                                                      q      <= q;
54                                              end
55                              end
56                      2'd1:   begin
57                                      req_in <= 1'b0;
58                                      sr[14] <= sr[13];
59                                      sr[13] <= sr[12];
60                                      sr[12] <= sr[11];
61                                      sr[11] <= sr[10];
62                                      sr[10] <= sr[09];
63                                      sr[09] <= sr[08];
64                                      sr[08] <= sr[07];
65                                      sr[07] <= sr[06];
66                                      sr[06] <= sr[05];
67                                      sr[05] <= sr[04];
68                                      sr[04] <= sr[03];
69                                      sr[03] <= sr[02];
70                                      sr[02] <= sr[01];
71                                      sr[01] <= sr[00];
72                                      sr[00] <= in;
73                                      q <= 2'd2;
74                                      out <= 16'd0;
75                                      cnt <= 4'd0;
76                              end
77                      2'd2: begin
78                                      out <= out + sr[cnt] * w[cnt];
79                                      cnt <= cnt + 4'd1;
80                                      if (cnt == 4'd14)
81                                              q <= 2'd3;
82                                      else
83
84                                              q <= q;
85
86                              end
87                      2'd3:   begin
88                                      ena_out <= 1'b1;
89                                      q <= 2'd0;
90                              end
91                      endcase
92              end
93  end
94  endmodule
```

Algoritmo 2: Implementação da forma direta de Filtros FIR.

```
 1  module fir_direct
 2  (
 3          input clk,
 4          input   signed   [9:0] x,
 5          output signed   [15:0] y
 6  );
 7  reg signed [9:0]   r   [14:0];
 8  reg signed [15:0] w  [14:0];
 9
10  initial begin
11  w[00] = -16'd2;
12  w[01] = 16'd5;
13  w[02] = -16'd12;
14  w[03] = 16'd25;
15  w[04] = -16'd47;
16  w[05] = 16'd87;
17  w[06] = -16'd158;
18  w[07] = 16'd269;
19  w[08] = -16'd155;
20  w[09] = 16'd63;
21  w[10] = -16'd29;
22  w[11] = 16'd15;
23  w[12] = -16'd7;
24  w[13] = 16'd3;
25  w[14] = -16'd1;
26  end
27
28  always @ (posedge clk) begin
29          r[0] <= x;
30          r[1] <= r[0];
31          r[2] <= r[1];
32          r[3] <= r[2];
33          r[4] <= r[3];
34          r[5] <= r[4];
35          r[6] <= r[5];
36          r[7] <= r[6];
37          r[8] <= r[7];
38          r[9] <= r[8];
39          r[10] <= r[9];
40          r[11] <= r[10];
41          r[12] <= r[11];
42          r[13] <= r[12];
43          r[14] <= r[13];
44  end
45
46  assign y = x*w[00]
47  + r[0]*w[01] + r[1]*w[02] + r[2]*w[03] + r[3]*w[04] + r[4]*w[05]
48  + r[5]*w[06]+ r[6]*w[07] + r[7]*w[08] + r[8]*w[09] + r[9]*w[10]
49  + r[10]*w[11] + r[11]*w[12]+ r[12]*w[13] + r[13]*w[14];
50  endmodule
```

Algoritmo 3: Implementação da forma transposta de Filtros FIR.

```
 1  module fir_transposed
 2  (
 3          input                clk,
 4          input   signed [9:0] x,
 5          output signed [15:0] y
 6  );
 7  reg signed [15:0] r [14:0];
 8  reg signed [15:0] w[14:0];
 9
10  initial begin
11
12  w[00] = -16'd2;
13  w[01] = 16'd5;
14  w[02] = -16'd12;
15  w[03] = 16'd25;
16  w[04] = -16'd47;
17  w[05] = 16'd87;
18  w[06] = -16'd158;
19  w[07] = 16'd269;
20  w[08] = -16'd155;
21  w[09] = 16'd63;
22  w[10] = -16'd29;
23  w[11] = 16'd15;
24  w[12] = -16'd7;
25  w[13] = 16'd3;
26  w[14] = -16'd1;
27  end
28
29  always @ (posedge clk) begin
30
31  r[0]<=x*w[14];
32  r[1]<=r[0]+x*w[13];
33  r[2]<=r[1]+x*w[12];
34  r[3]<=r[2]+x*w[11];
35  r[4]<=r[3]+x*w[10];
36  r[5]<=r[4]+x*w[09];
37  r[6]<=r[5]+x*w[08];
38  r[7]<=r[6]+x*w[07];
39  r[8]<=r[7]+x*w[06];
40  r[9]<=r[8]+x*w[05];
41  r[10]<=r[9]+x*w[04];
42  r[11]<=r[10]+x*w[03];
43  r[12]<=r[11]+x*w[02];
44  r[13]<=r[12]+x*w[01];
45  end
46  assign y = r[13]+x*w[00];
47
48  endmodule
```