

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Leandro da Graça Ribeiro

**Simulador de pulsos do TileCal em tempo real utilizando o processador
SAPHO e FPGA**

Juiz de Fora

2025

Leandro da Graça Ribeiro

**Simulador de pulsos do TileCal em tempo real utilizando o processador
SAPHO e FPGA**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área: Sistemas de Energia Elétrica.

Orientador: Prof. Dr. Luciano Manhães de Andrade Filho,

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

da Graça Ribeiro, Leandro.

Simulador de pulsos do TileCal em tempo real utilizando o processador
SAPHO e FPGA / Leandro da Graça Ribeiro. – 2025.

159 f. : il.

Orientador: Luciano Manhães de Andrade Filho

Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Faculdade
de Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, 2025.

1. Simulador de pulsos. 2. Alta taxa de eventos. 3. Calorímetro
Hadrônico de Telhas. I. Manhães de Andrade Filho, Luciano, orient. II.
Título.

Leandro da Graça Ribeiro

Simulador de Pulsos do TileCal em Tempo Real Utilizando o Processador SAPHO e FPGA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Sistemas de Energia Elétrica

Aprovada em 10 de setembro de 2025.

BANCA EXAMINADORA

Prof. Dr. Luciano Manhães de Andrade Filho - Orientador

Universidade Federal de Juiz de Fora

Prof. Dr. Rafael Antunes Nóbrega

Universidade Federal de Juiz de Fora

Prof. Dr. Herman Pessoa Lima Junior

Centro Brasileiro de Pesquisas Físicas

Juiz de Fora, 18/08/2025.



Documento assinado eletronicamente por **Luciano Manhaes de Andrade Filho, Professor(a)**, em 10/09/2025, às 18:31, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Herman Pessoa Lima Júnior, Usuário Externo**, em 11/09/2025, às 05:15, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Rafael Antunes Nobrega, Professor(a)**, em 13/09/2025, às 10:35, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **2558238** e o código CRC **B54166C9**.

Dedico este trabalho à minha família e amigos.
Agradeço por cada momento de alegria, incentivo e por
acreditarem em mim.

AGRADECIMENTOS

Gostaria, primeiramente, de agradecer a Deus, por me conceder saúde, força e perseverança para enfrentar cada etapa desta jornada. Sem Deus, sei que nada do que construí seria possível.

Agradeço à minha família, meu porto seguro e base de tudo que sou. Em especial, agradeço ao meu pai, Marco Antonio Ribeiro, e à minha mãe, Rosa Isméria Aparecida da Graça, pelo amor incondicional, pelos valores que me transmitiram e pelo apoio constante em todos os momentos da minha vida. Sei que, muitas vezes, colocaram meus sonhos e necessidades acima dos próprios interesses, fazendo sacrifícios que jamais poderei retribuir à altura. Ao meu irmão, Lucas da Graça Ribeiro, agradeço por ser um companheiro de vida e o melhor irmão que eu poderia pedir a Deus, sempre presente com apoio, amizade e incentivo.

À Mariana, minha namorada, agradeço por caminhar ao meu lado desde o início desta nova etapa, compartilhando alegrias e desafios. Sua paciência, compreensão e incentivo foram fundamentais para que eu mantivesse a motivação e a confiança, mesmo nos momentos mais difíceis. Sua presença tornou esta jornada mais leve e significativa, obrigado por sempre acreditar em mim.

Aos amigos que a universidade me presenteou, como o Ediward e o Guilherme, agradeço pelas conversas, pelas risadas, pelo companheirismo e pelo apoio em momentos difíceis. Foram vocês que tornaram essa caminhada, desde o começo da graduação, mais prazerosa, menos solitária e muito mais enriquecedora.

Ao CERN e as pessoas com quem convivo diariamente, em especial à Camila, ao Alam, ao Fábio e à Catarina, deixo minha gratidão por terem transformado o desafio de me adaptar a um novo país, com cultura e rotinas diferentes, em uma experiência repleta de bons momentos, trocas de conhecimento e amizade verdadeira.

Ao professor Luciano, meu orientador, agradeço por ter me recebido como orientando no meio do percurso do mestrado, confiando em meu trabalho e me oferecendo a oportunidade de atuar em uma área de pesquisa diretamente ligada a experimentos científicos reais, área que sempre admirei e acompanhei por meio de documentários, séries e filmes.

À Universidade Federal de Juiz de Fora (UFJF), agradeço pela estrutura, recursos e professores qualificados. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, que viabilizou a dedicação ao trabalho, e desempenha papel essencial para que milhares de estudantes e pesquisadores em todo o país contribuam para o avanço da ciência.

"What I cannot create, I do not understand."

Richard P. Feynman

RESUMO

O Grande Colisor de Hádrons (LHC) é o maior acelerador de partículas do mundo, operado pelo CERN, e investiga a estrutura fundamental da matéria por meio de colisões próton-próton em altíssimas energias. A cada ciclo de operação, seus detectores registram grandes volumes de dados. Entre 2026 e 2030, o LHC permanecerá desligado para as atualizações que antecedem a Fase 2, o *High Luminosity LHC* (HL-LHC), período em que não haverá medidas. Nesse contexto, simuladores capazes de operar em tempo real e reproduzir com alta fidelidade os sinais dos detectores tornam-se ferramentas estratégicas para manter o avanço das pesquisas, contribuindo para o desenvolvimento e validação de novas metodologias. Este trabalho apresenta o desenvolvimento de um simulador de pulsos do Tile Calorimeter (TileCal) do experimento ATLAS, implementado em *Field Programmable Gate Arrays* (FPGAs) a partir de código gerado pelo processador *soft-core* SAPHO. O objetivo é viabilizar, com baixa latência, a geração livre (*free-running*) de pulsos com efeito de *pile-up*, possibilitando testes sob a taxa de eventos do LHC (40 MHz) como se fossem dados reais. A metodologia integra: i) um gerador congruencial multiplicativo pseudoaleatório para definir a ocorrência de eventos; ii) endereçamento de tabela exponencial pré-computada para gerar amplitudes compatíveis com respostas físicas; e iii) modelagem do pulso por filtros digitais para emular o *pile-up*. O SAPHO compila um subconjunto de C em Verilog, alocando apenas o hardware necessário (ponto fixo ou flutuante), facilitando a exploração de arquiteturas distintas. Foram concebidos quatro simuladores: dois com processador inteiro (equações de filtragem separadas e combinadas) e dois com ponto flutuante (um monoprocessado e outro com cinco processadores em paralelo). Os resultados mostram ganhos expressivos, com redução do número de *clocks* por amostra de 5104 na arquitetura inteira, para 31 na versão paralela em ponto flutuante, ganho superior a duas ordens de grandeza em relação ao primeiro protótipo e próximo da viabilidade prática em FPGAs modernos.

Palavras-chave: FPGA, SAPHO, simulador de pulsos, calorímetro hadrônico de telhas, alta taxa de eventos, CERN.

ABSTRACT

The Large Hadron Collider (LHC) is the world's largest particle accelerator, operated by CERN, and investigates the fundamental structure of matter through high-energy proton-proton collisions. In each operation cycle, the detectors record large volumes of data. Between 2026 and 2030, the LHC will remain off for upgrades preceding Phase 2, the High Luminosity LHC (HL-LHC), during which no real data will be acquired. In this context, real-time simulators capable of accurately reproducing detector signals become strategic tools for maintaining research progress, contributing to the development and validation of new methodologies. This work presents the development of a pulse simulator for the Tile Calorimeter (TileCal) of the ATLAS experiment, implemented in Field Programmable Gate Arrays (FPGAs) from code generated by the soft-core SAPHO processor. The aim is to enable, with low latency, the free-running generation of pulses with pile-up effects, allowing tests at the LHC event rate (40 MHz) as if they were real data. The proposed methodology integrates: (i) a multiplicative congruential pseudo-random generator to define event occurrence; (ii) addressing of a precomputed exponential table to produce amplitudes consistent with physical responses; and (iii) pulse modeling using digital filters to emulate pile-up. SAPHO compiles a C subset into Verilog, allocating only the required hardware (fixed- or floating-point), thus facilitating the exploration of different architectures. Four simulators were designed: two with integer processors (separate and combined filtering equations) and two with floating-point processors (a single-core version and another with five parallel processors). The results show significant performance gains, with the number of clocks per sample reduced from 5104 in the integer architecture to 31 in the parallel floating-point version — an improvement of more than two orders of magnitude compared to the first prototype, approaching the practical feasibility in modern FPGAs.

Keywords: FPGA; SAPHO; pulse simulator; Tile Calorimeter; high event rate; CERN.

LISTA DE ILUSTRAÇÕES

Figura 1	– Partículas do Modelo Padrão	27
Figura 2	– Status de países vinculados ao CERN	37
Figura 3	– Visão do LHC	38
Figura 4	– Complexo de aceleradores do CERN	39
Figura 5	– Estrutura do ATLAS e seus componentes	42
Figura 6	– <i>Inner Detector (ID)</i>	43
Figura 7	– Localização dos Calorímetros no Experimento ATLAS . .	44
Figura 8	– Detector de Múons	45
Figura 9	– Visualização de um módulo típico do TileCal	49
Figura 10	– Formato do pulso do TileCal	50
Figura 11	– Exemplo do efeito de <i>pile-up</i>	55
Figura 12	– Diagrama de blocos do SAPHO.	59
Figura 13	– IDE SAPHO.	65
Figura 14	– Parametrização de Processador no SAPHO.	66
Figura 15	– Fluxograma do processo de criação de um projeto no SAPHO.	67
Figura 16	– Parametrização de Processador no SAPHO.	68
Figura 17	– Fluxograma base do Simulador.	73
Figura 18	– Amplitudes e valores na tabela exponencial.	88
Figura 19	– Fluxo geral de decisão do bloco exponencial.	90
Figura 20	– Circuito equivalente RLC do shaper implementado no FENICS.	92
Figura 21	– Resposta ao Impulso das 5 equações de filtro.	95
Figura 22	– Fluxograma do Simulador 1: execução de um ciclo para a geração de uma amostra.	106
Figura 23	– Fluxograma do Simulador 2 com ciclo contínuo de execução.	110
Figura 24	– Fluxograma do Simulador 3, detalhando os diferentes loops paralelos entre processador e gerador.	113
Figura 25	– Estrutura da Forma Direta II para filtros digitais	115
Figura 26	– Arquitetura geral do Simulador 4.	121
Figura 27	– Fluxograma de execução do Simulador 4, com divisão entre gerador pseudoaleatório, múltiplos processadores SAPHO e somador final.	122
Figura 28	– Saída do Simulador 1: forma de onda resultante com base em 800 amostras.	126
Figura 29	– Amostras geradas pelo Simulador 1 replicado em MATLAB.	127
Figura 30	– Estrutura RTL gerada para o Simulador 1, visualizada no ambiente Quartus.	127

Figura 31	– RTL Viewer do Simulador 2, com estrutura idêntica ao do Simulador 1.	129
Figura 32	– Saída temporal gerada pelo Simulador 2.	129
Figura 33	– Saída temporal gerada pelo Simulador 2.	130
Figura 34	– Estrutura RTL do Simulador 3, com separação entre gerador externo e processador SAPHO.	131
Figura 35	– Saída temporal do Simulador 3: forma de pulso gerada com arquitetura modular.	132
Figura 36	– Visualização RTL do Simulador 4: cinco núcleos SAPHO paralelos e somador no <i>top level</i>	134
Figura 37	– Recorte da simulação no ModelSim das amostras de saída.	135
Figura 38	– Amostras de saída do Simulador 4.	135
Figura 39	– Diferença entre HW (ganho 1) e MATLAB (ganho 1), 1000 amostras.	140
Figura 40	– Diferença entre HW (ganho 100) e MATLAB (ganho 100), 1000 amostras.	141
Figura 41	– Erro percentual em relação à variação de mantissa e expoente do processador.	142
Figura 42	– Erro percentual em função dos parâmetros do processador (versão ampliada).	143
Figura 43	– Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 23 e expoente = 8.	157
Figura 44	– Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 14 e expoente = 7.	158
Figura 45	– Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 14 e expoente = 6.	158
Figura 46	– Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 18 e expoente = 6.	159

LISTA DE TABELAS

Tabela 1 – Instruções e circuitos gerados automaticamente pelo Assembler, com suas respectivas ativações na ULA e modos de operação.	62
Tabela 2 – Funções, palavras-chave e operadores suportados pela linguagem C no SAPHO.	66
Tabela 3 – Comparação entre geradores verdadeiramente aleatórios e pseudoaleatórios.	77
Tabela 4 – Comparação geral entre métodos clássicos de PRNG (visão de alto nível).	80
Tabela 5 – Características do Simulador 1	103
Tabela 6 – Características do Simulador 2	108
Tabela 7 – Características do Simulador 3	114
Tabela 8 – Características da Implementação Final com Múltiplos Processadores SAPHO	118
Tabela 9 – Resumo comparativo entre os simuladores desenvolvidos	124
Tabela 10 – Desempenho do Simulador 1	126
Tabela 11 – Amostras inteiras produzidas em $1\mu s$ com $T_{clk} = 20$ ps (50 GHz). 136	
Tabela 12 – Métricas primárias de aderência entre HW e MATLAB (1000 amostras).	138
Tabela 13 – Métricas secundárias: viés, erros pontuais e medidas integrativas (1000 amostras).	139
Tabela 14 – Resumo de desempenho e arranjo lógico do Simulador 4.	144
Tabela 15 – Erro percentual para diferentes configurações de mantissa e expoente no Simulador 4 (1610 amostras, $1\mu s$ de simulação).	157

LISTA DE ABREVIATURAS E SIGLAS

ALICE	<i>A Large Ion Collider Experiment</i> – Experimento do LHC
ALU	<i>Arithmetic Logic Unit</i> – Unidade Lógica e Aritmética
ARM	<i>Advanced RISC Machines</i> – Arquitetura de processadores ARM
ASIC	<i>Application-Specific Integrated Circuit</i> – Circuito Integrado de Aplicação Específica
ATLAS	<i>A Toroidal LHC ApparatuS</i> – Detector do experimento no LHC
BRAM	<i>Block Random Access Memory</i> – Memória RAM em bloco
CERN	<i>Conseil Européen pour la Recherche Nucléaire</i> – Organização Europeia para a Pesquisa Nuclear
CMS	<i>Compact Muon Solenoid</i> – Experimento do LHC
CP	Carga-Paridade
DSP	<i>Digital Signal Processing</i> – Processamento Digital de Sinais
EC	Calorímetro Eletromagnético
EQM	Erro Quadrático Médio
FCC	<i>Future Circular Collider</i> – Colisor Circular Futuro
FPGA	<i>Field-Programmable Gate Array</i> – Arranjo de Portas Programáveis em Campo
GUTs	<i>Grand Unified Theories</i> – Teorias de Grande Unificação
HDL	<i>Hardware Description Language</i> – Linguagem de Descrição de Hardware
HC	Calorímetro Hadrônico
HL-LHC	<i>High-Luminosity Large Hadron Collider</i> – LHC de Alta Luminosidade
ID	<i>Inner Detector</i> – Detector Interno
IDE	<i>Integrated Development Environment</i> – Ambiente de Desenvolvimento Integrado
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
ILC	<i>International Linear Collider</i> – Colisor Linear Internacional

I/O	<i>Entrada/Saída – Input/Output</i>
LEP	<i>Large Electron–Positron Collider</i> – Grande Colisor Elétron–Pósitron
LHC	<i>Large Hadron Collider</i> – Grande Colisor de Hádrons
LHCb	<i>Large Hadron Collider beauty</i> – Experimento do LHC
Linac	<i>Linear Accelerator</i> – Acelerador Linear
LCG	<i>Linear Congruential Generator</i> – Gerador Congruencial Linear
LUT	<i>Look-Up Table</i> – Tabela de Consulta
MAE	<i>Mean Absolute Error</i> – Erro Absoluto Médio
MATLAB	<i>Matrix Laboratory</i> – Ambiente de programação e computação numérica
MLCG	<i>Multiplicative Linear Congruential Generator</i> – Gerador Congruencial Linear Multiplicativo
MET	<i>Missing Transverse Energy</i> – Momento Transversal Faltante
NaN	<i>Not a Number</i>
NRMSE	<i>Normalized Root Mean Square Error</i> – Erro Quadrático Médio Normalizado
PC	<i>Program Counter</i> – Contador de Programa
PMT	<i>Photomultiplier Tube</i> – Tubo Fotomultiplicador
PS	<i>Proton Synchrotron</i> – Síncrotron de Prótons
QCD	Cromodinâmica Quântica
RF	Radiofrequência
RISC	<i>Reduced Instruction Set Computer</i>
RMSE	<i>Root Mean Square Error</i> – Raiz do Erro Quadrático Médio
ROM	<i>Read-Only Memory</i> – Memória Somente de Leitura
SAPHO	<i>Scalable Architecture Processor for Hardware Optimization</i>
SCT	<i>SemiConductor Tracker</i> – Rastreador de Semicondutores
SoC	<i>System-on-Chip</i>
SPS	<i>Super Proton Synchrotron</i> – Síncrotron de Prótons

SUSY	Supersimetria
TileCal	<i>Tile Calorimeter</i> – Calorímetro de Telhas do ATLAS
TRT	<i>Transition Radiation Tracker</i> – Rastreador de Radiação de Transição
ULA	Unidade Lógica e Aritmética
VHDL	<i>VHSIC Hardware Description Language</i>

LISTA DE SÍMBOLOS

η	Pseudorrapidez (variável angular em detectores cilindro-azimutais).
$ \eta $	Módulo da pseudorrapidez (faixa de cobertura angular).
TeV	Tera-elétron-volt (unidade de energia de feixe/colisão).
MJ	Megajoule (energia macroscópica armazenada no feixe).
T	Tesla (unidade de indução/campo magnético).
MHz	Megahertz (unidade de frequência).
GHz	Gigahertz (unidade de frequência).
ns	Nanossegundo (unidade de tempo).
μs	Microssegundo (unidade de tempo).
ps	Picosegundo (unidade de tempo).
K	Kelvin (unidade de temperatura).
$SU(3)_C$	Grupo especial unitário associado à cromodinâmica quântica (QCD), responsável pela força forte.
$SU(2)_L$	Grupo especial unitário associado à interação fraca, atuando apenas sobre férmions de quiralidade esquerda.
$U(1)_Y$	Grupo unitário associado à hipercarga fraca, parte da formulação da interação eletrofraca.
u	Quark up, carga elétrica $+\frac{2}{3}$.
d	Quark down, carga elétrica $-\frac{1}{3}$.
c	Quark charm, carga elétrica $+\frac{2}{3}$.
s	Quark strange, carga elétrica $-\frac{1}{3}$.
t	Quark top, carga elétrica $+\frac{2}{3}$.
b	Quark bottom, carga elétrica $-\frac{1}{3}$.
e^-	Elétron, lépton de primeira geração, carga -1 .
μ^-	Múon, lépton de segunda geração, carga -1 .
τ^-	Tau, lépton de terceira geração, carga -1 .
ν_e	Neutrino eletrônico, associado ao elétron, neutro.
ν_μ	Neutrino muônico, associado ao múon, neutro.
ν_τ	Neutrino tauônico, associado ao tau, neutro.
γ	Fóton: bóson mediador da interação eletromagnética, sem massa e de alcance infinito.
g	Glúons: oito bósons mediadores da interação forte, responsáveis pelo confinamento dos quarks.
W^\pm	Bósons W: mediadores da interação fraca, massivos ($\sim 80 \text{ GeV}/c^2$).
Z^0	Bóson Z: mediador neutro da interação fraca, massivo ($\sim 91 \text{ GeV}/c^2$).
H^0	Bóson de Higgs: partícula escalar associada ao mecanismo de Higgs, responsável pela geração de massa das partículas elementares.
$x[n]$	Amostra de entrada no instante n .
$y[n]$	Saída total do simulador (após a soma das vias).

$y_i[n]$	Saída do i -ésimo filtro/ramo (via i).
$r_k\{z[n]\}$	Operador de atraso: $z[n - k]$ (atraso de k amostras).
r_{x_k}, r_{y_k}	Notação abreviada para atraso de k amostras de x e de y (p.ex., $r_{x_1} = x[n - 1]$, $r_{y_2} = y[n - 2]$).
$H_i(s)$	Função de transferência analógica do i -ésimo filtro (domínio de Laplace).
$H_6(s)$	Função de transferência analógica equivalente após combinação das vias.
$H(z)$	Função de transferência no domínio- z (discreto).
s	Variável complexa do domínio de Laplace.
z	Variável complexa do domínio- z .
T	Período de amostragem (intervalo entre amostras).
a_i, b_i	Coefficientes (denominador e numerador) dos filtros IIR.
$a_i^{(j)}, b_i^{(j)}$	Coefficientes do j -ésimo biquadro/seção de 2 ^a ordem.
y^{HW}	Saída medida no hardware (FPGA) na amostra n .
y^{MAT}	Saída de referência (MATLAB).
$e[n]$	Resíduo ponto a ponto: $e[n] = y^{\text{HW}}[n] - y^{\text{MAT}}[n]$.
N	Número de amostras consideradas na janela de comparação.
\bar{z}	Média amostral da sequência z .
$\text{Var}(z)$	Variância amostral da sequência z .
σ	Desvio-padrão amostral ($\sigma = \sqrt{\text{Var}(\cdot)}$).
ρ	Correlação de Pearson entre duas séries ($-1 \leq \rho \leq 1$).
R^2	Coefficiente de determinação: fração da variância explicada.
$ e _{\text{max}}$	Erro absoluto máximo.
Δp	Diferença de pico (absoluta ou relativa) entre HW e referência.
Δt_p	Diferença (em amostras) entre as posições de pico.
ΔA	Diferença relativa de área (soma discreta ao longo da janela).
F_{sim}	Frequência mínima de operação da FPGA para acompanhar a taxa de eventos.
N_{clocks}	Número de ciclos de clock por amostra gerada.
F_{eventos}	Taxa de eventos a ser acompanhada (p.ex., 40 MHz no LHC).
T_{clk}	Período de clock da FPGA.
T_{amostra}	Tempo por amostra: $T_{\text{amostra}} = N_{\text{clocks}} T_{\text{clk}}$.
f_{LHC}	Frequência de cruzamentos de feixe do LHC (40 MHz).
G	Ganho aplicado às amostras (escala multiplicativa).
rnd	Registrador/estado do gerador pseudoaleatório (32 bits).
a, c	Parâmetros multiplicativo e aditivo do gerador congruencial.
rnd_idx	Índice de acesso à tabela exponencial.
ccc	Limiar/condição de ocupação (aceitação de evento).
Tab_exp	Tabela exponencial (mapa de amplitudes).
exp_out	Amplitude lida da tabela exponencial.
rand_out	Valor aplicado como entrada comum às cinco vias (após ocupação & tabela).
out_en1	Sinal de habilitação para sincronizar a soma final das vias.

SUMÁRIO

1	INTRODUÇÃO	19
1.1	MOTIVAÇÃO	20
1.2	OBJETIVOS	22
1.3	ESTRUTURA DA DISSERTAÇÃO	23
2	FUNDAMENTAÇÃO TEÓRICA	25
2.1	FÍSICA DE PARTICULAS	25
2.1.1	Contexto Histórico	25
2.1.2	O Modelo Padrão da Física de Partículas	26
2.1.2.1	<i>Férmions: Os Constituintes da Matéria</i>	26
2.1.2.2	<i>Bósons: Mediadores das Interações</i>	28
2.1.2.3	<i>Quebra de Simetria e Mecanismo de Higgs</i>	29
2.1.3	Limitações do Modelo Padrão	29
2.1.4	Além do Modelo Padrão	29
2.1.5	Experimentos em Física de Partículas	30
2.1.5.1	<i>Aceleradores e detectores</i>	30
2.1.5.2	<i>Outros aceleradores históricos</i>	31
2.1.6	Física de Neutrinos	31
2.1.7	Cosmologia e Física de Partículas	31
2.1.8	Futuro da Física de Partículas	32
2.1.9	Considerações Finais	32
2.2	ACELERADORES DE PARTÍCULAS	32
2.2.1	Princípios de Funcionamento	33
2.2.2	Tipos de Acionamento	34
2.2.2.1	<i>Acionamento Eletrostático</i>	34
2.2.2.2	<i>Acionamento por Radiofrequência</i>	34
2.2.2.3	<i>Acionamento por Plasma</i>	34
2.2.3	Importância e Futuro	35
2.3	O CERN E O LHC	35
2.3.1	Colaboração Internacional e Inovação Tecnológica	40
2.3.2	O Futuro do CERN	40
2.4	O EXPERIMENTO ATLAS	41
2.5	CALORÍMETRO HADRÔNICO DE TELHAS (TILECAL)	45
2.5.1	Arquitetura e Funcionamento	47
2.5.2	Sistema de Calibração e Monitoramento	49
2.5.3	Sistema de Trigger e Fluxo de Dados	51

2.5.4	Pulso de Saída e Desafios do <i>Pile-up</i>	53
2.5.5	Considerações Finais sobre o TileCal	56
3	FERRAMENTAS DE SIMULAÇÃO E DESENVOLVIMENTO	58
3.1	SAPHO	58
3.1.1	Arquitetura de Hardware do Processador	58
3.1.2	Unidade Lógica-Aritmética e Recursos Dinâmicos	61
3.1.3	Organização de Memória e Controle	62
3.1.4	Arquitetura de Pipeline	63
3.1.5	Representação de Ponto Flutuante Customizada	64
3.1.6	Ferramentas de Desenvolvimento	65
3.2	TECNOLOGIA FPGA	68
3.2.1	Vantagens para Aplicações em Tempo Real e Sistemas Reconfiguráveis	70
4	METODOLOGIA	72
4.1	GERADOR DE NÚMEROS ALEATÓRIOS	73
4.1.1	Introdução da geração computacional de números aleatórios	74
4.1.2	Geração Física vs. Pseudoaleatória	75
4.1.2.1	<i>Fontes de aleatoriedade física</i>	76
4.1.2.2	<i>Geradores pseudoaleatórios</i>	76
4.1.2.3	<i>Vantagens e limitações</i>	77
4.1.3	Métodos Clássicos de Geração Pseudoaleatória	77
4.1.3.1	<i>Geradores Congruenciais Lineares (LCG e MLCG)</i>	78
4.1.3.2	<i>Xorshift</i>	78
4.1.3.3	<i>Mersenne Twister (MT19937)</i>	79
4.1.3.4	<i>Geradores baseados em LFSR</i>	79
4.1.3.5	<i>Comparação e implicações práticas em hardware</i>	79
4.1.4	Justificativa do Método Adotado neste Trabalho	80
4.1.5	Implementação no Simulador do TileCal	82
4.1.5.1	<i>Impacto na fidelidade das simulações</i>	83
4.2	TABELA EXPONENCIAL	84
4.2.1	Formulação da Distribuição Exponencial Contínua	85
4.2.2	Discretização da Distribuição	86
4.2.2.1	<i>Aproximação por intervalos discretos</i>	87
4.2.2.2	<i>Construção da tabela exponencial</i>	87
4.2.2.3	<i>Vantagens da discretização</i>	87
4.2.3	Taxa de Ocupação e Fluxo Geral do Bloco Exponencial	88
4.3	MODELAGEM DE PULSOS	91
4.3.1	Circuito equivalente do Shaper (FENICS)	92
4.3.1.1	<i>Função de transferência global do shaper</i>	93

4.3.2	Decomposição em filtros parciais	93
4.3.3	Discretização dos filtros	95
4.3.4	Equações de diferenças e implementação no simulador	98
5	SIMULADORES	101
5.1	SIMULADOR 1	102
5.1.1	Arquitetura Geral	103
5.2	SIMULADOR 2	107
5.3	SIMULADOR 3	111
5.4	SIMULADOR 4	116
5.4.1	Equações de Filtragem	118
5.4.2	Arquitetura Geral	120
6	RESULTADOS	124
6.1	SIMULADOR 1	125
6.2	SIMULADOR 2	128
6.3	SIMULADOR 3	131
6.4	SIMULADOR 4	133
7	CONSIDERAÇÕES FINAIS	146
	REFERÊNCIAS	149
	APÊNDICE A – PRODUÇÃO CIENTÍFICA RESULTANTE	
	DA PESQUISA	156
	APÊNDICE B – Tabelas e Gráficos dos Simuladores	157

1 INTRODUÇÃO

A exploração científica das leis fundamentais do universo é uma jornada que atravessa milênios de investigação, iniciando-se com os primeiros questionamentos filosóficos na Grécia Antiga e desenvolvendo-se continuamente até as sofisticadas teorias científicas da contemporaneidade. Desde os pré-socráticos, que buscavam compreender a constituição da natureza por meio de elementos como água, fogo e ar, passando pela revolução copernicana, pelas leis de Newton e pela formulação da mecânica quântica, a humanidade tem ampliado de maneira progressiva sua visão sobre a constituição da matéria, da energia e do espaço-tempo. Essa busca incessante pela compreensão da realidade culminou na consolidação de áreas interdisciplinares como a física de partículas, a cosmologia e a engenharia de sistemas complexos, cujo papel é essencial para viabilizar os experimentos mais avançados da ciência moderna.

O avanço das tecnologias de observação e manipulação de sistemas físicos em escala microscópica permitiu que a ciência experimental atingisse níveis sem precedentes de precisão e sofisticação. Detectores capazes de registrar interações subatômicas com altíssima resolução, algoritmos inteligentes aplicados ao tratamento de grandes volumes de dados e estruturas computacionais altamente paralelizadas são exemplos de recursos desenvolvidos nas últimas décadas. Tais tecnologias não apenas aumentam a capacidade de explorar as fronteiras do conhecimento, mas também fomentam novas descobertas e consolidam paradigmas científicos anteriormente apenas teóricos.

Paralelamente, a engenharia, enquanto disciplina aplicada, tornou-se um elemento indissociável do progresso científico. A construção de aceleradores de partículas, como o Grande Colisor de Hádrons (LHC), exige conhecimentos avançados em eletromagnetismo, criogenia, ciência dos materiais, computação embarcada e controle automatizado. O sucesso de projetos dessa magnitude depende não apenas da formulação teórica precisa das interações físicas a serem testadas, mas também da habilidade técnica de projetar, fabricar e operar dispositivos em condições extremas de temperatura, radiação e sincronização temporal.

A interseção entre física e engenharia transcende a simples implementação de dispositivos experimentais, representando uma convergência profunda entre diferentes formas de produzir conhecimento e resolver problemas complexos. A engenharia, especialmente eletrônica e computacional, atua como elemento central no desenvolvimento de soluções que tornam viáveis os experimentos de fronteira da física contemporânea. Em vez de abordagens isoladas, os grandes projetos científicos modernos demandam colaborações interdisciplinares nas quais físicos, engenheiros, matemáticos e especialistas em ciência da computação trabalham em sinergia. Um exemplo emblemático dessa filosofia é o Centro Europeu de Pesquisas Nucleares (CERN), que integra profissionais de mais de cem países

com o objetivo comum de desvendar os princípios fundamentais do universo por meio de experimentos em larga escala.

Essa colaboração tem se mostrado indispensável diante da crescente complexidade dos experimentos em física de altas energias. À medida que os aceleradores de partículas, como o LHC, operam com luminosidades cada vez maiores — isto é, com taxas elevadas de colisões por segundo —, o número de eventos físicos gerados e registrados cresce de forma exponencial. Cada colisão pode produzir milhares de partículas secundárias, cujas trajetórias e energias devem ser detectadas e analisadas com extrema precisão. O volume de dados gerado por esses eventos, aliado à necessidade de processamento em tempo real, impõe desafios tecnológicos consideráveis. Tais demandas impulsionam o desenvolvimento de arquiteturas eletrônicas avançadas e soluções computacionais de alta performance, consolidando o papel central da engenharia eletrônica e da computação embarcada na pesquisa científica de ponta.

Além disso, muitas das soluções tecnológicas concebidas inicialmente para a física de altas energias acabam por influenciar outros setores estratégicos da sociedade. Tecnologias como imageamento por ressonância magnética, algoritmos de aprendizado de máquina para diagnóstico médico, sistemas de radar, comunicações seguras e redes embarcadas são exemplos de inovações cujas origens estão vinculadas a demandas específicas de grandes experimentos científicos. Dessa forma, a fronteira entre ciência básica e aplicação tecnológica torna-se cada vez mais fluida, promovendo um ciclo contínuo de retroalimentação entre teoria, experimento e inovação.

Neste cenário, a presente dissertação se insere como uma contribuição à fronteira entre a pesquisa fundamental e a engenharia aplicada, abordando desafios que emergem da necessidade de melhorar continuamente a eficiência, a precisão e a confiabilidade dos sistemas utilizados em experimentos de grande porte. Embora os próximos capítulos desta dissertação apresentem com profundidade os aspectos técnicos específicos do problema investigado, esta seção teve como propósito situar o leitor no contexto científico, histórico e tecnológico em que se insere a proposta desenvolvida.

1.1 MOTIVAÇÃO

Nas últimas décadas, a física de partículas tem avançado significativamente na compreensão das interações fundamentais da natureza, graças ao desenvolvimento de experimentos de grande escala como os conduzidos no CERN. O LHC representa uma das maiores conquistas da engenharia moderna, sendo o maior e mais poderoso acelerador de partículas já construído. Seu funcionamento permite colidir prótons a altíssimas energias, recriando, em escala microscópica, condições semelhantes às do universo primordial. Desde o início de sua operação, em 2008, o LHC tem sido responsável por marcos fundamentais na física, como a descoberta do bóson de Higgs em 2012 pelos experimentos ATLAS e

CMS (ATLAS COLLABORATION, 2008).

O sucesso desse acelerador, no entanto, impõe novos desafios. O aumento contínuo da luminosidade dos feixes, que corresponde à taxa de colisões por segundo, exige maior desempenho dos sistemas eletrônicos e computacionais envolvidos nos experimentos. Estima-se que a taxa de interações nas futuras rodadas de operação atinja valores da ordem de bilhões de eventos por segundo, com condições de *pile-up* que superam 200 interações simultâneas por cruzamento de feixe (APOLLINARI *et al.*, 2015). Esse cenário gera não apenas uma explosão no volume de dados a ser processado, mas também impõe rigorosas exigências de latência, sincronização e filtragem em tempo real nos sistemas de leitura e aquisição dos detectores.

Dentro desse contexto, o experimento ATLAS se destaca como um dos mais ambiciosos e completos instrumentos de detecção do LHC. Composto por múltiplos subsistemas especializados, o ATLAS tem como objetivo registrar e analisar os produtos das colisões com máxima precisão. Dentre esses subsistemas, o *Tile Calorimeter* (TileCal) é responsável por medir a energia de hádrons e desempenha papel crucial na reconstrução de eventos físicos. Seu funcionamento baseia-se na conversão da luz gerada por cintiladores em sinais elétricos por meio de tubos fotomultiplicadores (*Photomultiplier Tubes* – PMTs), cujos pulsos contêm informações essenciais sobre a energia e o tempo de chegada das partículas (AGUIAR, 2023).

Para garantir a qualidade dos dados adquiridos, é fundamental que os sistemas de leitura do TileCal sejam submetidos a procedimentos contínuos de calibração, validação e testes sob condições controladas. A simulação precisa dos pulsos gerados pelo detector — com controle sobre parâmetros como tempo de subida, decaimento e intensidade — permite validar algoritmos de reconstrução, analisar a resposta dos sistemas eletrônicos e identificar falhas ou desvios operacionais. Isso se torna especialmente relevante frente às exigências impostas pelas novas taxas de eventos, que pressionam os limites de resposta em tempo real dos sistemas embarcados.

Dessa forma, a motivação deste trabalho nasce da necessidade concreta de aprimorar as ferramentas de teste e simulação dos sinais do TileCal. O objetivo não é apenas validar os sistemas diante de um cenário extremo, como o previsto no *High-Luminosity LHC*, mas também construir uma base robusta e flexível de emulação eletrônica que permita o desenvolvimento de soluções mais precisas e confiáveis para a física de altas energias. Tais avanços impactam diretamente a estabilidade operacional do experimento ATLAS e contribuem para o sucesso contínuo das futuras campanhas de coleta e análise de dados no CERN.

1.2 OBJETIVOS

Esta dissertação tem como objetivo principal o desenvolvimento de uma ferramenta de emulação de pulsos hadrônicos do calorímetro TileCal, por meio da implementação do processador SAPHO em um dispositivo do tipo *Field-Programmable Gate Array* (FPGA). A proposta consiste em projetar um sistema capaz de operar de forma autônoma em modo *free-running*, simulando sinais eletrônicos compatíveis com os pulsos reais observados nas células do TileCal, considerando parâmetros como forma de onda, taxa de amostragem, decaimento, empilhamento e amplitude.

Para alcançar esse propósito, estabelecem-se os seguintes objetivos específicos:

- Modelar matematicamente os pulsos gerados no TileCal, com base em dados reais e propriedades físicas do detector, visando representar adequadamente seus parâmetros característicos;
- Implementar a lógica do processador SAPHO em FPGA, utilizando linguagens de descrição de hardware adequadas (como Verilog ou VHDL), assegurando sua operação em tempo real com baixa latência e elevada estabilidade;
- Projetar uma arquitetura que permita a geração contínua de pulsos sem depender de sinais de *trigger* externos, viabilizando testes de longa duração e avaliações sob diferentes condições de taxa de eventos;
- Integrar o simulador a sistemas de aquisição e leitura de dados, permitindo sua utilização em bancadas de teste do TileCal para validação e calibração de algoritmos de reconstrução de energia;
- Validar experimentalmente o sistema desenvolvido, comparando os sinais gerados pelos simuladores em hardware, com valores esperados gerados pelos respectivos simuladores reproduzidos em software, através de medidas estatísticas, como por exemplo, o erro percentual ($E(\%)$).

O uso de FPGAs justifica-se pela necessidade de execução determinística de tarefas de processamento em tempo real, algo fundamental nos ambientes de aquisição de dados em experimentos de física de altas energias, onde a latência mínima e a reconfigurabilidade são requisitos técnicos indispensáveis. Além disso, a arquitetura paralela dos FPGAs permite a geração precisa de sinais físicos com alta frequência de repetição, possibilitando o emulador operar em condições próximas às reais enfrentadas no experimento ATLAS.

Dessa forma, o trabalho visa não apenas contribuir com a infraestrutura de testes e calibração do TileCal, mas também propor uma solução escalável e flexível para desafios semelhantes em outros subsistemas eletrônicos de detectores modernos.

1.3 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada de forma a conduzir o leitor do contexto científico ao desenvolvimento, validação e análise do emulador proposto.

- **Capítulo 1 — Introdução:** apresenta o pano de fundo histórico-científico, a motivação do trabalho, os objetivos geral e específicos, e delinea a contribuição pretendida no contexto do ATLAS/TileCal e do HL-LHC.
- **Capítulo 2 — Fundamentação Teórica:** reúne os conceitos necessários sobre física de partículas (Modelo Padrão e além), aceleradores, o CERN e o LHC, o experimento ATLAS e, em particular, o Tile Calorimeter. Discute ainda a formação do pulso eletrônico, a janela de amostragem e os desafios de *pile-up* em altas luminosidades.
- **Capítulo 3 — Ferramentas de Simulação e Desenvolvimento:** descreve o ecossistema utilizado no trabalho: o processador *soft-core* SAPHO (arquitetura, ULA, representação numérica e *toolflow*), e a tecnologia FPGA enquanto plataforma-alvo, destacando vantagens para processamento determinístico e em tempo real.
- **Capítulo 4 — Metodologia:** detalha a modelagem adotada para geração dos pulsos hadrônicos e sua execução em hardware: gerador pseudoaleatório (LCG) e controle de ocupação, tabela exponencial para amplitudes, discretização via transformada bilinear e implementação dos filtros (primeira e segunda ordem), além do fluxo geral do simulador.
- **Capítulo 5 — Simuladores:** apresenta três arquiteturas funcionais do emulador. O *Simulador 1* valida o encadeamento completo com filtros separados; o *Simulador 2* compacta a resposta em uma equação unificada; o *Simulador 3* explora ponto flutuante e integração com módulos externos em Verilog, discutindo implicações de desempenho, latência e uso de recursos; o *Simulador 4*, por fim, busca desenvolver o paralelismo das equações de filtragem do shaper, de forma síncrona, com o objetivo de reduzir ainda mais o número de ciclos por amostra.
- **Capítulo 6 — Resultados e Discussão:** consolida a validação qualitativa e quantitativa dos sinais gerados, incluindo métricas de forma de onda (EQM, correlação, resposta temporal), análise de ocupação e *pile-up*, além de síntese/uso de área e estimativas de *throughput* em FPGA, comparando as variantes do simulador.
- **Capítulo 7 — Conclusões e Perspectivas:** sintetiza as contribuições, limitações e lições do desenvolvimento, e sugere desdobramentos, como integração em bancadas do TileCal, variações de taxa de eventos, calibração fina de coeficientes, mitigação de *pile-up* em tempo real e extensão a outros subsistemas.

Ao final, apresentam-se as **Referências** bibliográficas e **Apêndices**, nos quais são disponibilizados a produção científica resultante da pesquisa, tabela com valores de erros percentuais para o simulador 4 e gráficos da diferença comparativa com o MATLAB para variações de mantissa e expoente do processador.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 FÍSICA DE PARTICULAS

A física de partículas, ou física de altas energias, constitui uma das áreas mais fundamentais da física moderna, preocupando-se com a compreensão das menores unidades constituintes da matéria e das forças fundamentais que as regem. Seu objetivo é descrever os blocos elementares do universo e como eles interagem para formar toda a estrutura conhecida da natureza, desde os átomos que compõem a matéria visível até os fenômenos mais extremos no cosmos.

Esta área da ciência se apoia fortemente em experimentos conduzidos em aceleradores de partículas, observações cosmológicas e modelos matemáticos altamente sofisticados. Através desses estudos, a física de partículas tem revelado uma realidade invisível ao olho humano, onde entidades subatômicas como quarks, glúons e léptons desempenham papéis fundamentais.

Além de suas contribuições teóricas, os avanços tecnológicos desenvolvidos para a física de partículas têm aplicações vastas em outras áreas do conhecimento, como a medicina (com tomografia por emissão de pósitrons, PET), a computação (como o desenvolvimento da *World Wide Web* no CERN) e a indústria de semicondutores (WAMBERSIE; GAHBAUER, 1996). Por isso, sua relevância transcende a fronteira da pesquisa pura, influenciando direta e indiretamente a sociedade moderna.

2.1.1 Contexto Histórico

A busca por entender a constituição da matéria acompanha o ser humano há milênios. Na Grécia Antiga, filósofos como Leucipo e Demócrito já postulavam a existência de unidades indivisíveis, os “átomos” (BRITANNICA, 2020). Embora rudimentar, essa visão filosófica é notável por antecipar ideias que só seriam testadas cientificamente dois milênios depois.

No século XIX, a teoria atômica de Dalton trouxe rigor empírico à ideia do átomo como unidade básica da matéria (HARTLEY, 1967). Posteriormente, a descoberta do elétron por J.J. Thomson em 1897 revelou que o átomo possuía estrutura interna (PAIS, 1986). Poucos anos depois, Rutherford, com seu famoso experimento de espalhamento de partículas alfa, revelou que os átomos possuíam um núcleo denso e carregado positivamente. Em 1932, James Chadwick completou o trio de constituintes nucleares ao descobrir o nêutron.

A descoberta do nêutron consolidou o modelo do átomo como uma estrutura nuclear complexa, formada por prótons e nêutrons cercados por elétrons. No entanto, na década de 1930, a descoberta do múon uma espécie de "elétron pesado", iniciou uma explosão

na “zoologia de partículas”. Durante os anos seguintes, foram descobertos píons, káons, hipérons e outras partículas. A física de partículas tornou-se então um campo ativo, exigindo a construção de aceleradores e detectores cada vez mais sofisticados.

Essa complexidade levou os físicos a buscar uma estrutura mais unificada, que pudesse explicar a grande quantidade de partículas observadas em experimentos (HALZEN; MARTIN, 1984). Assim surgiu o **Modelo Padrão**, uma teoria que descreve as partículas elementares e três das quatro forças fundamentais conhecidas: forte, fraca e eletromagnética.

2.1.2 O Modelo Padrão da Física de Partículas

A busca por um modelo que pudesse unificar e explicar essas descobertas levou ao desenvolvimento do Modelo Padrão da física de partículas, uma teoria que descreve com precisão as partículas fundamentais e, assim como mencionado anteriormente, três das quatro forças fundamentais conhecidas: a força eletromagnética, a força nuclear forte e a força nuclear fraca. A gravidade permanece fora dessa descrição quântica, sendo abordada pela relatividade geral (THOMSON, 2013).

Matematicamente, o Modelo Padrão é descrito por uma teoria quântica de campos baseada na simetria:

$$SU(3)_C \times SU(2)_L \times U(1)_Y \quad (2.1)$$

onde:

- $SU(3)_C$ representa a simetria da cromodinâmica quântica (QCD), responsável pela força forte (GRIFFITHS, 2008);
- $SU(2)_L \times U(1)_Y$ é a simetria da interação eletrofraca, que unifica as interações fraca e eletromagnética (GRIFFITHS, 2008).

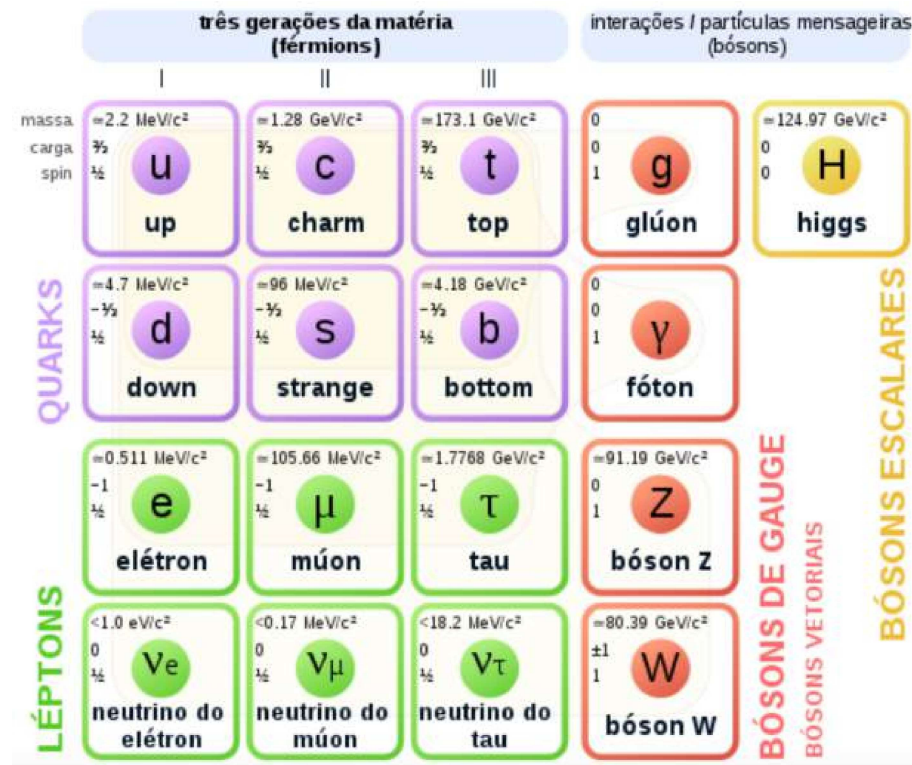
Esse formalismo permite descrever de maneira coesa as interações entre partículas, com o uso de bósons de gauge e campos de férmions.

As partículas fundamentais são divididas em duas grandes categorias: férmions e bósons. Os férmions são responsáveis por constituir a matéria ordinária, já os bósons são mediadores das forças fundamentais (HALZEN; MARTIN, 1984). A Figura 1 é uma ilustração das partículas fundamentais do Modelo Padrão.

2.1.2.1 Férmions: Os Constituintes da Matéria

Os férmions são o grupo de partículas que compõem toda a matéria observável no universo. Caracterizam-se por possuírem spin semi-inteiro ($\frac{1}{2}$) e obedecerem ao Princípio

Figura 1 – Partículas do Modelo Padrão



Fonte: (WIKIMEDIA, 2025)

de Exclusão de Pauli, segundo o qual dois férmions não podem ocupar simultaneamente o mesmo estado quântico. Essa propriedade é crucial para explicar a formação de camadas eletrônicas em átomos, a estabilidade de moléculas e as propriedades macroscópicas dos sólidos e líquidos (GRIFFITHS, 2008).

No Modelo Padrão, os férmions dividem-se em duas subclasses fundamentais: quarks e léptons. Cada uma está organizada em três gerações, as quais diferem principalmente pela massa e tempo de vida das partículas. As partículas da primeira geração são as mais leves e estáveis, formando a maior parte da matéria comum, enquanto as gerações superiores aparecem apenas em condições de energia extremamente elevadas, como em colisões de aceleradores ou em processos astrofísicos de alta energia.

- **Quarks:** Existem seis “sabores” de quarks — up (u), down (d), charm (c), strange (s), top (t) e bottom (b). Cada quark transporta uma carga elétrica fracionária ($+\frac{2}{3}$ ou $-\frac{1}{3}$) e interage pela força forte, mediada pelos glúons, além de participarem das interações eletromagnética e fraca. Devido ao fenômeno de confinamento da Cromodinâmica Quântica, quarks nunca são observados isoladamente, mas sim agrupados em hádrons: mésons (par quark-antiquark) e bárions (três quarks), como os prótons e nêutrons (PERKINS, 2000).
- **Léptons:** Incluem o elétron (e^-), o múon (μ^-), o tau (τ^-) e seus neutrinos associados

$(\nu_e, \nu_\mu, \nu_\tau)$. Os léptons carregados interagem pelas forças eletromagnética e fraca, enquanto os neutrinos, sendo eletricamente neutros e de massa muito pequena, interagem apenas pela força fraca, tornando sua detecção extremamente desafiadora. Os léptons das gerações superiores decaem rapidamente em partículas mais leves, geralmente produzindo um elétron ou múon e um par de neutrinos (GRIFFITHS, 2008).

Cada geração sucessiva de férmions é substancialmente mais pesada e instável do que a anterior, decaindo rapidamente em partículas de gerações inferiores. Essa hierarquia de massas e instabilidades sustenta inúmeros processos em física de altas energias e astrofísica de partículas.

2.1.2.2 *Bósons: Mediadores das Interações*

Os bósons são partículas de spin inteiro (tipicamente spin 1) que atuam como portadores das forças fundamentais no Modelo Padrão. Diferentemente dos férmions, eles podem ocupar o mesmo estado quântico simultaneamente, facilitando a mediação de interações entre as partículas de matéria.

- **Fóton (γ)**: Quantum da radiação eletromagnética, sem massa e com alcance infinito, responsável pelas interações entre partículas carregadas e por fenômenos como a luz visível e as ondas de rádio.
- **Glúons (g)**: Oito tipos de bósons que transportam a “carga de cor” na Cromodinâmica Quântica. São responsáveis pela força forte, que confina os quarks dentro dos hádrons. O acoplamento dos glúons aumenta à medida que a distância entre quarks diminui, explicando o confinamento e a liberdade assintótica (HALZEN; MARTIN, 1984).
- **Bósons W^\pm e Z^0** : Mediadores da interação fraca, possuem massas elevadas ($\sim 80\text{--}90 \text{ GeV}/c^2$) (OLIVE *et al.*, 2016), o que restringe o alcance dessa força a escalas subatômicas muito curtas ($\sim 10^{-18}\text{m}$). São fundamentais em processos de decaimento beta e nas oscilações de neutrinos.
- **Bóson de Higgs (H^0)**: Proposto em 1964 e detectado em 2012 pelos experimentos ATLAS e CMS no LHC, esse bóson escalar é associado ao mecanismo de Higgs, que confere massa a outras partículas elementares por meio da quebra espontânea da simetria eletrofraca (ATLAS COLLABORATION, 2012).

A confirmação experimental do bóson de Higgs representou o último elemento necessário para validar o Modelo Padrão, marcando um dos maiores triunfos da física de partículas moderna.

2.1.2.3 Quebra de Simetria e Mecanismo de Higgs

O mecanismo de Higgs é crucial para o Modelo Padrão, pois permite que os bósons W e Z adquiram massa enquanto o fóton permanece sem massa. Isso ocorre por meio da chamada *quebra espontânea de simetria*, quando o campo de Higgs adquire um valor esperado diferente de zero no vácuo. Esse campo interage com as partículas fundamentais, atribuindo massa proporcional à intensidade dessa interação.

Matematicamente, a quebra da simetria $SU(2)_L \times U(1)_Y$ para $U(1)_{EM}$ dá origem à separação entre os bósons massivos W^\pm , Z^0 e o fóton.

2.1.3 Limitações do Modelo Padrão

Apesar de seu enorme sucesso, o Modelo Padrão apresenta limitações conceituais e experimentais:

- **Gravidade:** O MP não incorpora a gravitação, descrita pela relatividade geral, uma teoria clássica. A unificação entre gravitação e mecânica quântica permanece um dos maiores desafios da física moderna.
- **Massa dos neutrinos:** O Modelo Padrão originalmente considera neutrinos sem massa. No entanto, experimentos de oscilação de neutrinos demonstram que eles têm massa, ainda que muito pequena.
- **Matéria escura:** Compreende cerca de 27% do universo e não interage com a luz. Nenhuma partícula do Modelo Padrão explica esse fenômeno.
- **Energia escura:** Corresponde a aproximadamente 68% do conteúdo energético do universo, responsável pela expansão acelerada do cosmos. Sua origem permanece desconhecida.
- **Assimetria matéria-antimatéria:** O universo observável contém muito mais matéria do que antimatéria, embora o Big Bang devesse ter criado ambas em proporções iguais.

2.1.4 Além do Modelo Padrão

Várias teorias além do Modelo Padrão já foram propostas:

- **Supersimetria (SUSY):** propõe uma simetria entre férmions e bósons, introduzindo parceiros supersimétricos para cada partícula do Modelo Padrão (HAGIWARA, 2001).

- **Teorias de Grande Unificação (GUTs):** propõem que todas as forças fundamentais (exceto a gravidade) sejam unificadas em uma única força a altas energias (BLUMENHAGEN; MOSTER; WEIGAND, 2006).
- **Teoria das Cordas:** sugere que as partículas fundamentais são modos de vibração de cordas unidimensionais (POLCHINSKI, 1998a; POLCHINSKI, 1998b).
- **Modelos com dimensões extras:** propõem que existam mais dimensões espaciais além das três usuais (POLCHINSKI, 1998b).

Essas teorias ainda não foram confirmadas experimentalmente, mas muitos experimentos em curso e planejados buscam por indícios de nova física.

2.1.5 Experimentos em Física de Partículas

Os avanços na física de partículas estão intrinsecamente ligados ao desenvolvimento de grandes aceleradores e detectores, fruto de colaborações internacionais que reúnem milhares de cientistas. A seguir apresentam-se os principais aparelhos e experimentos que moldaram nosso entendimento do mundo subatômico.

2.1.5.1 Aceleradores e detectores

O *Large Hadron Collider* (LHC) é o maior acelerador de partículas do mundo, instalado no CERN, na fronteira franco-suíça. Com seus 27 km de túnel circular, o LHC colide prótons a energias de até 13 TeV no centro de massa, recriando condições próximas às do universo primordial (CERN, 2025a). Nele estão instalados quatro experimentos principais:

- **ATLAS (do inglês, *A Toroidal LHC ApparatuS*) e CMS (do inglês, *Compact Muon Solenoid*):** detectores de propósito geral, projetados para investigar uma ampla gama de fenômenos, desde os mecanismos de quebra espontânea de simetria até a busca por nova física. Ambos foram essenciais na descoberta do bóson de Higgs em 2012 (CHATRCHYAN *et al.*, 2008; ATLAS COLLABORATION, 2008).
- **LHCb (*Large Hadron Collider beauty*):** especializado em colisões que produzem hádrons contendo quarks bottom (b), com foco em estudar violação de CP e diferenças entre matéria e antimatéria (JR. *et al.*, 2008).
- **ALICE (*A Large Ion Collider Experiment*):** dedicado ao estudo de colisões de íons pesados (chumbo-chumbo), que permitem a criação e caracterização do plasma de quarks e glúons, um estado de matéria presente nos primeiros instantes após o Big Bang (AAMODT *et al.*, 2008).

2.1.5.2 *Outros aceleradores históricos*

- **SLAC (Stanford Linear Accelerator Center)**: acelerador linear que, em 1968, revelou a estrutura de “partons” (quarks) nos prótons por meio de experimentos de espalhamento profundo (BRITANNICA, 2025).
- **Tevatron (Fermilab)**: até 2011, foi o maior acelerador circular dos Estados Unidos. Nos detectores CDF e DØ, em 1995, comprovou a existência do quark top (CDF COLLABORATION, 1995; DØ COLLABORATION, 1995).

2.1.6 **Física de Neutrinos**

A descoberta da oscilação de neutrinos, que implica que eles possuem massa, foi uma das maiores surpresas da física moderna e rendeu o Prêmio Nobel de Física de 2015. Isso requer uma extensão do Modelo Padrão, que originalmente tratava os neutrinos como partículas sem massa.

Alguns dos maiores experimentos de detecção e estudo de neutrinos são:

- **Super-Kamiokande**: detector de água em um observatório subterrâneo no Japão, projetado para estudar oscilações de neutrinos solares e atmosféricos, evidenciando que os neutrinos possuem massa (SUPER-KAMIOKANDE COLLABORATION, 1998).
- **DUNE (Deep Underground Neutrino Experiment)**: próxima geração de experimento de neutrinos em cavernas profundas nos EUA, com detectores de argônio líquido, para investigar oscilações de neutrinos de feixe de longa distância e buscar violações de simetria (GIL-BOTELLA, 2024).
- **IceCube**: observatório de neutrinos emérgicos enterrado na geleira do Polo Sul, capaz de detectar neutrinos cósmicos e estudar fenômenos astrofísicos de altíssima energia (ICECUBE COLLABORATION, 2017).

Esses resultados têm implicações profundas na cosmologia, na astrofísica e na busca por nova física.

2.1.7 **Cosmologia e Física de Partículas**

A física de partículas também desempenha um papel central na cosmologia. O universo primordial era uma sopa densa e quente de partículas fundamentais, e as leis da física de partículas determinam sua evolução. Conceitos como inflação, bariogênese, nucleossíntese primordial e radiação cósmica de fundo estão intrinsecamente ligados a essa área.

Além disso, a matéria escura, que compõe cerca de 27% do conteúdo energético do universo, ainda não foi identificada. Acredita-se que ela seja composta por partículas além do Modelo Padrão, como WIMPs ou axions, atualmente alvo de intensa pesquisa experimental (PARTICLE DATA GROUP, 2024).

2.1.8 Futuro da Física de Partículas

O futuro da física de partículas envolve tanto experimentos de alta energia quanto experimentos de precisão. Projetos propostos incluem:

- **FCC (Future Circular Collider):** proposto pelo CERN, com energia de até 100 TeV (CERN, 2025).
- **ILC (International Linear Collider):** colisor linear para estudar o bóson de Higgs com alta precisão (INTERNATIONAL LINEAR COLLIDER, 2013).
- **Experimentos de neutrinos de próxima geração:** como DUNE, nos EUA e Hyper-Kamiokande, localizado no Japão (HYPER-KAMIOKANDE COLLABORATION, 2018).

Esses projetos visam explorar fenômenos que possam revelar a física além do Modelo Padrão, aprofundar o entendimento das massas dos neutrinos, da matéria escura e da unificação das forças.

2.1.9 Considerações Finais

A física de partículas é uma das áreas mais fundamentais e desafiadoras da ciência moderna. Ao investigar as menores escalas do universo, ela fornece as chaves para compreendermos suas maiores estruturas e origens. Apesar dos avanços extraordinários nas últimas décadas, ainda há grandes mistérios a serem desvendados. O século XXI promete ser um período de descobertas transformadoras que podem revolucionar nossa compreensão do cosmos e da própria realidade.

2.2 ACELERADORES DE PARTÍCULAS

Os aceleradores de partículas representam uma das maiores conquistas tecnológicas e científicas do século XX, sendo ferramentas indispensáveis para a exploração das interações fundamentais da natureza. Por meio da aceleração de partículas subatômicas a velocidades próximas à da luz, esses dispositivos permitem a investigação de fenômenos físicos em escalas de energia extremamente elevadas, que não são observáveis em condições naturais. Além de contribuírem significativamente para a física de altas energias, os aceleradores

têm aplicações em diversas áreas, como medicina, indústria, segurança nacional e ciência dos materiais (GRUPEN; SHWARTZ, 2008).

O desenvolvimento dos aceleradores de partículas surgiu da necessidade de entender a composição da matéria em níveis cada vez mais profundos. Desde o tubo de raios catódicos, no final do século XIX, até as modernas instalações como o *Large Hadron Collider* (LHC), a evolução desses dispositivos acompanhou o progresso das tecnologias de controle de campos elétricos e magnéticos. Hoje, os aceleradores são essenciais não apenas para a descoberta de novas partículas, como o bóson de Higgs, mas também para o avanço de técnicas aplicáveis à sociedade, como a radioterapia, a análise de materiais e a produção de radioisótopos (ATLAS COLLABORATION, 2012).

A trajetória dos aceleradores começou no final do século XIX, com dispositivos como o tubo de raios catódicos, que permitiu o estudo do comportamento de elétrons em ambientes controlados. A inovação no campo ocorreu em 1928, quando Ernest Lawrence e M. Stanley Livingston desenvolveram o Ciclotron, que utilizava campos magnéticos e elétricos para acelerar partículas em trajetórias espirais, possibilitando a construção de máquinas compactas e eficientes em termos de energia (LAWRENCE; LIVINGSTON, 1932).

Durante as décadas seguintes, dispositivos como o sincrociclotron e o síncrotron introduziram ajustes dinâmicos no campo magnético, permitindo a aceleração de partículas a energias relativísticas. Na década de 1970, a introdução dos colisores, que permitiram colisões frontais de feixes de partículas, maximizou a energia disponível para a criação de novas partículas e possibilitou a investigação de fenômenos além do Modelo Padrão (SESSLER; WILSON, 2007).

A partir dos anos 1980, os aceleradores passaram a ser utilizados em projetos científicos de grande escala. O LEP (*Large Electron-Positron Collider*), no CERN, foi um dos primeiros a explorar colisões de alta energia de elétrons e pósitrons, preparando o terreno para a criação do LHC. Inaugurado em 2008, o LHC tornou-se o maior e mais potente acelerador de partículas do mundo, com 27 km de circunferência e capacidade para colidir prótons a 13 TeV, e com luminosidade até 10^{34} (BRAIBANT; GIACOMELLI; SPURIO, 2011). Essa infraestrutura permitiu avanços sem precedentes, como a confirmação do bóson de Higgs em 2012 e a exploração de fenômenos relacionados à matéria escura e à violação de simetria CP (CMS COLLABORATION, 2012).

2.2.1 Princípios de Funcionamento

O funcionamento dos aceleradores baseia-se em dois mecanismos fundamentais: a aceleração por campos elétricos e o direcionamento por campos magnéticos. Enquanto os campos elétricos aumentam a energia cinética das partículas, os campos magnéticos controlam suas trajetórias, garantindo a estabilidade e o foco do feixe. Essa interação

é ajustada cuidadosamente para minimizar perdas e maximizar a eficiência energética, especialmente em trajetórias circulares, onde há maior emissão de radiação síncrotron (JACKSON, 1998).

Aceleradores lineares (*linacs*) impulsionam partículas ao longo de um caminho reto, sendo amplamente utilizados em aplicações que exigem feixes de alta precisão, como a radioterapia. A ausência de perdas por radiação síncrotron torna os linacs uma escolha eficiente para experimentos de pequena e média escala, além de serem utilizados como injetores para grandes aceleradores circulares (WIEDEMANN, 2007).

Nos aceleradores circulares, as partículas percorrem órbitas controladas por campos magnéticos, permitindo que sejam continuamente aceleradas em cada volta. Apesar das perdas energéticas, essa abordagem permite atingir energias acumulativas elevadas, sendo ideal para experimentos de física de altas energias. Colisores como o LHC são exemplos dessa tecnologia, viabilizando colisões frontais que proporcionam a máxima energia de interação possível (GRUPEN; SHWARTZ, 2008).

2.2.2 Tipos de Acionamento

2.2.2.1 *Acionamento Eletrostático*

Aceleradores eletrostáticos, como o gerador de Van de Graaff e o acelerador Cockcroft-Walton, utilizam campos elétricos constantes para acelerar partículas. Embora limitados pela rigidez dielétrica dos materiais, esses dispositivos são utilizados em aplicações industriais, pesquisa de baixa energia e produção de feixes iônicos (SESSLER; WILSON, 2007).

2.2.2.2 *Acionamento por Radiofrequência*

Os aceleradores que utilizam cavidades de radiofrequência (RF) geram campos elétricos oscilantes que permitem a repetição do processo de aceleração. Essa abordagem é amplamente empregada em linacs e síncrotrons, sendo crucial para alcançar energias relativísticas e manter a estabilidade do feixe (WIEDEMANN, 2007).

2.2.2.3 *Acionamento por Plasma*

Os aceleradores de plasma, uma tecnologia emergente, utilizam ondas de plasma para gerar campos elétricos intensos, permitindo a aceleração em distâncias extremamente curtas. Com potencial para revolucionar o campo, esses dispositivos prometem reduzir significativamente o tamanho e o custo das instalações, além de alcançar energias superiores com menor impacto ambiental (TAJIMA; DAWSON, 1979).

2.2.3 Importância e Futuro

Os aceleradores de partículas não apenas consolidaram seu papel como ferramentas indispensáveis para a física de altas energias, mas também abriram novas perspectivas para diversas áreas do conhecimento humano. Desde o início do século XX, a evolução tecnológica desses dispositivos permitiu avanços científicos que transformaram nossa compreensão do universo, revelando partículas fundamentais e validando teorias como o Modelo Padrão. No entanto, o impacto dos aceleradores vai além da ciência pura: suas aplicações em setores como a medicina, a indústria e a tecnologia de materiais demonstram seu valor prático, tornando-os essenciais para o desenvolvimento socioeconômico e tecnológico global.

No futuro, o campo dos aceleradores de partículas enfrenta novos desafios e oportunidades. A busca por energias ainda mais altas, com o objetivo de investigar fenômenos que vão além do Modelo Padrão, como a matéria escura, a energia escura e as dimensões extras, impulsiona o desenvolvimento de tecnologias cada vez mais sofisticadas. Iniciativas como o *Future Circular Collider* (FCC), projetado para suceder ao LHC e possibilitar novas descobertas, expandindo ainda mais os horizontes da física de partículas.

Além disso, o avanço das tecnologias de miniaturização e eficiência energética, como os aceleradores baseados em plasma, representa uma mudança de paradigma. A aceleração em distâncias extremamente curtas pode tornar esses dispositivos mais compactos, econômicos e acessíveis a centros de pesquisa menores, democratizando o acesso a experimentos de alta energia e ampliando o leque de aplicações práticas. Isso permitirá que, no futuro, aceleradores de partículas sejam usados não apenas em grandes instalações internacionais, mas também em universidades, hospitais e indústrias, fomentando um ecossistema global de inovação científica e tecnológica.

Por fim, o futuro dos aceleradores de partículas está intrinsecamente ligado à colaboração internacional. Projetos como o CERN exemplificam a importância do esforço conjunto entre nações para enfrentar desafios científicos de grande escala. Essa cooperação global continuará a ser um pilar fundamental para o desenvolvimento de novas gerações de aceleradores, garantindo que a ciência avance de forma coletiva e sustentável, em benefício de toda a humanidade. Dessa forma, os aceleradores de partículas não são apenas instrumentos científicos, mas também símbolos do progresso humano e da capacidade de superação das fronteiras do conhecimento.

2.3 O CERN E O LHC

O CERN (Organização Europeia para Pesquisa Nuclear, na sigla em inglês) é um dos maiores e mais respeitados centros de pesquisa em física de partículas do mundo (CERN, 2025b). Localizado em Genebra, na Suíça, o centro de pesquisa desempenha um papel central na física moderna, sendo um dos principais centros de desenvolvimento de

aceleradores de partículas e tecnologias relacionadas. Sua missão inclui a investigação das leis fundamentais da natureza, o desenvolvimento de tecnologias de ponta e a promoção da colaboração internacional em ciência e engenharia.

Após a Segunda Guerra Mundial, houve um período de grande avanço na física nuclear, onde muitos cientistas, como Niels Bohr e Albert Einstein, estavam tentando compreender melhor a estrutura da matéria. Durante este período, as potências europeias reconheciam a importância de investir em pesquisa científica de grande escala. Em 1950, o físico suíço Léon Brillouin sugeriu a ideia de um centro de pesquisa pan-europeu dedicado à física nuclear, que se tornaria o CERN.

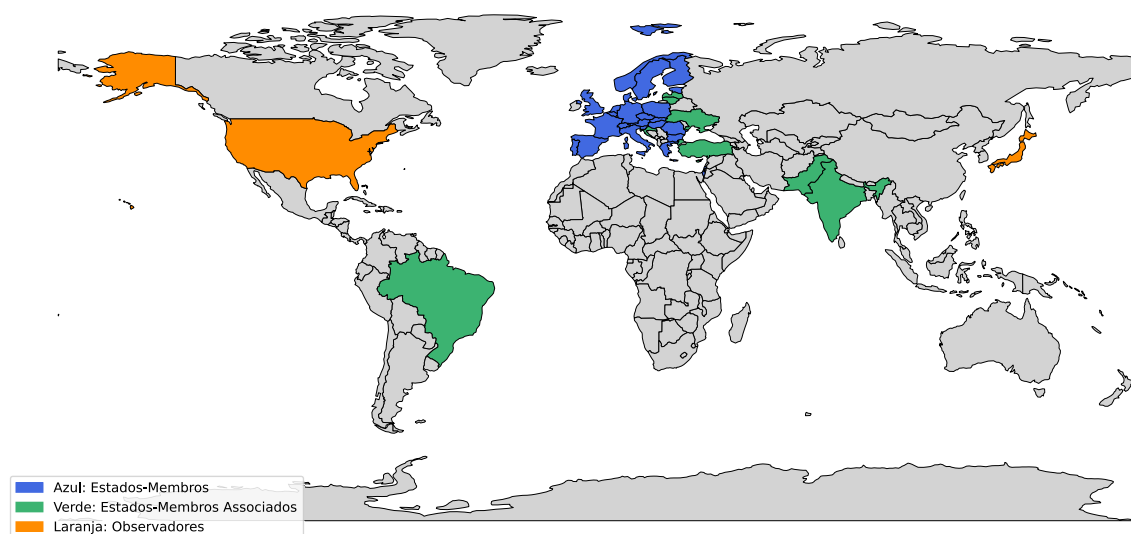
Formalmente, o CERN foi fundado em 1954 pelos doze Estados-Membros Fundadores — Bélgica, Dinamarca, França, Alemanha, Grécia, Itália, Noruega, Países Baixos, Reino Unido, Suécia, Suíça e Portugal — com o objetivo de desenvolver aceleradores de partículas e realizar experimentos em física nuclear. O primeiro equipamento a entrar em operação foi o Sincrociclotron, em 1957, seguido pelo Síncrotron de Prótons, que iniciou suas atividades em 1959, marcando o começo da participação do CERN em experimentos de alta energia em física de partículas.

Os Estados-Membros possuem deveres e privilégios especiais. Eles contribuem para os custos de capital e operacionais dos programas do CERN e são representados no Conselho, responsável por todas as decisões importantes sobre a Organização e suas atividades.

Atualmente, o CERN conta com 25 Estados-Membros plenos: Áustria, Bélgica, Bulgária, República Tcheca, Dinamarca, Estônia, Finlândia, França, Alemanha, Grécia, Hungria, Israel, Itália, Países Baixos, Noruega, Polônia, Portugal, Romênia, Sérvia, Eslováquia, Eslovênia, Espanha, Suécia, Suíça e Reino Unido. O Chipre encontra-se em estágio prévio como Estado-Membro Associado, e Brasil, Croácia, Índia, Letônia, Lituânia, Paquistão, Turquia e Ucrânia são Estados-Membros Associados. Os Estados Unidos da América e o Japão possuem o status de Observadores no Conselho do CERN. A distribuição geográfica pode ser conferida na Figura 2.

O centro de pesquisa se tornou um centro internacional de pesquisa de ponta, com ênfase na física de partículas, buscando entender as interações fundamentais da matéria e a origem do universo. A ideia de construir um acelerador de partículas de alta energia para estudar as interações subatômicas começou a ser discutida nas décadas de 1980 e 1990, quando a comunidade científica buscava maneiras de expandir a compreensão das partículas fundamentais. O Grande Colisor de Hádrons (LHC) foi projetado para ser o sucessor do Super Proton Synchrotron (SPS) do CERN e, mais tarde, foi decidido que ele se tornaria o maior acelerador de partículas do mundo (CERN, 2025a). Sua construção começou em 1998, e o projeto foi um esforço colaborativo envolvendo milhares de cientistas, engenheiros e técnicos de diferentes países. Sua construção foi concluída em 2008, após

Figura 2 – Status de países vinculados ao CERN



Fonte: Elaborado pelo autor (2025).

uma série de desafios técnicos e financeiros.

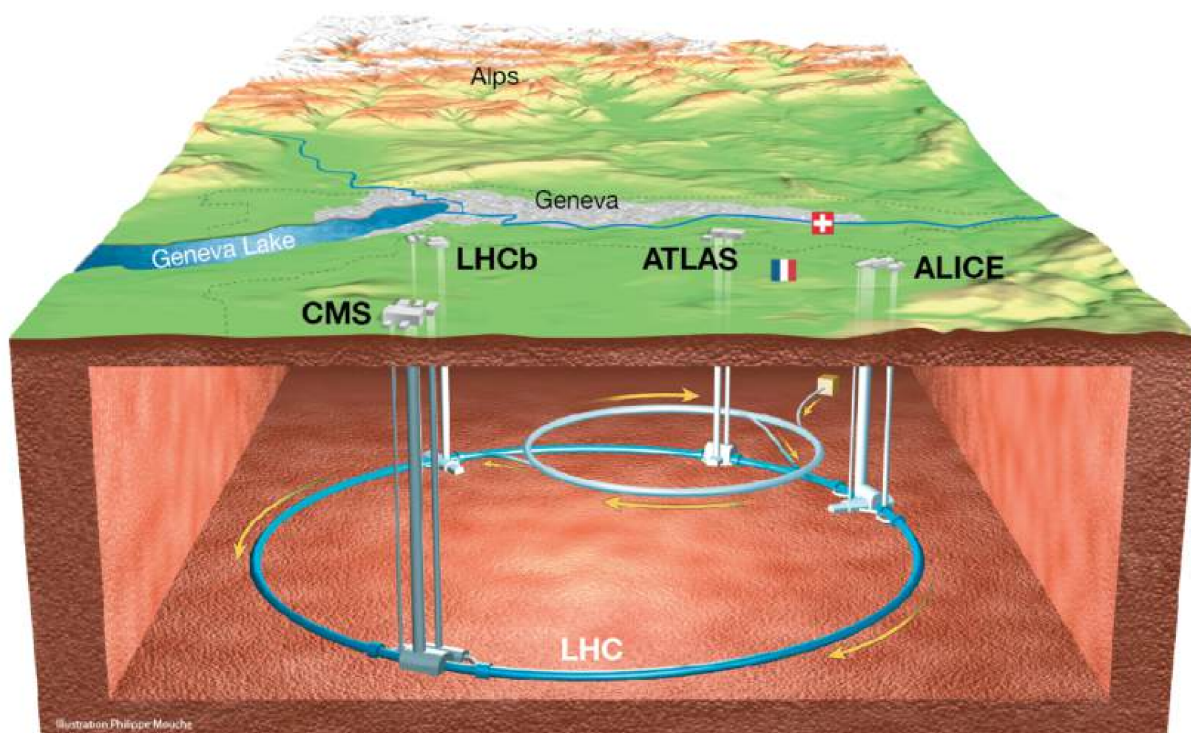
O LHC é um acelerador de partículas circular com 27 quilômetros de circunferência, localizado a uma profundidade entre 45 e 170 metros sob a terra, como pode ser visto na Figura 3 (CERN, 2025a). Ele ocupa uma parte significativa da infraestrutura subterrânea do CERN e atravessa a fronteira entre a Suíça e a França (MOUCHE, 2014). O acelerador é composto por 8 seções principais que contêm potentes magnetos supercondutores, além de complexos sistemas de resfriamento e controle de partículas. Esse acelerador foi projetado para operar com uma série de feixes de prótons, íons pesados e outros tipos de partículas, acelerando-os até velocidades próximas à da luz.

O funcionamento do LHC é baseado em princípios da física de aceleradores e da teoria eletromagnética. No coração do LHC, os feixes de partículas são acelerados utilizando campos elétricos gerados por cavidades ressonantes de radiofrequência (RF). Esses campos aumentam a velocidade das partículas à medida que elas circulam ao longo do acelerador. É utilizado um sistema complexo de ímãs supercondutores para manter as partículas em suas trajetórias circulares e concentrá-las em feixes extremamente finos.

Os feixes do colisor são compostos por grupos de cerca de $1,15 \times 10^{11}$ prótons, acelerados a energias de modo que sua velocidade se aproxima muito da da luz. Cada feixe carrega aproximadamente 362 MJ de energia cinética, equivalente ao poder explosivo de cerca de 77 kg de TNT. O intervalo temporal entre a passagem de dois grupos consecutivos pelos pontos de colisão é de 25 ns, ou seja, os feixes se cruzam com uma frequência de 40 MHz (EVANS; BRYANT, 2008).

Quando os feixes de partículas atingem uma alta energia, são colididos em pontos

Figura 3 – Visão do LHC



Fonte: (MOUCHE, 2014)

específicos, onde os detectores são localizados. Essas colisões geram partículas secundárias que são capturadas por detectores como o ATLAS, o CMS, o ALICE e o LHCb. O acelerador pode realizar milhões de colisões por segundo, o que gera uma enorme quantidade de dados que são posteriormente analisados para observar fenômenos subatômicos.

O principal objetivo do Colisor de Hádrons é explorar o Modelo Padrão da física de partículas e investigar fenômenos que vão além deste modelo. O Modelo Padrão descreve as partículas fundamentais e as forças que governam as interações entre elas, mas existem questões que ainda permanecem sem resposta, como a natureza da matéria escura, a origem das massas das partículas e o comportamento das partículas em energias extremamente altas.

A infraestrutura de colisão é projetada para permitir a exploração dessas questões de maneira sistemática. Por meio das colisões de partículas de alta energia, os cientistas podem criar condições extremas semelhantes às que existiam nos primeiros momentos após o Big Bang. Isso possibilita estudar a criação de partículas exóticas e observar fenômenos como o comportamento das partículas fundamentais, a quebra de simetria e a descoberta de novas partículas e interações.

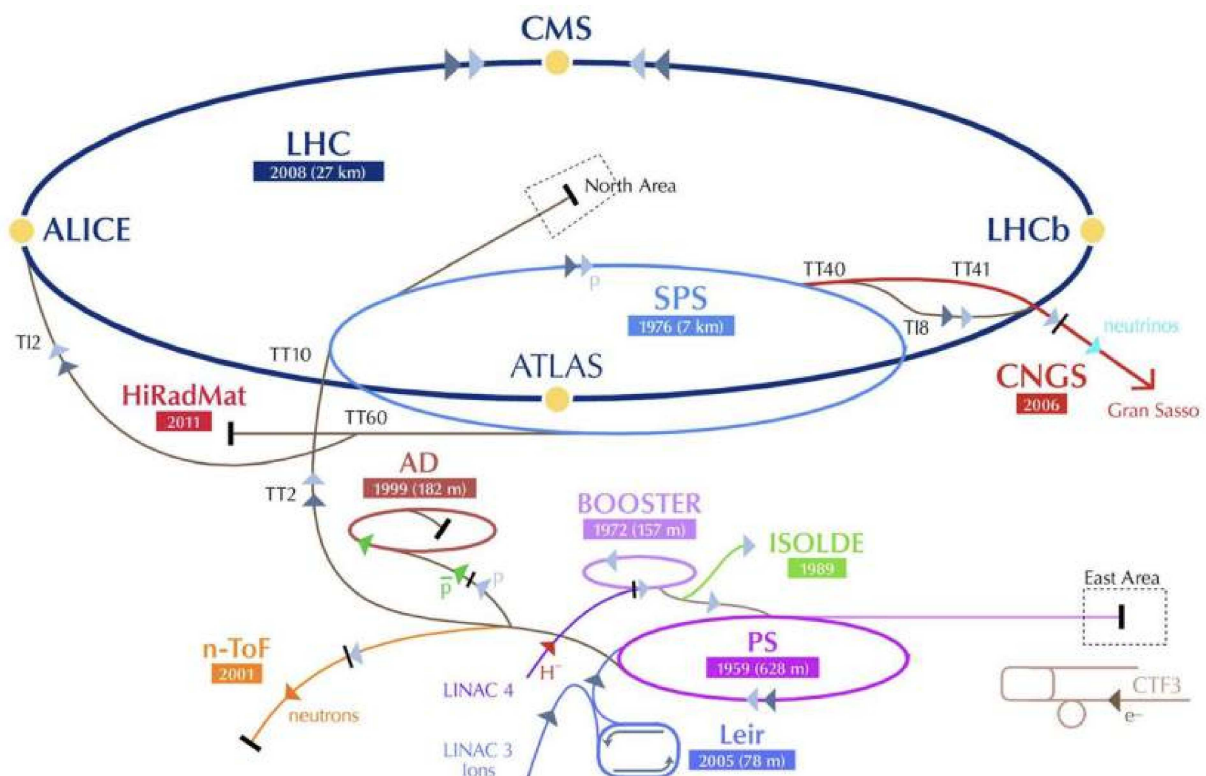
Uma das maiores conquistas do LHC foi a descoberta do bóson de Higgs, uma partícula que foi prevista teoricamente em 1964 por Peter Higgs e outros cientistas. O

bóson de Higgs é a partícula associada ao campo de Higgs, o qual confere massa a outras partículas fundamentais, como os quarks e os léptons (HALZEN; MARTIN, 1984). Sem o campo de Higgs, as partículas não teriam massa, e o universo como conhecemos não seria possível.

Em 2012, após anos de pesquisa e experimentação, os cientistas do CERN anunciaram a descoberta do bóson de Higgs, uma realização histórica que confirmou uma das últimas previsões do Modelo Padrão. Essa descoberta foi feita através do trabalho conjunto dos experimentos ATLAS e CMS, que detectaram sinais do bóson de Higgs em colisões de alta energia no LHC. Tal realização foi amplamente celebrada pela comunidade científica e resultou no Prêmio Nobel de Física de 2013, concedido a Peter Higgs e François Englert, por suas contribuições à teoria que previu a existência da partícula.

O LHC está equipado com vários detectores de partículas de alta precisão, sendo que os experimentos mais notáveis são o ATLAS, o CMS, o ALICE e o LHCb, que podem ser visualizados na Figura 4. Cada um desses detectores tem um objetivo específico e utiliza uma combinação de tecnologias avançadas para observar e registrar as colisões de partículas geradas no LHC.

Figura 4 – Complexo de aceleradores do CERN



Fonte: (MARCASTEL, 2013)

Os aceleradores de partículas no CERN são essenciais para a realização de experimentos de física de partículas. Eles geram feixes de partículas subatômicas, como prótons

e elétrons, e os aceleram a energias extremamente altas, permitindo colisões entre essas partículas. O CERN possui vários aceleradores de partículas, cada um projetado para realizar tarefas específicas, desde a aceleração inicial até a colisão final das partículas.

Além do LHC, que é o acelerador mais famoso e poderoso, o CERN também opera outros aceleradores como o *Proton Synchrotron* (PS), o *Super Proton Synchrotron* (SPS) e o Linac 4, que são usados em diferentes etapas do processo de aceleração e experimentação. O Linac 4, por exemplo, é responsável pela aceleração inicial dos feixes de prótons antes que eles sejam enviados para o PS e o SPS.

2.3.1 Colaboração Internacional e Inovação Tecnológica

O CERN é um exemplo notável de colaboração internacional em ciência e tecnologia. Embora esteja localizado na Suíça, o CERN envolve cientistas, engenheiros e técnicos de mais de 100 países, com mais de 10.000 colaboradores trabalhando no centro. O CERN não é apenas uma organização de pesquisa, mas também uma plataforma global para a troca de conhecimentos e o desenvolvimento de novas tecnologias. Muitos dos avanços tecnológicos que ocorreram no CERN tiveram impactos significativos fora do campo da física, como a criação da *World Wide Web* por Tim Berners-Lee, que originalmente foi desenvolvida para compartilhar informações entre cientistas e facilitar a colaboração (BERNERS-LEE *et al.*, 1994).

O instituto também desenvolve e compartilha tecnologias essenciais para os aceleradores de partículas, como detectores, sistemas de resfriamento supercondutores, software de análise de dados e novas técnicas de computação. Essas inovações têm aplicações em diversas áreas, incluindo medicina (por exemplo, em terapias de radiação), indústria e segurança nacional. O uso de aceleradores de partículas no CERN ajudou a aprimorar técnicas de diagnóstico médico, como a tomografia por emissão de pósitrons (PET), além de possibilitar o desenvolvimento de materiais mais avançados e resistentes.

2.3.2 O Futuro do CERN

A entidade está constantemente explorando novas fronteiras da física e da tecnologia. O centro está envolvido em vários projetos futuros que visam ampliar ainda mais os limites do conhecimento humano. Seu futuro está marcado por projetos como o *Future Circular Collider* (FCC), que é um plano para construir um acelerador de partículas ainda maior que o LHC, com o objetivo de explorar energias muito mais altas. O FCC visa investigar questões fundamentais como a natureza da matéria escura e a possibilidade de novas partículas além do Modelo Padrão. O novo acelerador também poderá abrir caminho para o estudo de novas teorias da física, como a supersimetria e a gravidade quântica.

Além disso, a organização continua a buscar maneiras de melhorar a eficiência e reduzir os custos das suas operações, com um foco crescente em inovações tecnológicas, como

a aceleração por plasma e o uso de computação quântica. Essas tecnologias emergentes têm o potencial de transformar o futuro dos aceleradores de partículas, tornando-os mais compactos, eficientes e acessíveis a uma gama ainda maior de pesquisadores e instituições em todo o mundo.

O CERN desempenha um papel fundamental na exploração da física de partículas e no avanço da ciência e da tecnologia. Desde a sua fundação, a organização tem sido um centro de inovação e colaboração internacional, proporcionando importantes descobertas sobre a natureza do universo e impulsionando o desenvolvimento de tecnologias que têm aplicações em diversos campos. O centro continua a estar na vanguarda da pesquisa científica, explorando as fronteiras do conhecimento e desenvolvendo novas tecnologias que beneficiarão a humanidade em muitas áreas. O futuro do CERN é promissor, com novos projetos e colaborações que visam expandir ainda mais os limites do nosso entendimento sobre o universo.

2.4 O EXPERIMENTO ATLAS

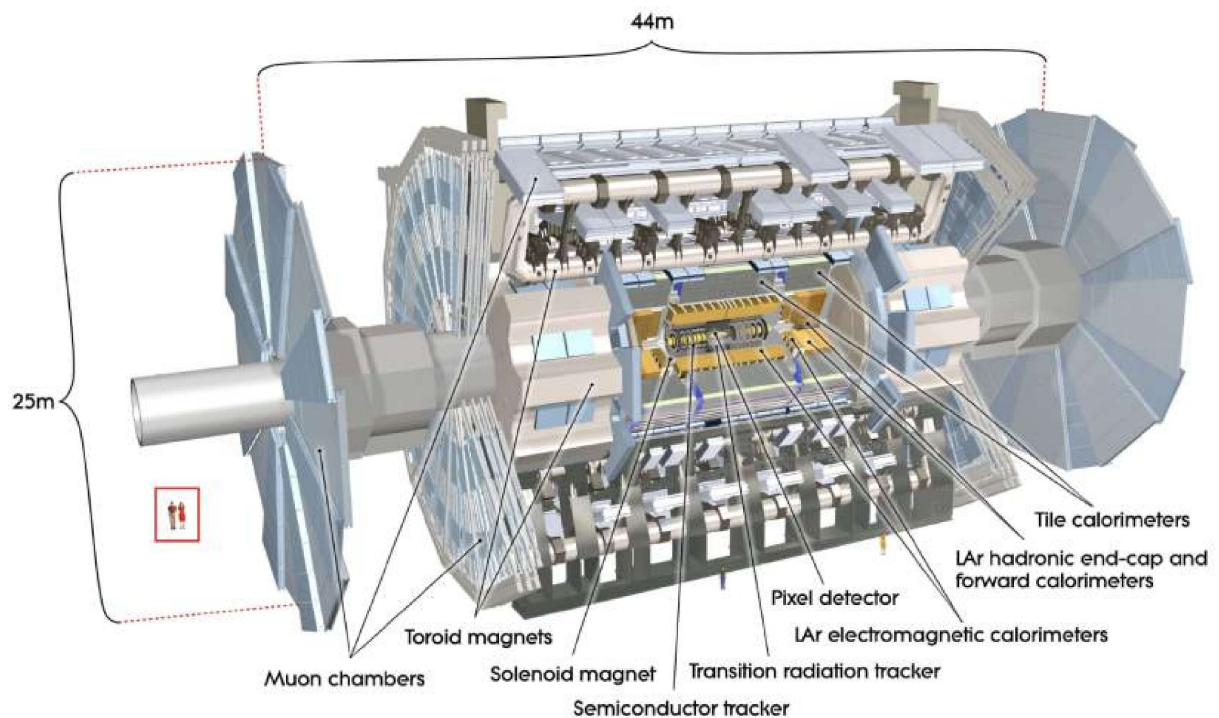
Sendo um dos quatro principais detectores instalados ao longo do LHC, juntamente com os experimentos CMS, ALICE e LHCb, o experimento ATLAS (do inglês *A Toroidal LHC Apparatus*) é um projeto de larga escala, operado por uma colaboração internacional que reúne mais de 3000 pesquisadores provenientes de diversas instituições acadêmicas e científicas distribuídas globalmente. O ATLAS é classificado como um detector de propósito geral, ou seja, foi projetado com o objetivo de maximizar a capacidade de investigação das diversas possibilidades físicas proporcionadas pela elevada energia e luminosidade do colisor (ATLAS COLLABORATION, 1994).

Ao contrário de experimentos especializados em processos específicos, o ATLAS busca detectar e analisar uma ampla gama de interações entre partículas, tanto aquelas previstas pelo Modelo Padrão da física de partículas quanto possíveis fenômenos que extrapolam os limites dessa teoria. Dessa forma, o experimento permite a realização de medidas de alta precisão em processos eletrofracos, de cromodinâmica quântica e de física de sabores, ao mesmo tempo em que oferece uma plataforma robusta para a busca de indícios de novas partículas, interações exóticas ou violações de simetrias fundamentais. A descoberta do bóson de Higgs, anunciada em 2012 em conjunto com o experimento CMS, destaca-se como o principal resultado alcançado até o momento.

Assim como ilustrado na Figura 5, o detector apresenta dimensões notáveis, com aproximadamente 44 metros de comprimento, 25 metros de altura e 25 metros de largura. Sua geometria é cilíndrica, composta por uma seção central (denominada *barrel*) e duas regiões de extremidade (*end-caps*), dispostas simetricamente em torno do ponto de colisão denominado Ponto 1 do LHC. A estrutura interna do detector é organizada em camadas concêntricas, nas quais são distribuídos subsistemas especializados na detecção e medição

de diferentes tipos de partículas e propriedades físicas.

Figura 5 – Estrutura do ATLAS e seus componentes



Fonte: (AAD *et al.*, 2011)

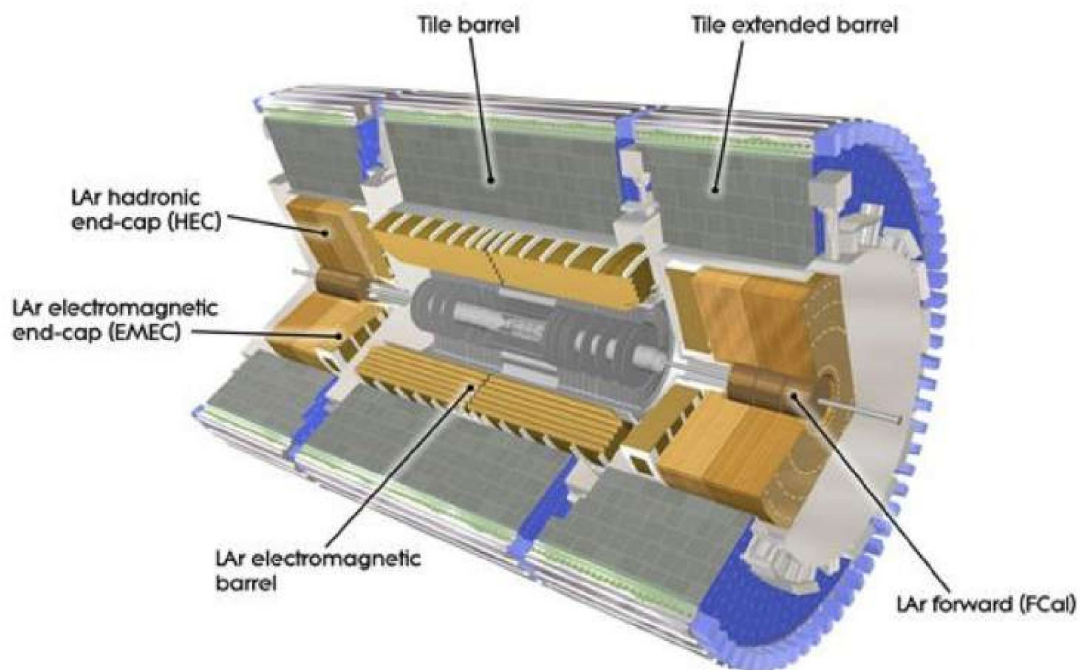
O objetivo primordial do ATLAS consiste em identificar e caracterizar as partículas geradas nas colisões próton-próton, determinando trajetória, energia e momento. O desafio experimental é grande devido ao elevado número de colisões simultâneas que ocorrem a cada intervalo de 25 nanosegundos — efeito conhecido como *pile-up*. Para enfrentar esse cenário, o experimento incorpora uma série de requisitos técnicos:

- alta granularidade espacial, necessária para resolver vértices distintos em regiões densamente povoadas por partículas;
- ampla aceitação geométrica, garantindo cobertura angular próxima de 4π estereorradianos;
- resposta temporal compatível com o espaçamento entre feixes do LHC, exigindo tempos de leitura inferiores a 25 ns;
- alta resolução em energia, fundamental para discriminar partículas e reconstruir eventos com precisão;
- capacidade de rastreamento de partículas carregadas, a fim de localizar o vértice primário da colisão próton-próton e identificar vértices secundários associados a partículas instáveis;

A principal função do Inner Detector é detectar as partículas eletricamente carregadas que atravessam a região central do detector, permitindo a reconstrução precisa de suas trajetórias curvas, a partir das quais é possível estimar o momento linear. As medições realizadas pelo ID são não destrutivas, ou seja, as partículas sofrem perdas de energia desprezíveis ao atravessá-lo, sem serem absorvidas. Esse subsistema é envolvido por um solenóide central que gera um campo magnético de aproximadamente 2 T, necessário para curvar as trajetórias das partículas carregadas.

Mais externamente ao ID localizam-se os calorímetros, responsáveis pela absorção total das partículas e pela medição de sua energia — caracterizando um sistema de medida destrutiva. Devido às diferenças nos mecanismos de interação entre partículas eletromagnéticas (como elétrons e fótons) e hádrons (como prótons e píons), os calorímetros são subdivididos em dois sistemas distintos: o calorímetro eletromagnético (EC) e o calorímetro hadrônico (HC). O EC é posicionado primeiro, absorvendo partículas eletromagnéticas, enquanto o HC, localizado após o EC, é destinado à contenção de hádrons, que apresentam maior poder de penetração. Sua estrutura pode ser observada na Figura 7.

Figura 7 – Localização dos Calorímetros no Experimento ATLAS

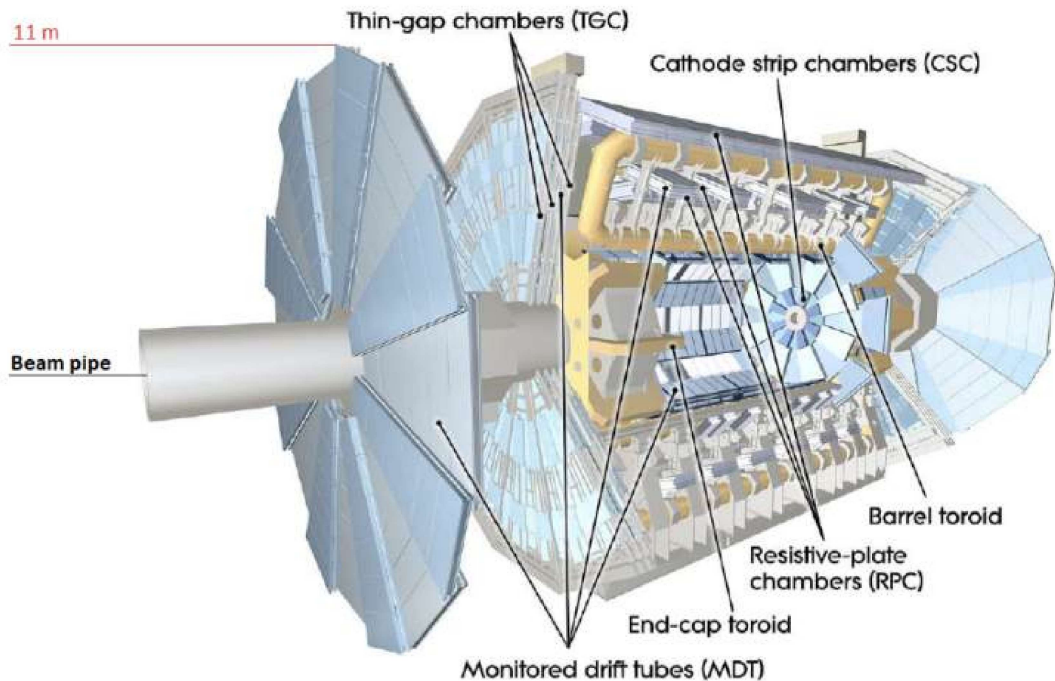


Fonte: (PEREIRA, 2019)

A região mais externa do detector ATLAS é ocupada pelo espectrômetro de múons. Como os múons são partículas muito penetrantes, atravessam praticamente todos os demais subsistemas sem grandes perdas de energia. No entanto, sua detecção é essencial para a análise de uma ampla gama de fenômenos físicos. O espectrômetro de múons é responsável por identificar essas partículas e medir seu momento com precisão. Para

possibilitar a curvatura de suas trajetórias, o sistema é envolto por um conjunto de grandes ímãs toroidais de núcleo de ar, compostos por oito bobinas supercondutoras dispostas ao redor do barril central e duas estruturas adicionais nos tampões terminais, conforme a Figura 8. Esses ímãs geram o campo magnético necessário para a deflexão dos múons, possibilitando sua análise cinemática detalhada.

Figura 8 – Detector de Múons



Fonte: (AAD *et al.*, 2011)

2.5 CALORÍMETRO HADRÔNICO DE TELHAS (TILECAL)

O Calorímetro Hadrônico é uma das partes cruciais do detector ATLAS, projetado para medir a energia de partículas hadrônicas, como prótons, nêutrons, píons e kaons, produzidas nas colisões de partículas no Grande Colisor de Hádrons (LHC). Este componente desempenha um papel fundamental na reconstrução do evento total, permitindo que os físicos identifiquem e caracterizem adequadamente as partículas que não interagem com outros subsistemas, como os elétrons e fótons, que são detectados principalmente pelo calorímetro eletromagnético.

O Calorímetro Hadrônico no ATLAS é dividido em duas partes principais, baseadas em tecnologias distintas:

- **Calorímetro Hadrônico de Barril** (*Barrel Hadronic Calorimeter – TileCal*): Posicionado ao redor da região central do detector, cobrindo aproximadamente a faixa de pseudorapidez $|\eta| < 1,7$. Trata-se de um calorímetro de cintilador, utilizando

aço como material absorvente e cintiladores plásticos como meio ativo, com a luz coletada por fotomultiplicadores (ATLAS COLLABORATION, 1996b).

- **Calorímetro Hadrônico de Tampo** (*End-Cap Hadronic Calorimeter – LAr HEC*): Localizado nas extremidades do detector, cobrindo as regiões com $1,5 < |\eta| < 3,2$. É um calorímetro de argônio líquido, utilizando cobre como material absorvente e o próprio argônio líquido como meio ativo (ATLAS COLLABORATION, 1996a).

Ambos operam pelo princípio de amostragem, em que um material absorvente denso alterna com um meio ativo capaz de detectar as partículas secundárias geradas nas interações dos hádrons. No TileCal, a energia depositada no cintilador é convertida em luz e detectada por tubos fotomultiplicadores. No LAr HEC, a ionização do argônio líquido produz sinais elétricos diretamente, que são amplificados e digitalizados pelo sistema de leitura.

Quando uma partícula hadrônica interage com o material absorvente, produz um chuveiro de partículas secundárias. A energia dessas partículas é proporcional à energia da partícula incidente, permitindo a medição da energia total depositada no detector (AGARAS, 2021).

O Calorímetro Hadrônico de Telhas (TileCal) é, portanto, um dos subdetectores mais importantes do experimento ATLAS. Ele foi projetado para medir a energia de partículas hadrônicas resultantes das colisões de alta energia no LHC, desempenhando um papel crucial na reconstrução de jatos hadrônicos, na identificação de partículas quase invisíveis, como neutrinos, e na medição do momento transversal faltante (MET), que pode ser indicativo da presença de partículas hipotéticas como aquelas relacionadas à matéria escura.

Desde sua concepção, o TileCal foi projetado como um dos pilares do sistema de detecção do ATLAS, atendendo à necessidade de medir com elevada precisão a energia de hádrons em um ambiente de altíssima taxa de eventos. O projeto priorizou robustez mecânica, estabilidade a longo prazo e resistência à radiação acumulada, considerando que as taxas de colisão do LHC atingem até 40 milhões de cruzamentos de feixes por segundo. A escolha por um calorímetro de amostragem com cintiladores plásticos e estrutura em aço permitiu combinar alta confiabilidade com facilidade de manutenção, por meio de módulos substituíveis (AAD *et al.*, 2010).

Com a transição para o *High-Luminosity LHC* (HL-LHC), prevista para meados da próxima década, o ambiente de operação do TileCal se tornará ainda mais desafiador: espera-se um aumento significativo da luminosidade integrada, do *pile-up* médio e da ocupação por célula calorimétrica. Isso exige eletrônica de leitura capaz de lidar com maior taxa de dados, algoritmos de reconstrução mais sofisticados e estratégias de calibração adaptadas à maior degradação por radiação (GONÇALO, 2019).

2.5.1 Arquitetura e Funcionamento

Estruturalmente, o TileCal é composto por módulos cilíndricos dispostos em torno do feixe de prótons, segmentados em três regiões funcionais (MARJANOVIĆ, 2018):

- **Barril Central (Central Barrel):** cobre a região de pseudorapidez $|\eta| < 1,0$ e é formado por 64 módulos dispostos azimutalmente. Cada módulo é dividido longitudinalmente em três camadas de profundidade distintas (A, BC e D), otimizadas para amostrar diferentes estágios do desenvolvimento do chuveiro hadrônico.
- **Barril Estendido (Extended Barrel):** localizado nas extremidades do barril central, abrange $0,8 < |\eta| < 1,7$ e utiliza geometria e materiais idênticos, porém com menor espessura longitudinal para acomodar a transição com os calorímetros de tampão.
- **Interface de Transição:** região de sobreposição e alinhamento mecânico entre o barril central e o barril estendido, assegurando cobertura calorimétrica contínua e minimizando lacunas em η .

O princípio de funcionamento baseia-se na alternância de placas de aço inoxidável (absorvedor) e telhas de plástico cintilante (meio ativo). O aço induz a cascata de partículas secundárias — o chuveiro hadrônico —, enquanto o cintilador converte a passagem de partículas carregadas em pulsos de luz. Essa luz é transportada por fibras ópticas até os PMTs, que a convertem em sinais elétricos proporcionais à energia depositada (ATLAS COLLABORATION, 1996b; DYER, 2004). A escolha pelo aço inoxidável como absorvedor garante não apenas densidade suficiente para conter o chuveiro na região central, mas também estabilidade mecânica frente a deformações térmicas e efeitos de radiação ao longo da vida útil do detector.

A segmentação do TileCal desempenha papel central na capacidade de reconstruir eventos com elevada resolução espacial. Na direção **radial**, a energia é amostrada em camadas sucessivas ao longo do raio, permitindo acompanhar a evolução longitudinal do chuveiro hadrônico. Na direção **longitudinal**, essa segmentação possibilita estimar a profundidade de penetração das partículas e identificar eventuais flutuações no desenvolvimento do chuveiro. Já na direção **azimutal**, a divisão em módulos fornece a distribuição angular das partículas, fundamental para a reconstrução precisa de jatos hadrônicos e para a correlação com outros subsistemas (ATLAS COLLABORATION, 1996b). Essa granularidade não apenas viabiliza uma reconstrução tridimensional detalhada, como também é essencial para procedimentos de calibração localizada e para a mitigação de efeitos como o *pile-up*.

As partículas carregadas que atravessam o material ativo excitam as moléculas do cintilador plástico, produzindo fótons na faixa do ultravioleta e visível. Esses pulsos de luz

são coletados por fibras ópticas posicionadas estrategicamente entre as telhas de cintilação, de modo a maximizar a captura isotrópica da emissão e minimizar perdas por atenuação. As fibras transportam a luz até os tubos fotomultiplicadores, dispositivos sensíveis que convertem a radiação óptica em sinais elétricos (MLYNARÍKOVÁ *et al.*, 2017).

Os PMTs utilizados no TileCal combinam três características cruciais para operação no LHC (STAROVOITOV *et al.*, 2022):

- **Alta eficiência quântica**, garantindo a conversão efetiva da luz em elétrons no fotocatodo.
- **Capacidade de operar em altas taxas de eventos** sem saturação, preservando a linearidade da resposta mesmo sob condições de alta luminosidade.
- **Resistência à radiação**, assegurando desempenho estável ao longo de anos de operação contínua.

Internamente, o sinal inicial gerado no fotocatodo é amplificado por uma cadeia de dinodos, resultando em um ganho total da ordem de 10^6 a 10^7 , suficiente para produzir pulsos elétricos detectáveis com excelente relação sinal-ruído.

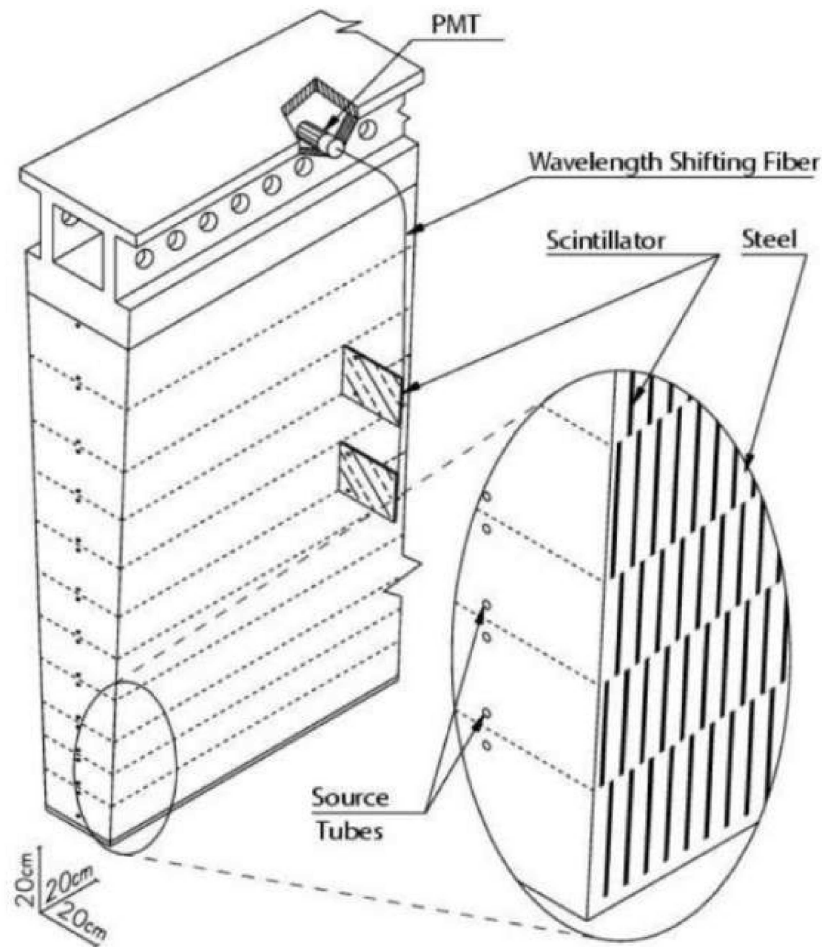
Após a fotoconversão e amplificação, os sinais são encaminhados ao circuito conformador. Este estágio é responsável por padronizar a forma do pulso analógico, filtrando componentes indesejadas de alta frequência e ajustando a largura temporal para compatibilidade com a janela de amostragem do sistema de leitura (GONÇALVES, 2021). A padronização garante que a amplitude do pulso seja proporcional à energia depositada e que o tempo de crista seja reprodutível, facilitando a aplicação de algoritmos de reconstrução como o *Optimal Filtering*. Essa uniformidade é especialmente relevante para as rotinas de calibração e monitoramento, que periodicamente verificam a integridade óptica, a resposta dos PMTs e a estabilidade eletrônica do canal.

Considerando essa necessidade de uniformidade, cada célula do TileCal é composta por um arranjo de telhas cintiladoras acopladas a dois PMTs, posicionados em lados opostos, o que proporciona redundância e maior robustez na leitura. Cada PMT representa um canal de leitura individual, totalizando aproximadamente 10.000 canais em todo o detector (MLYNARÍKOVÁ *et al.*, 2017).

A estrutura modular do TileCal permite segmentação longitudinal e radial eficientes, otimizando a cobertura angular e facilitando tanto o processo de manutenção quanto de calibração. A Figura 9 apresenta uma visualização tridimensional de um módulo típico do TileCal, evidenciando a disposição das camadas de absorvedor e cintilador ao longo da direção perpendicular ao feixe de partículas.

O sinal característico gerado em cada canal, após passar pelo conformador, é digitalizado em 7 amostras espaçadas de 25 ns, perfazendo um tempo total de aproximadamente

Figura 9 – Visualização de um módulo típico do TileCal



Fonte: (ATLAS COLLABORATION, 2008)

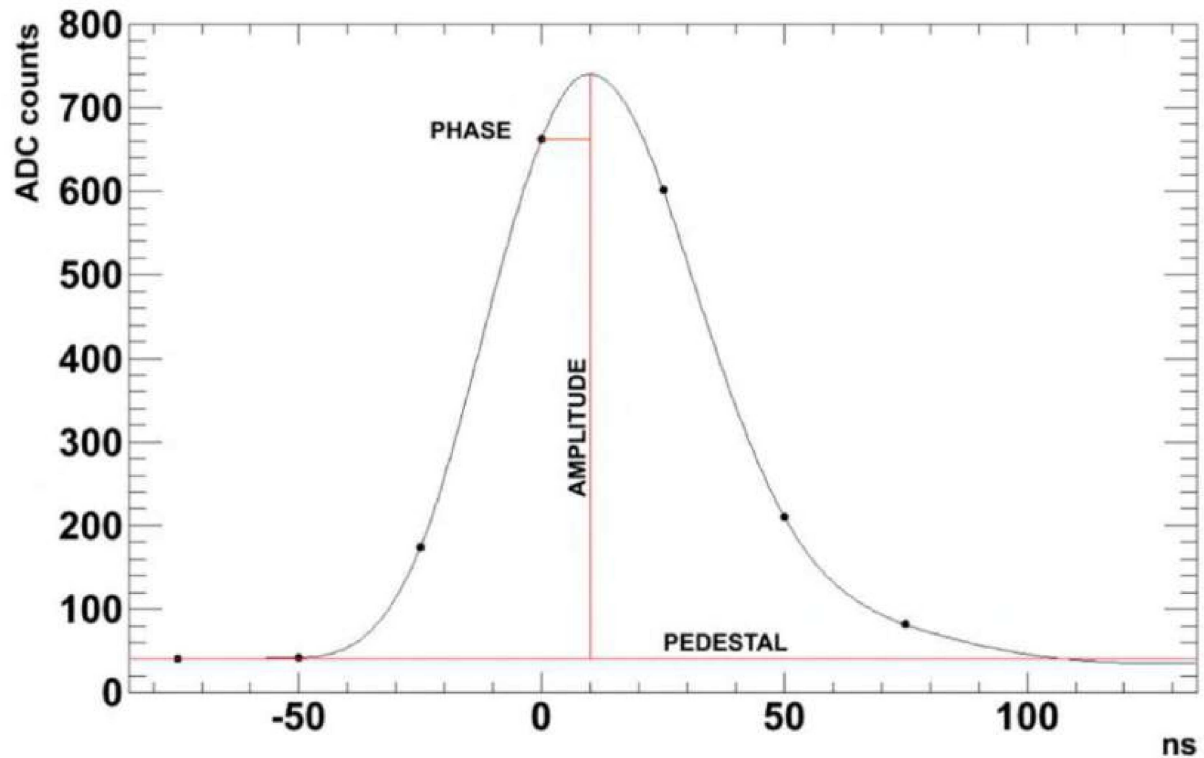
150 ns. A taxa de amostragem de 40 MHz é sincronizada com a frequência das colisões do LHC, conforme ilustrado na Figura 10.

2.5.2 Sistema de Calibração e Monitoramento

A operação estável e precisa do TileCal depende de um sistema de calibração e monitoramento capaz de compensar variações de ganho, degradação óptica, envelhecimento de componentes e efeitos de radiação ao longo do tempo. Para isso, o detector conta com um conjunto integrado de técnicas complementares que permitem avaliar tanto a cadeia óptica quanto a eletrônica de leitura, desde o cintilador até a digitalização final do sinal.

- **Calibração com fonte de Césio-137:** uma fonte radioativa selada percorre, por meio de tubos hidráulicos, todas as células calorimétricas. Os fótons gama emitidos interagem uniformemente com o material ativo, permitindo verificar a resposta absoluta do sistema óptico e dos fotomultiplicadores. Essa calibração fornece uma referência direta para a uniformidade entre canais e detecta variações lentas na eficiência de coleta de luz (ATLAS COLLABORATION, 2010).

Figura 10 – Formato do pulso do TileCal



Fonte: (PERALVA, 2015)

- **Pulso de laser:** utilizado para monitorar rapidamente mudanças no ganho dos PMTs e na transparência das fibras ópticas. O sistema de laser gera pulsos de luz de curta duração e intensidade controlada, distribuídos simultaneamente para todos os canais. Essa técnica é executada com alta frequência, inclusive durante pausas entre feixes, permitindo o acompanhamento quase em tempo real das condições do detector (ATLAS COLLABORATION, 2017).
- **Sistema de injeção de carga:** injeta pulsos elétricos calibrados diretamente na eletrônica de leitura, contornando o sistema óptico. Isso possibilita a avaliação da linearidade, estabilidade e resposta dinâmica da cadeia eletrônica, além de facilitar a identificação de canais com mau funcionamento eletrônico ou variações de ganho independentes do sistema óptico (ATLAS COLLABORATION, 2014).
- **Calibração com feixes de partículas:** realizada durante períodos específicos de comissionamento ou testes, utiliza feixes controlados de partículas conhecidas (como múons de teste) para validar a resposta do detector. Os resultados são comparados com previsões obtidas por simulação de Monte Carlo, garantindo consistência entre a resposta medida e o modelo físico esperado (ATLAS COLLABORATION, 2013).

Cada método atua sobre uma parte específica da cadeia de detecção, mas a

integração entre eles é o que assegura o desempenho global do TileCal. A fonte de Césio garante a calibração absoluta e homogênea; o sistema de laser detecta variações rápidas no ganho; a injeção de carga isola a resposta da eletrônica; e os feixes de teste fornecem uma validação física independente.

Além disso, os dados de calibração são utilizados de forma contínua nos algoritmos de reconstrução de energia. Correções de ganho e fatores de equalização são aplicados ainda na etapa de processamento inicial, garantindo que o *trigger* e o armazenamento dos eventos sejam baseados em medidas confiáveis. Esse monitoramento constante também permite identificar, com antecedência, degradações de componentes, possibilitando intervenções de manutenção preventiva e reduzindo o risco de perda de dados físicos relevantes.

Em conjunto, o sistema de calibração e monitoramento do TileCal assegura que a resposta do detector permaneça uniforme e estável ao longo de toda a operação, mesmo diante de variações ambientais e das condições extremas do LHC. Dessa forma, é possível preservar a qualidade dos dados desde a primeira etapa de aquisição, mantendo a integridade das medições para análise física subsequente.

2.5.3 Sistema de Trigger e Fluxo de Dados

No ambiente do LHC, onde dezenas de milhões de colisões de prótons ocorrem a cada segundo, é inviável registrar todos os eventos produzidos. Para lidar com esse desafio, o experimento ATLAS conta com um sistema de *trigger* capaz de identificar, em tempo real, as interações potencialmente mais interessantes e descartar o grande volume de eventos não relevantes. O objetivo é reduzir a taxa de dados de 40 MHz para valores compatíveis com as capacidades de armazenamento e processamento, preservando a integridade das informações essenciais para a análise física. Essa filtragem é especialmente importante para subsistemas como o TileCal, que gera medições contínuas de energia a cada colisão e precisa entregar respostas rápidas para alimentar o processo decisório do *trigger* de hardware.

O sistema de *trigger* do ATLAS, em sua concepção original implementada no Run 1 do LHC, já operava com uma arquitetura hierárquica de dois níveis (HALLER, 2005). O primeiro nível (*Level-1*), implementado em hardware dedicado, executava a filtragem inicial com base em informações dos calorímetros e do sistema de múons, reduzindo a taxa de 40 MHz para cerca de 75 kHz. O segundo nível, denominado *High Level Trigger* (HLT), processava os eventos aceitos pelo *Level-1* utilizando uma fazenda de servidores, aplicando algoritmos de reconstrução mais completos e cortando a taxa para aproximadamente 200 Hz de eventos gravados para análise *offline*. Essa arquitetura serviu como base para as melhorias implementadas nos períodos operacionais subsequentes.

Durante o Run 2 e o Run 3, o sistema de *trigger* foi implementado em dois níveis principais que operam de forma hierárquica e integrada, combinando rapidez na

filtragem inicial com sofisticação na análise subsequente (ATLAS COLLABORATION, 2020; BARRUÉ, 2024):

- **Nível 1 (L1) – Hardware:** executa a primeira etapa de seleção utilizando eletrônica dedicada, baseada principalmente em FPGAs e ASICs de alta velocidade. Essa fase processa sinais vindos de diferentes subsistemas, como os calorímetros (TileCal e LAr) e o sistema de múons, identificando padrões de interesse como jatos hadrônicos energéticos, fótons isolados ou múons de alto momento transversal. No caso do TileCal, as energias reconstruídas são agregadas em *trigger towers*, que resumem a informação espacial e energética de regiões do detector, permitindo decisões rápidas. Essa filtragem reduz a taxa de 40 MHz para cerca de 100 kHz, aplicando cortes em energia depositada, topologia do evento e pseudorapidez.
- **Nível Alto (HLT – do inglês, *High Level Trigger*) – Software:** processa os eventos aceitos pelo L1 em uma fazenda de servidores de alto desempenho, executando algoritmos mais sofisticados, capazes de realizar reconstruções parciais ou completas de pistas, jatos e variáveis globais como o momento transversal faltante (*MET*). Aqui, a informação completa do detector é utilizada, incluindo calibrações dinâmicas e correções para *pile-up*. A taxa de saída é reduzida para aproximadamente 1 kHz, correspondente ao número de eventos armazenados para análise *offline*.

A partir de 2030, com a entrada em operação do *High-Luminosity LHC* (HL-LHC) (EINSWEILER, 2017), o cenário experimental mudará drasticamente, com luminosidade instantânea muito maior e taxas de *pile-up* atingindo até 200 interações por cruzamento. Esse aumento impõe a necessidade de uma reformulação profunda do sistema de *trigger*, tanto em termos de arquitetura quanto de capacidade de processamento:

- **Nível 0 (L0) – Hardware:** novo estágio inicial de filtragem, com latência expandida para até 10 μ s e maior granularidade nos dados de entrada (FILIMONOV *et al.*, 2020). Essa etapa reduzirá a taxa de 40 MHz para aproximadamente 1 MHz, aproveitando não apenas informações calorimétricas mais detalhadas, mas também dados preliminares do rastreador interno (*Inner Tracker*). Para o TileCal, isso significa fornecer medições de energia de alta resolução e já pré-processadas, com correções rápidas aplicadas no próprio hardware (HEINRICH, 2021).
- **Nível 1 (L1) – Hardware:** receberá os eventos selecionados pelo L0 e aplicará algoritmos ainda mais sofisticados, com uso extensivo de FPGAs de última geração e lógica programável capaz de executar reconstruções parciais diretamente no hardware (CERVELLÓ, 2022). Aqui, a taxa será reduzida para aproximadamente 400 kHz, e os cortes poderão considerar informações combinadas de múltiplos subsis-

temas, incluindo a associação direta de depósitos de energia no TileCal com trilhas reconstruídas no rastreador.

- **Event Filter (EF) – Software:** substituirá o HLT atual, operando com acesso integral aos dados completos do evento. O EF executará algoritmos avançados de física de alto nível, incluindo reconstruções detalhadas e técnicas sofisticadas de rejeição de *pile-up*. Essa última etapa reduzirá a taxa final para cerca de 10 kHz, permitindo armazenar um número significativamente maior de eventos de interesse em comparação com o sistema atual (HEINRICH, 2021).

A transição de dois para três estágios no *trigger* representa um avanço significativo, pois amplia a granularidade e o nível de detalhes disponíveis já nas primeiras decisões (FILIMONOV *et al.*, 2020). Além disso, a expansão da latência permitida possibilitará a execução de algoritmos mais complexos no próprio hardware, aproveitando ao máximo o potencial de processamento paralelo de FPGAs modernos. Para o TileCal, essa evolução implica desafios técnicos adicionais: será necessário otimizar a leitura e o processamento de sinais para fornecer dados de alta precisão em janelas temporais extremamente curtas, mantendo a estabilidade e a linearidade da resposta mesmo sob condições de ocupação elevadíssimas.

O aumento do número de eventos gravados acarretará em um crescimento expressivo no volume anual de dados. Globalmente, estima-se que os recursos de armazenamento do ATLAS ultrapassem 1–2 EB em disco e mais de 2 EB em fita por volta de 2031, com ambos dobrando até 2041, refletindo o impacto direto da maior taxa de eventos e da complexidade adicional imposta pela sobreposição extrema (ATLAS COLLABORATION, 2025). Isso exigirá não apenas maior capacidade de armazenamento e rede, mas também sistemas de reconstrução *offline* mais robustos, capazes de lidar com a intensificação do *pile-up* e de aplicar correções precisas de energia em ambientes altamente congestionados.

Assim, a evolução do sistema de *trigger* no HL-LHC não se trata apenas de lidar com um maior fluxo de eventos, mas também de antecipar e mitigar, já nas primeiras etapas de seleção, os efeitos adversos do *pile-up* extremo. Essa integração entre filtragem em tempo real e reconstrução precisa será determinante para garantir que o TileCal continue a desempenhar um papel central na identificação de fenômenos raros e na manutenção da qualidade das medidas, preparando o caminho para os desafios apresentados na próxima seção.

2.5.4 Pulso de Saída e Desafios do *Pile-up*

A operação do sistema de *trigger* no ATLAS está intimamente ligada à qualidade dos sinais gerados pelos seus subdetectores. Entre os fatores que mais degradam essa qualidade, destaca-se o *pile-up*, um fenômeno característico de ambientes de altíssima

luminosidade. Ele ocorre quando sinais provenientes de diferentes colisões de prótons se sobrepõem temporalmente dentro da mesma janela de leitura, produzindo um pulso composto cuja forma não corresponde a nenhum evento isolado, mas sim à soma de várias contribuições energéticas.

No TileCal, essa situação é particularmente relevante devido à sua resposta temporal relativamente longa, com pulsos que se estendem por cerca de 150 ns. Em um acelerador como o LHC, no qual os feixes se cruzam a cada 25 ns (40 MHz), isso significa que um único pulso pode conter sobreposição de até seis interações consecutivas. Essa superposição ocorre antes mesmo da digitalização, de forma que a informação bruta já chega ao sistema de aquisição contaminada por contribuições de múltiplos eventos.

Durante o Run 2 e Run 3, o número médio de interações simultâneas por cruzamento (*pile-up médio* $\langle\mu\rangle$) variou entre aproximadamente 20 e 60. Na Fase 2 do HL-LHC, prevista para iniciar em 2030, estima-se que esse valor possa atingir a faixa de 140 a 200, estabelecendo um cenário inédito em termos de densidade de eventos e complexidade de reconstrução. O impacto direto no TileCal se traduz na distorção de parâmetros fundamentais do pulso, como:

- **Amplitude:** proporcional à energia depositada, podendo ser artificialmente aumentada pela contribuição de eventos secundários.
- **Tempo de pico:** deslocado devido à soma de sinais com atrasos distintos, afetando a reconstrução temporal.
- **Largura temporal:** ampliada pela sobreposição, reduzindo a capacidade de separar pulsos próximos.

Essas alterações levam a erros sistemáticos na estimativa de energia e a uma maior incerteza na identificação do evento principal. As regiões de maior ocupação do detector — em especial próximas a $\eta \approx 0$, onde a densidade de partículas por colisão é elevada — são as mais suscetíveis a esse efeito.

Do ponto de vista matemático, o sinal observado $S(t)$ pode ser representado como:

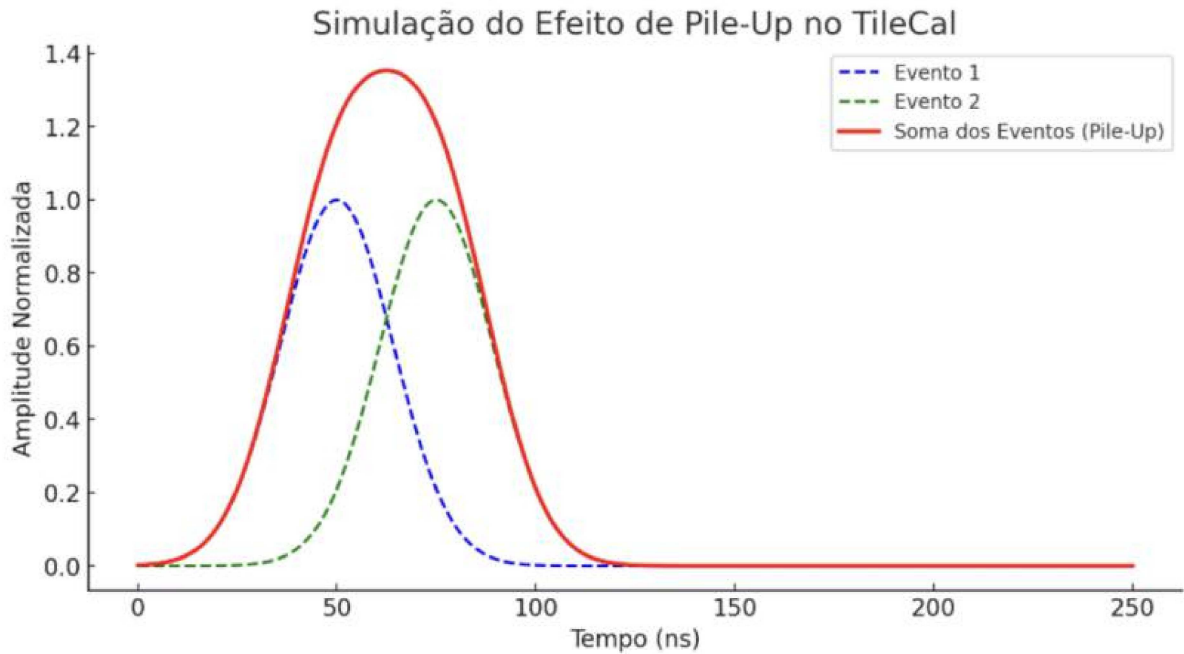
$$S(t) = \sum_{i=1}^N A_i \cdot P(t - t_i), \quad (2.2)$$

em que A_i representa a amplitude proporcional à energia de cada colisão, t_i o atraso relativo de chegada e $P(t)$ a forma de pulso de um evento isolado. Em condições ideais ($N = 1$), a reconstrução é direta. Entretanto, à medida que N cresce — como ocorrerá no HL-LHC —, a separação das contribuições individuais exige métodos de deconvolução e filtragem mais avançados.

A Figura 11 ilustra esse efeito: dois sinais independentes (curvas azul e verde) se sobrepõem, resultando em um pulso composto (curva vermelha) com amplitude maior

e pico deslocado, mascarando as características originais de cada evento. Esse tipo de distorção impacta não apenas a reconstrução de jatos, mas também a determinação precisa do momento transversal faltante (MET), crucial para buscas de nova física.

Figura 11 – Exemplo do efeito de *pile-up*.



Fonte: Elaborado pelo autor (2025).

O desafio torna-se ainda maior porque o *trigger* de hardware — especialmente o L0 e o L1 introduzidos na Fase 2 — precisará tomar decisões rápidas com base em sinais já afetados pelo *pile-up*, antes que o processamento detalhado do *Event Filter* seja realizado. Por isso, estratégias de mitigação devem estar presentes desde as etapas iniciais do processamento, incluindo:

- **Eletrônica de leitura de alta precisão:** módulos com maior taxa de amostragem e menor ruído eletrônico, permitindo discriminar pulsos separados por poucos nanossegundos.
- **Reconstrução com *Optimal Filtering* expandido:** uso de mais coeficientes e incorporação de parâmetros temporais explícitos na estimativa da energia.
- **Deconvolução adaptativa:** algoritmos que ajustam o modelo do pulso às condições instantâneas de ocupação.
- **Técnicas de *machine learning*:** redes neurais treinadas para reconhecer padrões de sobreposição e separar sinais individuais.

- **Integração com informações do rastreador:** correlação espaço-temporal entre trajetórias reconstruídas e depósitos de energia calorimétrica.

Na Fase 2, a mitigação bem-sucedida do *pile-up* será determinante para manter a relevância científica do TileCal. Preservar a resolução de energia e a precisão temporal permitirá que o detector continue desempenhando seu papel central na física de hádrons do ATLAS, viabilizando a observação de fenômenos raros mesmo em meio a um fundo estatisticamente dominante.

Essa necessidade de desempenho constante, mesmo sob condições extremas, reforça a importância dos sistemas de calibração e monitoramento do TileCal, que garantem a uniformidade de resposta e a estabilidade das medições ao longo de toda a operação, e que serão discutidos na próxima subseção.

2.5.5 Considerações Finais sobre o TileCal

O desempenho do TileCal no ATLAS resulta de uma combinação equilibrada entre robustez mecânica, precisão na leitura e integração eficiente com o sistema de aquisição e *trigger*. Desde o início das operações do LHC, o calorímetro desempenha papel central na medição de energia de hádrons, contribuindo para a reconstrução de jatos, determinação do momento transversal faltante (MET) e identificação de processos raros. A sua arquitetura modular, aliada a um sistema de leitura redundante com aproximadamente 10.000 canais, garante alta confiabilidade mesmo em condições de operação contínua e intensa, assegurando que dados críticos estejam sempre disponíveis para a tomada de decisão em tempo real.

A transição para o HL-LHC introduz, contudo, um cenário operacional sem precedentes. A intensificação da sobreposição de dados, associada ao aumento das taxas de eventos, impõe requisitos adicionais de desempenho. A resolução de energia, métrica para avaliar performance, é descrita por:

$$\frac{\sigma(E)}{E} = \frac{50\%}{\sqrt{E}} \oplus 3\%, \quad (2.3)$$

Além disso, é necessário também ampliar a precisão temporal, melhorar a rejeição de ruído e reforçar a integração com outros subsistemas do detector. Neste novo regime, a fronteira entre aquisição e processamento se torna mais estreita: decisões de *trigger* em hardware deverão considerar dados calorimétricos já corrigidos para efeitos de empilhamento, enquanto a reconstrução *offline* dependerá de algoritmos capazes de separar, com alta fidelidade, sinais sobrepostos de múltiplos eventos.

As atualizações previstas para a Fase 2 abrangem toda a cadeia operacional do TileCal. Entre elas, destacam-se a adoção de eletrônica de leitura com maior largura de banda e resolução temporal, o desenvolvimento de algoritmos de filtragem e deconvolução

adaptativa, o uso intensivo de técnicas de *machine learning* treinadas em dados reais e simulados, e a integração plena com as informações do rastreador para correlação espaço-temporal mais precisa. Essas melhorias não visam apenas manter o desempenho atual, mas reposicionar o TileCal como um elemento ativo no processo de seleção e interpretação dos eventos mais relevantes para a física de altas energias.

Assim, a continuidade do sucesso científico do ATLAS na próxima década dependerá diretamente da capacidade do TileCal de evoluir frente ao desafio imposto pelo HL-LHC. Ao combinar hardware de última geração, processamento paralelo em FPGAs e métodos avançados de reconstrução de sinais, o detector permanecerá apto a fornecer medições precisas e confiáveis, sustentando descobertas que poderão ultrapassar os limites do Modelo Padrão e consolidando seu papel como um dos pilares da instrumentação de partículas no CERN.

3 FERRAMENTAS DE SIMULAÇÃO E DESENVOLVIMENTO

Nesse capítulo serão abordadas as ferramentas necessárias que foram utilizadas no desenvolvimento dessa dissertação.

3.1 SAPHO

O desenvolvimento de sistemas embarcados com requisitos específicos de desempenho, área e consumo de energia tem impulsionado o uso de processadores *soft-core*, especialmente em plataformas reconfiguráveis como os FPGAs. Tradicionalmente, esses processadores possuem uma arquitetura fixa e generalista, o que implica na alocação de uma quantidade padronizada de recursos de hardware, independentemente da complexidade do programa a ser executado. Tal abordagem frequentemente resulta em subutilização de recursos, aumento do consumo e ineficiência na síntese .

Nesse contexto, o SAPHO (*Scalable Architecture Processor for Hardware Optimization*) surge como uma alternativa inovadora. Trata-se de um processador *soft-core* de código aberto projetado para realizar a alocação automática e personalizada de recursos de hardware durante a compilação (NIPS, 2025). Sua principal motivação está na possibilidade de adaptar dinamicamente sua arquitetura interna às necessidades do programa, otimizando a utilização dos recursos lógicos da FPGA e reduzindo significativamente o overhead estrutural.

Ao contrário de processadores convencionais como o NIOS II, cuja parametrização é limitada a conjuntos pré-definidos de configurações, o SAPHO propõe um modelo no qual tanto a arquitetura quanto os blocos funcionais internos são gerados de forma direcionada com base nas instruções utilizadas no código fonte (AGUIAR, 2023). Essa abordagem confere ao processador um grau elevado de escalabilidade e eficiência, tornando-o especialmente útil em aplicações que demandam customização em nível de hardware.

Por se tratar de uma arquitetura orientada à otimização, o SAPHO encontra aplicação em ambientes onde o equilíbrio entre flexibilidade de programação e eficiência estrutural é essencial, como em algoritmos de processamento de sinais, controle em tempo real e sistemas de aquisição de dados. Sua estrutura modular e adaptável constitui o ponto de partida para as seções seguintes, nas quais serão explorados com mais profundidade seus aspectos arquiteturais, operacionais e de desenvolvimento.

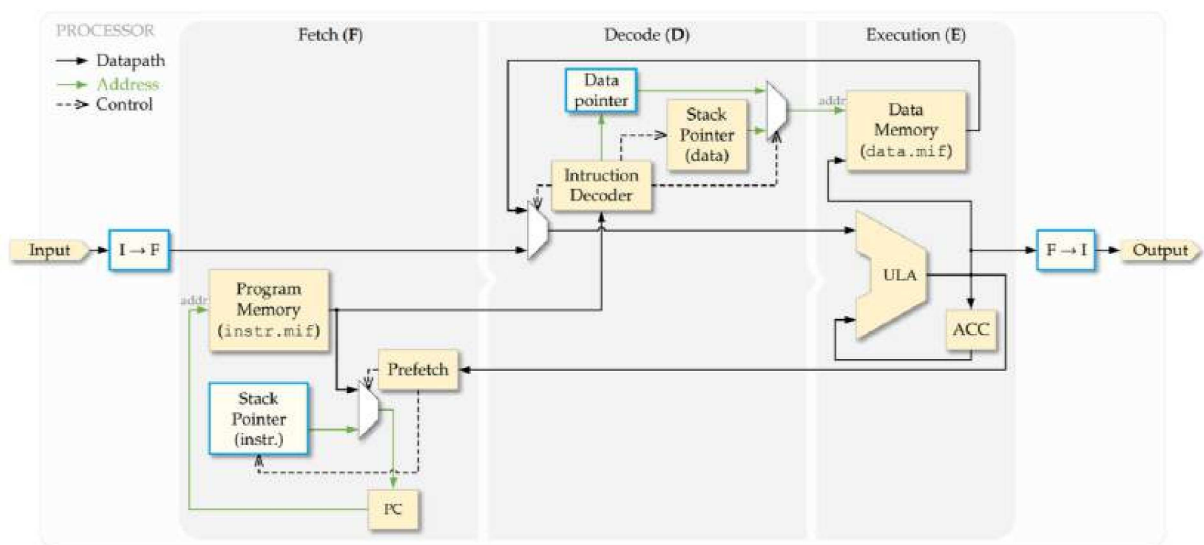
3.1.1 Arquitetura de Hardware do Processador

Dando continuidade à proposta de um processador flexível e otimizado, a arquitetura do SAPHO foi concebida com base em uma organização Harvard, utilizando um conjunto reduzido de instruções (*Reduced Instruction Set Computer* – RISC), permitindo simplicidade e eficiência tanto na lógica de controle quanto no uso dos recursos lógicos da

FPGA. Essa arquitetura não é fixa: os blocos de hardware são instanciados dinamicamente durante a compilação, conforme a análise do código-fonte embarcado, o que torna o processador altamente customizável e adaptável à aplicação-alvo (NIPS, 2025).

O processador é dividido em blocos funcionais principais, entre os quais se destacam a Unidade Lógica e Aritmética (ULA), as memórias de dados e de instruções, os registradores auxiliares, e os circuitos de controle de fluxo, como o contador de programa (*Program Counter*) e os ponteiros de pilha. (NIPS, 2025) A Figura 12 ilustra essa organização, destacando em azul os módulos que são gerados apenas quando demandados pelas instruções do programa. Tal característica permite uma redução significativa na ocupação de área lógica e na complexidade do roteamento interno da FPGA.

Figura 12 – Diagrama de blocos do SAPHO.



Fonte: (NIPS, 2025)

O fluxo de execução é segmentado em três estágios de pipeline: *fetch*, *decode* e *execute*. Essa segmentação permite a execução de uma instrução por ciclo de clock, sem penalizações mesmo em estruturas de controle como desvios condicionais ou chamadas de sub-rotina, graças ao uso de uma ULA combinacional. Além disso, o SAPHO inclui uma pilha de dados e uma pilha de instruções, ambas compartilhadas nas memórias principais, e controladas por ponteiros dedicados, que são instanciados conforme a necessidade de suporte a recursividade ou sub-rotinas.

Outro diferencial importante na arquitetura está na possibilidade de configuração da ULA para operar com aritmética de ponto fixo ou ponto flutuante. No caso desta última, módulos de conversão são automaticamente inseridos nos barramentos de entrada e saída, de modo a garantir compatibilidade com periféricos ou sistemas que operem com representação em complemento de dois. Essa flexibilidade aritmética é especialmente relevante em aplicações que envolvem cálculo intensivo ou requisitos de precisão variável.

Ainda com base na Figura 12, é possível destacar diversos blocos funcionais importantes para a operação do SAPHO:

1. **Conversores Int-Float e Float-Int:** Quando a ULA é configurada para trabalhar com números em ponto flutuante, são automaticamente incluídos nos barramentos circuitos responsáveis por converter valores entre ponto fixo e ponto flutuante. O SAPHO adota um formato próprio de ponto flutuante, projetado especificamente para otimizar uso de recursos em FPGAs.
2. **Memórias:** A arquitetura do SAPHO permite que as memórias de dados e instruções sejam ajustadas dinamicamente de acordo com o tamanho do código fonte. O Assembler realiza uma análise completa do programa e define, de forma automática, a quantidade exata de endereços necessários. Além disso, o conteúdo dessas memórias é exportado em arquivos do tipo `.mif`, utilizados posteriormente na implementação física em FPGA.
3. **Prefetch:** Esse bloco atua realizando a busca antecipada da próxima instrução enquanto a atual ainda está em execução. Ele também realiza a separação entre o código da operação (opcode) e os operandos, além de controlar o decodificador de instruções e o contador de programa.
4. **Program Counter (PC):** Responsável por apontar para a instrução corrente na memória de programa, esse bloco é incrementado automaticamente a cada instrução executada. Em instruções de desvio (como saltos e chamadas), ele assume um valor específico calculado pelo decodificador. Sua largura em bits é determinada pelo tamanho total da memória de instruções, sendo configurada automaticamente pelo Assembler.
5. **Stack Pointer (SP):** O SAPHO pode instanciar dois ponteiros de pilha distintos: um para a pilha de dados e outro para a pilha de instruções. Esses blocos apontam para os endereços mais altos das memórias correspondentes e são utilizados em instruções como `PUSH`, `POP`, `CALL` e `RETURN`. A pilha de instruções só é gerada caso o compilador detecte o uso de sub-rotinas no código.
6. **Register File:** Caso o programa contenha vetores ou arrays, o SAPHO instanciará automaticamente um bloco de registradores que permite indexação por deslocamento. Isso possibilita o acesso eficiente a qualquer posição do array com base em um índice.
7. **Unidade Lógico-Aritmética (ULA):** A ULA do SAPHO é altamente parametrizável. Seu funcionamento é adaptado de acordo com as instruções que o compilador gera para a aplicação em questão. Além de opções configuráveis como largura de dados e tipo de aritmética (ponto fixo ou ponto flutuante), o Assembler ajusta automaticamente os recursos internos para otimizar o desempenho e o uso de lógica.

Esses blocos funcionais refletem o caráter adaptável da arquitetura SAPHO, sendo parte fundamental da geração automática de hardware eficiente para aplicações específicas.

Ao possibilitar que recursos como multiplicadores, comparadores e conversores sejam gerados apenas quando utilizados, o SAPHO assegura uma implementação altamente otimizada para o conjunto específico de instruções de cada aplicação. Os detalhes desses blocos, bem como os mecanismos de parametrização automática, serão aprofundados nas próximas subseções.

3.1.2 Unidade Lógica-Aritmética e Recursos Dinâmicos

A Unidade Lógica-Aritmética (ULA) do SAPHO representa um dos blocos mais versáteis de sua arquitetura, sendo responsável pela execução das instruções aritméticas e lógicas presentes no programa embarcado. Diferentemente de processadores soft-core convencionais, nos quais a microarquitetura da ULA é fixa, o SAPHO adota uma abordagem parametrizável que permite instanciar dinamicamente apenas os circuitos necessários com base nas instruções identificadas durante o processo de compilação.

Essa característica garante não apenas uma economia significativa de recursos lógicos na FPGA, como também contribui para uma maior eficiência energética e ocupação reduzida da área lógica. A ULA pode ser configurada para operar tanto com aritmética de ponto fixo quanto de ponto flutuante, sendo esta última implementada por meio de um formato customizado que otimiza as operações típicas de sistemas embarcados.

Entre os recursos implementáveis na ULA, destacam-se operações como multiplicação, divisão, operações lógicas bit a bit, comparações, inversões e deslocamentos. Além disso, o processador é capaz de reconhecer instruções que envolvem controle de fluxo e acesso indireto à memória, adaptando a geração de hardware para contemplar pilhas de instrução e recursos de endereçamento específicos.

A Tabela 1 apresenta a correspondência entre instruções presentes no código e os blocos lógicos que são automaticamente instanciados pelo *Assembler* durante o processo de geração do processador. A partir da configuração escolhida — incluindo o tipo de aritmética (ponto-fixa ou ponto-flutuante) e os recursos necessários — é possível identificar quais componentes internos da ULA serão sintetizados. Algumas das instruções indicadas em negrito representam extensões e melhorias desenvolvidas neste trabalho, cujo funcionamento será aprofundado na Seção 5.1.

Tabela 1 – Instruções e circuitos gerados automaticamente pelo Assembler, com suas respectivas ativações na ULA e modos de operação.

Instrução	Circuito	ULA	P. Fixo	P. Flut.
DIV	Divisão	X	X	X
OR	Ou bit a bit	X	X	
LOR	Ou lógico	X	X	X
GRE	Maior que	X	X	X
MOD	Resto da divisão	X	X	
MLT	Multiplicação	X	X	X
LES	Menor que	X	X	X
EQU	Igual a	X	X	X
AND	And bit a bit	X	X	
LAN	And lógico	X	X	X
INV	Inversor bit a bit	X	X	
LIN	Inversor lógico	X	X	X
SHR	Shift para direita	X	X	
SHL	Shift para esquerda	X	X	
SRS	Shift com sinal	X	X	
CALL	Pilha de instrução		X	X
SRF	Endereçamento indireto		X	X
PSET	Set se for positivo		X	X
NORM	Normalização	X	X	
ABS	Valor absoluto	X	X	X
SIGN	Sinalização	X		X

Fonte: (AGUIAR, 2023)

Essa abordagem de alocação sob demanda é conduzida pelo compilador Assembler, que interpreta o código gerado em linguagem Assembly e determina exatamente quais unidades funcionais devem ser instanciadas, incluindo a lógica de controle, os circuitos da ALU e os módulos auxiliares para conversão de formato numérico. Dessa forma, o processador torna-se altamente adaptável às necessidades de cada aplicação, mantendo o desempenho de execução em um ciclo de clock por instrução, mesmo em estruturas com desvios condicionais.

Essa estratégia de personalização do hardware durante o processo de síntese representa uma evolução significativa em relação às arquiteturas genéricas, alinhando-se à proposta de otimização de recursos e escalabilidade do SAPHO.

3.1.3 Organização de Memória e Controle

A estrutura de memória do SAPHO foi concebida para garantir flexibilidade e eficiência na utilização dos recursos lógicos do hardware. O modelo adotado permite que tanto a memória de dados quanto a de programa sejam ajustadas automaticamente com base no conteúdo do código a ser executado, eliminando desperdícios por superdimensionamento.

Essa abordagem torna possível a adaptação do sistema a aplicações com diferentes níveis de complexidade, mantendo sempre a utilização mínima necessária de registradores e blocos de memória.

O controle da execução é realizado por blocos especializados que coordenam a leitura e escrita nos endereços apropriados. O contador de programa, configurável de acordo com o tamanho da memória instruída, realiza o apontamento para a próxima operação a ser executada e pode ser redirecionado quando instruções de salto são detectadas. Essa lógica garante fluidez no fluxo de instruções e suporte a desvios condicionais.

Complementando esse mecanismo, o SAPHO incorpora pilhas dedicadas tanto para variáveis temporárias quanto para o armazenamento de endereços de retorno, quando há chamadas de sub-rotinas. Essas pilhas ocupam regiões superiores das memórias e são manipuladas por ponteiros que regulam sua profundidade. A estrutura de pilha de instruções, por sua vez, é criada apenas quando necessária, otimizando ainda mais o consumo de recursos.

A existência de blocos como o *Prefetch*, responsável por adiantar a leitura da próxima instrução enquanto a atual ainda está em execução, reflete a preocupação com a eficiência do fluxo de dados no processador. Esse mecanismo também atua na separação entre opcode e operandos, otimizando o decodificador e a lógica sequencial.

A arquitetura foi projetada para permitir expansibilidade e modularidade, favorecendo a reutilização de blocos e facilitando o desenvolvimento de sistemas multicore. Isso é possível graças à padronização das conexões e ao controle rigoroso dos ciclos de leitura e escrita, que mantêm a consistência da execução mesmo em cenários mais complexos.

3.1.4 Arquitetura de Pipeline

A arquitetura de execução do SAPHO é baseada em um pipeline de três estágios, permitindo a sobreposição de instruções e a obtenção de alto rendimento com mínimo controle adicional. Esse modelo, tradicionalmente adotado em arquiteturas RISC, é explorado de forma eficiente devido à ausência de unidades com latência variável ou estruturas complexas de reordenação.

Durante a execução, os estágios de busca, decodificação e execução são organizados de maneira a garantir que uma nova instrução possa ser iniciada a cada ciclo de clock, independentemente do tipo de operação em andamento. Tal comportamento é viabilizado, principalmente, pelo fato de a ULA operar de forma combinacional, evitando a necessidade de ciclos adicionais para completar as operações aritméticas e lógicas.

Outro aspecto relevante é a estabilidade do pipeline mesmo na presença de desvios condicionais. Diferentemente de arquiteturas que exigem mecanismos de predição de saltos ou preenchimento de bolhas, o SAPHO lida com chamadas e retornos de sub-rotinas por meio de pilhas específicas, mantendo o fluxo contínuo sem penalidades estruturais. A

lógica de controle do contador de programa é ajustada dinamicamente conforme o fluxo de instruções, garantindo que o pipeline siga em execução contínua.

A interação entre os blocos de controle, como o registrador acumulador e os decodificadores, com os módulos de memória e ULA, é coordenada por sinais sincronizados com o clock global, assegurando que os dados corretos estejam disponíveis no estágio apropriado. Essa organização permite ao SAPHO manter simplicidade estrutural e eficiência temporal, mesmo em aplicações embarcadas com exigência de resposta rápida e determinística.

3.1.5 Representação de Ponto Flutuante Customizada

Embora o padrão IEEE 754 seja amplamente utilizado para representar números em ponto flutuante em sistemas computacionais, ele impõe um alto custo em termos de recursos lógicos e latência, principalmente quando implementado em FPGAs. Tal padrão foi concebido para aplicações generalistas, incluindo o tratamento de casos especiais como *NaN* (Not a Number), infinitos e valores subnormais. No entanto, tais recursos são raramente necessários em aplicações embarcadas específicas, como aquelas voltadas à reconstrução de sinais físicos em experimentos de altas energias.

Para contornar essa limitação e otimizar a síntese em hardware, o SAPHO adota uma representação simplificada e eficiente de números em ponto flutuante. Essa estrutura mantém a lógica conceitual do IEEE 754 — com separação entre sinal, mantissa e expoente —, porém com codificação otimizada para minimizar o consumo de recursos lógicos e facilitar a integração com a ULA combinacional.

A codificação adotada representa a mantissa em módulo (somente valores positivos), enquanto o expoente utiliza representação em complemento de dois, simplificando as operações aritméticas no nível do circuito. Além disso, não há bits dedicados a representar valores especiais, eliminando a necessidade de lógica adicional de verificação.

A equação que rege essa representação pode ser descrita da seguinte forma (NIPS, 2025):

$$x = (-1)^S \times M \times 2^E \quad (3.1)$$

onde S é o bit de sinal, M representa a mantissa normalizada e E o expoente em complemento de dois.

Uma vantagem prática desta estrutura é a flexibilidade na escolha do número de bits da mantissa e do expoente, permitindo ao projetista ajustar o equilíbrio entre precisão numérica e área de hardware. Tal parametrização é feita diretamente na IDE do SAPHO, sendo automaticamente instanciada nos barramentos de entrada e saída caso a ULA esteja configurada para operar em ponto flutuante.

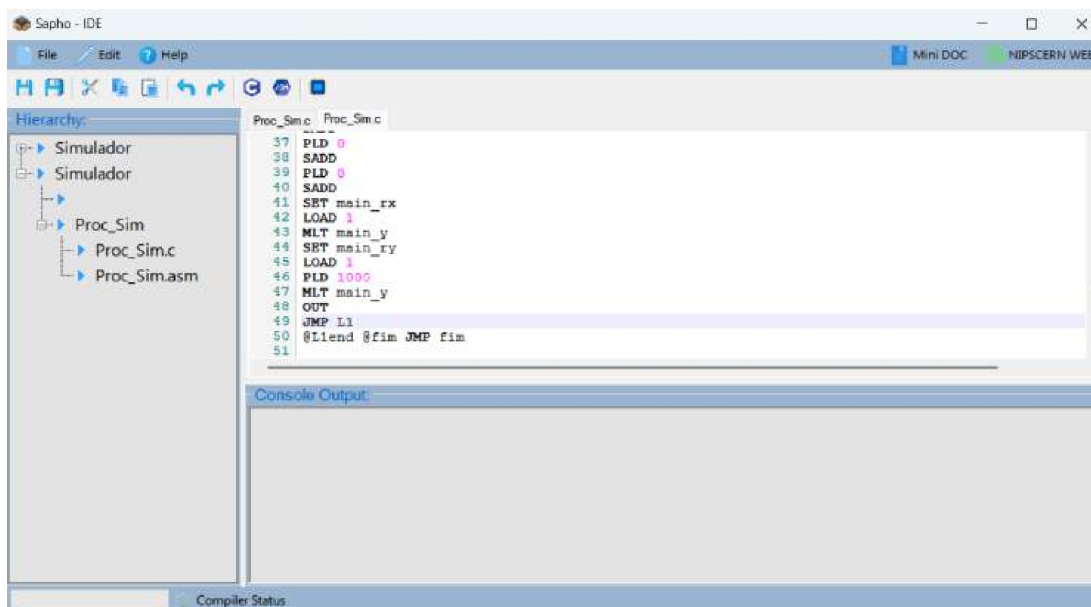
Dessa forma, a representação customizada adotada pelo SAPHO oferece uma alternativa mais enxuta e eficiente para aplicações que não demandam suporte integral ao padrão IEEE, possibilitando maior desempenho e economia de recursos lógicos na implementação final.

3.1.6 Ferramentas de Desenvolvimento

O desenvolvimento de aplicações no SAPHO é facilitado por um conjunto de ferramentas que acompanham o processador, oferecendo suporte completo desde a escrita do código até a geração final dos arquivos em Verilog. Essa abordagem integrada contribui para a usabilidade do sistema, permitindo que usuários com conhecimento limitado em projeto digital possam desenvolver arquiteturas otimizadas com relativa facilidade.

O ambiente de desenvolvimento integrado (IDE) foi construído em linguagem C#, com interface gráfica voltada à simplicidade e clareza na navegação entre arquivos, compiladores e configurações. Nela, é possível gerenciar múltiplos processadores em um mesmo projeto, configurar parâmetros como número de portas I/O, tamanho das pilhas e tipo de aritmética (fixa ou flutuante), além de acompanhar mensagens de compilação e acessar diretamente os códigos-fonte em C++ e Assembly.

Figura 13 – IDE SAPHO.



Fonte: (Elaborado pelo autor (2025).)

A linguagem utilizada na programação é o C++, um subconjunto da linguagem C tradicional. Ela foi projetada para ser simples, evitando elementos complexos como alocação dinâmica de memória ou ponteiros genéricos. Arrays unidimensionais com tamanho fixo são permitidos, bem como sub-rotinas, que são tratadas automaticamente pelo compilador com a inserção de pilhas de instrução, caso necessário. A sintaxe reconhecida cobre

operadores aritméticos básicos, lógicos e relacionais, além de comandos de controle como `if`, `while` e `return`, como pode ser vista na Tabela 2.

Tabela 2 – Funções, palavras-chave e operadores suportados pela linguagem C no SAPHO.

Categoria	Conteúdo
Funções	<code>in()</code> , <code>out()</code>
Palavras-chave	<code>int</code> , <code>float</code> , <code>void</code> , <code>return</code> , <code>while</code> , <code>if</code> , <code>else</code>
Operadores	<code>-</code> , <code>+</code> , <code>*</code> , <code>/</code> , <code><</code> , <code>></code> , <code>!</code> , <code>%</code> , <code>&</code> , <code> </code> , <code>>></code> , <code><<</code> , <code>>=</code> , <code><=</code> , <code>==</code> , <code>!=</code> , <code>&&</code> , <code> </code> , <code>@</code> , <code>/></code>

O compilador **Assembler**, por sua vez, é responsável por transformar o código Assembly gerado pelo compilador C em arquivos necessários à implementação física. Ele produz automaticamente os arquivos `.mif` com os conteúdos das memórias de dados e de programa, além de gerar o código Verilog contendo a descrição parametrizada do processador, incluindo somente os blocos estritamente necessários para a execução do código compilado.

Ao incluir um novo processador em um projeto dentro da IDE do SAPHO, o usuário é direcionado para uma interface chamada *Configuration Wizard*, responsável por reunir todos os parâmetros essenciais que definem o comportamento e a estrutura do processador. Essa interface, exemplificada na Figura 14, permite configurar desde aspectos numéricos até recursos de memória e conectividade do núcleo a ser gerado.

Figura 14 – Parametrização de Processador no SAPHO.

The image shows a 'Configuration Wizard' window with a blue background and white text. It is divided into several sections, each with a title in bold:

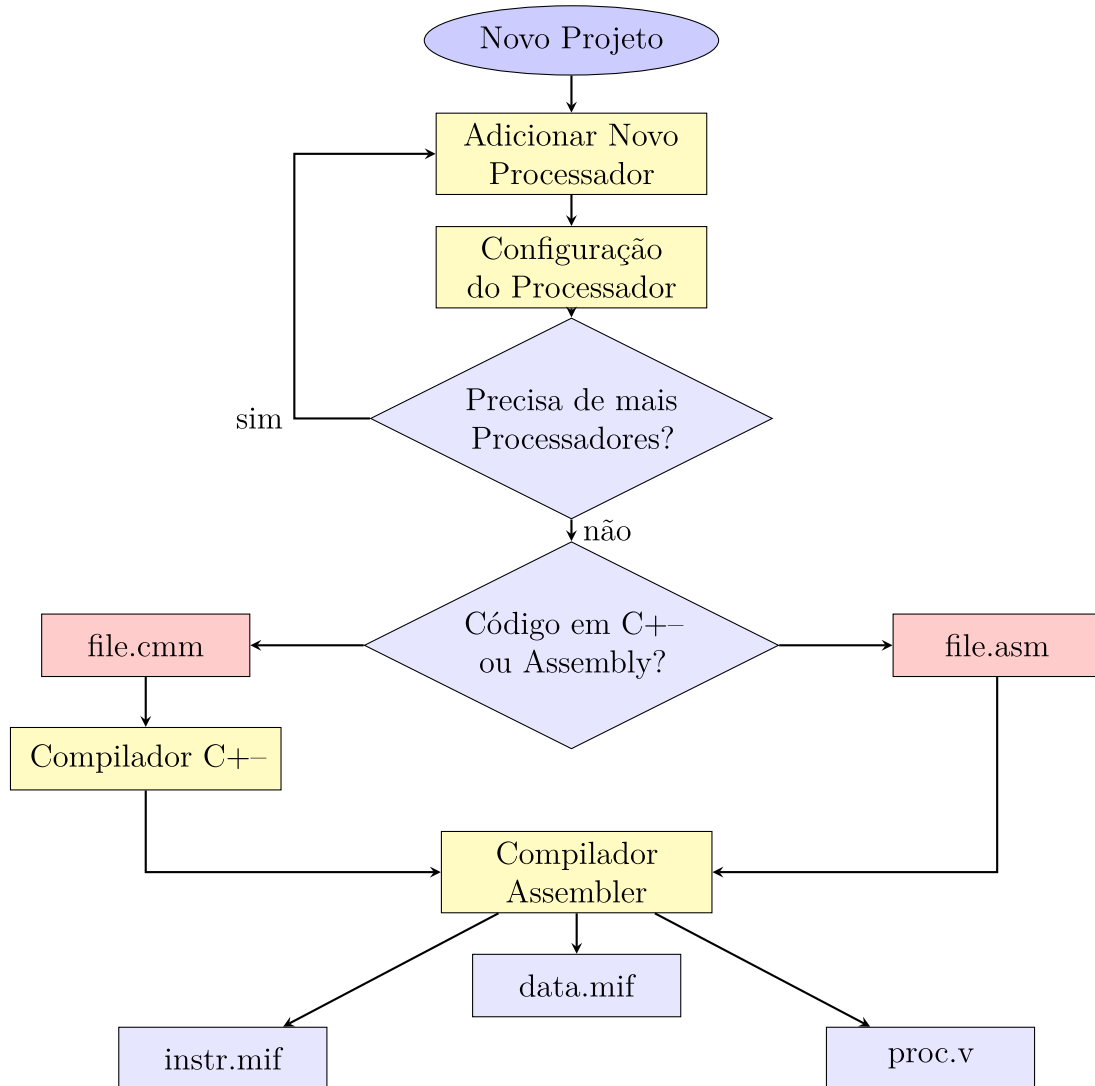
- General Settings:** Contains a text field for 'Processor Name' with the value 'Proc_Sim'.
- ULA Settings:** Contains a 'Processor Type' section with two radio buttons: 'Fixed Point' (selected) and 'Floating Point'. Below this are three text fields: 'N bits' (value 32), 'Nb Mantissa' (empty), and 'Nb Exponent' (empty).
- Memory Stack Settings:** Contains two text fields: 'Data Stack Size' (value 10) and 'Instruction Stack Size' (value 10).
- I/O Settings:** Contains two text fields: 'Number of Input Ports' (value 2) and 'Number of Output Ports' (value 2).
- Gain:** Contains a text field for 'Gain' (value 8) and a label '#GAIN: 2(8) = 256'.

At the bottom center of the window is a large 'Generate' button.

Fonte: (Elaborado pelo autor (2025).)

Dentre as opções oferecidas, é possível selecionar o tipo de aritmética utilizada pelo processador — seja ponto fixo, com precisão inteira definida, ou ponto flutuante, com parametrização separada para mantissa e expoente. O usuário também define o número de bits para representação dos dados, o tamanho reservado para as pilhas de dados e de instruções, além da quantidade de portas de entrada e saída disponíveis, que influenciam diretamente no acoplamento com outros módulos e na estruturação dos fluxos de dados. O fluxograma do processo de criação pode ser visto na Figura 15:

Figura 15 – Fluxograma do processo de criação de um projeto no SAPHO.



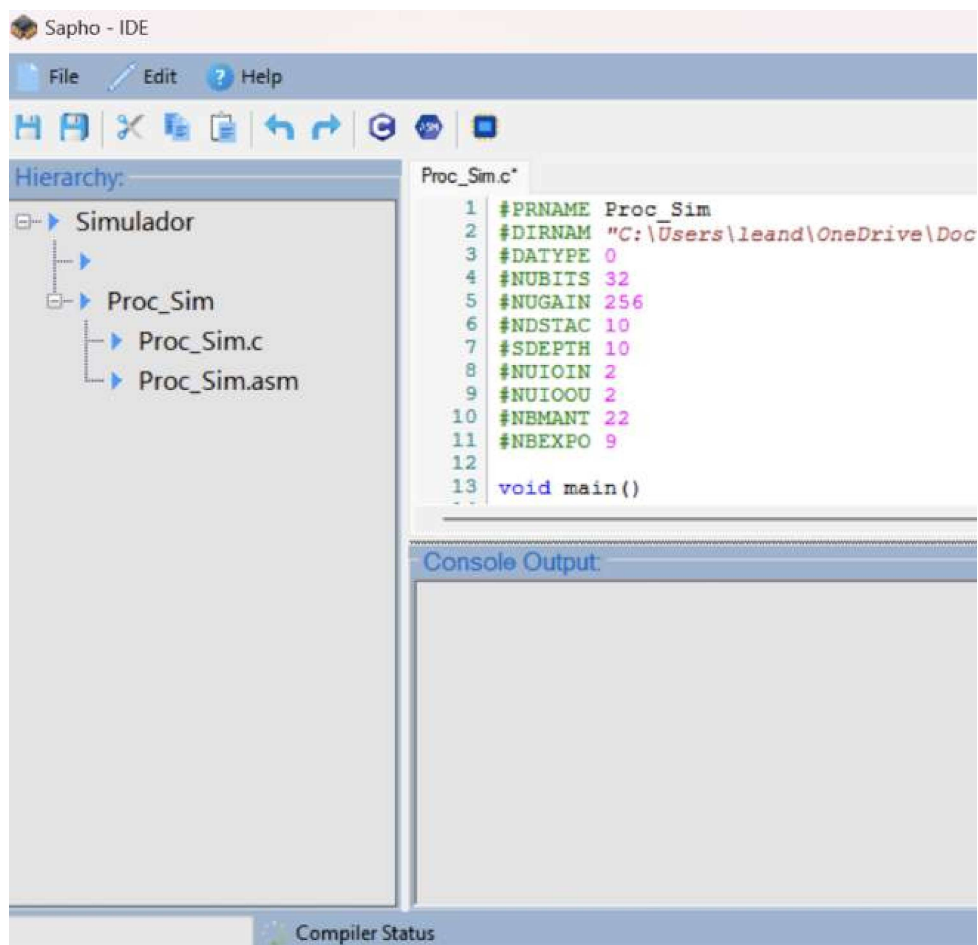
Fonte: (Elaborado pelo autor (2025).)

As escolhas feitas nessa etapa são automaticamente convertidas em diretivas de compilação, inseridas no início do código-fonte em linguagem C++. Tais diretivas orientam tanto a montagem das instruções quanto a alocação dos blocos lógicos durante a geração do hardware. Como resultado da compilação, são criados os arquivos `data.mif` e `instr.mif`, que contêm, respectivamente, os dados de inicialização das memórias de dados e de programa. Um arquivo adicional em Verilog é também gerado, contendo a descrição

completa do processador com base nas configurações escolhidas, como pode ser visto na Figura 15.

Esse fluxo permite uma adaptação rápida do processador às especificidades do projeto, garantindo que o software compilado esteja totalmente compatível com o hardware que será sintetizado. A Figura 14 apresenta a interface de configuração de um novo processador dentro do ambiente SAPHO, e a Figura 16 ilustra parte de um código em C+- gerado com a configuração do processador do exemplo da figura anterior.

Figura 16 – Parametrização de Processador no SAPHO.



Fonte: (Elaborado pelo autor (2025).)

Com esse conjunto de ferramentas, o SAPHO oferece uma cadeia de desenvolvimento completa e adaptável, desde a modelagem de alto nível até a síntese do hardware em FPGA, promovendo autonomia e eficiência no desenvolvimento de arquiteturas embarcadas customizadas.

3.2 TECNOLOGIA FPGA

Os dispositivos *Field-Programmable Gate Array* (FPGAs) representam uma classe de circuitos integrados de lógica programável amplamente utilizada em sistemas embarcados

que exigem alto desempenho, flexibilidade e processamento em tempo real. Diferentemente dos processadores tradicionais, que seguem um fluxo de execução sequencial de instruções, as FPGAs permitem a construção de circuitos digitais personalizados por meio da programação de sua estrutura interna, possibilitando a implementação de arquiteturas paralelas, *pipelines* otimizados e algoritmos digitais dedicados diretamente no hardware (TRIMBERGER, 2015).

A sigla FPGA refere-se, literalmente, a um "arranjo de portas programáveis em campo", o que significa que sua estrutura lógica pode ser definida pelo usuário após a fabricação do chip. Essa característica contrasta fortemente com os circuitos integrados do tipo ASIC (*Application-Specific Integrated Circuit*), cuja lógica é fixa e definida no processo de fabricação, tornando qualquer modificação subsequente impossível. Assim, enquanto os ASICs são otimizados para máxima eficiência em grandes volumes de produção, as FPGAs destacam-se pela versatilidade e pelo curto ciclo de desenvolvimento, sendo ideais para prototipagem rápida, pesquisa aplicada e aplicações que exigem atualizações regulares (KUON; ROSE, 2008).

Arquiteturalmente, uma FPGA é composta por uma matriz regular de blocos lógicos programáveis (*Logic Elements* – LEs), conectados entre si por uma rede de interconexão reconfigurável. Cada bloco lógico é capaz de implementar funções booleanas arbitrárias por meio de tabelas de verdade (*Look-Up Tables* – LUTs), além de contar com *flip-flops* para armazenamento de estado e, em muitos casos, recursos adicionais como somadores, comparadores e multiplicadores dedicados (*DSP slices*). Esses recursos possibilitam a construção de circuitos síncronos completos, incluindo unidades aritméticas, máquinas de estados, controladores de fluxo e até mesmo processadores inteiros, como o SAPHO, objeto de estudo deste trabalho.

Além dos blocos lógicos, as FPGAs modernas incorporam memórias embarcadas (*Block RAMs* – BRAMs), interfaces de entrada/saída configuráveis (*I/O banks*), transceptores de alta velocidade e, em alguns casos, núcleos de processadores embutidos (*hard-core*), como processadores ARM em dispositivos do tipo *System-on-Chip* (SoC FPGA). A presença dessas estruturas heterogêneas permite a criação de sistemas completos em um único chip (*SoC*), combinando processamento em hardware customizado e software embarcado, otimizando desempenho, consumo energético e área ocupada.

A natureza reconfigurável das FPGAs permite que o projeto lógico seja adaptado dinamicamente a diferentes aplicações. Essa flexibilidade é especialmente valiosa em contextos onde os requisitos do sistema evoluem com o tempo, ou onde há necessidade de ajustes finos em resposta a mudanças nas condições operacionais. Em experimentos científicos, como os realizados no CERN, essa característica permite o refinamento de algoritmos de leitura e filtragem de dados sem a necessidade de substituir o hardware instalado.

Outro aspecto técnico relevante é a capacidade das FPGAs de operar com latência mínima e em tempo determinístico. Isso as torna particularmente adequadas para aplicações de aquisição e processamento de sinais em tempo real, como é o caso dos sistemas de leitura e calibração do calorímetro TileCal. A possibilidade de paralelismo massivo, aliada à alta frequência de operação e à previsibilidade do comportamento temporal, permite a construção de arquiteturas que atendem aos rigorosos requisitos de desempenho impostos por experimentos de física de altas energias.

Do ponto de vista do desenvolvimento, os projetos para FPGA são descritos por meio de linguagens de descrição de hardware (HDLs), como VHDL e Verilog, que especificam o comportamento lógico do circuito a ser implementado. Após a descrição do projeto, uma cadeia de ferramentas (*toolchain*) realiza etapas como síntese lógica, mapeamento para a arquitetura alvo, colocação e roteamento dos blocos lógicos e geração do *bitstream* final que será carregado na FPGA (BROWN; VRANESIC, 2007). Esse fluxo de desenvolvimento permite controle total sobre a arquitetura resultante, com possibilidade de otimizações específicas para desempenho, área ou consumo de energia.

Em resumo, as FPGAs configuram uma plataforma versátil, poderosa e amplamente adotada em sistemas embarcados de alto desempenho. Sua combinação de flexibilidade, paralelismo, determinismo temporal e possibilidade de customização torna essa tecnologia uma escolha natural para aplicações que requerem controle preciso sobre o fluxo de dados e a lógica de processamento, como é o caso do simulador de pulsos do TileCal investigado nesta dissertação.

3.2.1 Vantagens para Aplicações em Tempo Real e Sistemas Reconfiguráveis

A crescente demanda por sistemas embarcados capazes de operar com alta taxa de eventos, baixa latência e comportamento determinístico tem impulsionado o uso de FPGAs em aplicações críticas de tempo real. A arquitetura reconfigurável desses dispositivos permite a personalização de circuitos digitais de forma a atender com precisão os requisitos temporais e lógicos impostos por algoritmos específicos, sem a necessidade de modificar fisicamente o hardware. Esse paradigma representa uma alternativa estratégica frente a soluções tradicionais baseadas em microprocessadores generalistas ou circuitos integrados fixos (ASICs).

Uma das principais vantagens das FPGAs é a sua capacidade de operar com altas frequências de clock, compatíveis com os requisitos de sistemas que necessitam de respostas rápidas e previsíveis. Além disso, sua natureza fortemente paralela permite a execução simultânea de múltiplas operações lógicas e aritméticas, sem depender de ciclos sequenciais de instruções. Isso é particularmente relevante em aplicações como a simulação de pulsos do TileCal, onde a geração contínua e sincronizada de sinais exige processamento determinístico, livre de incertezas introduzidas por pipelines variáveis ou caches.

Outro aspecto decisivo é a execução determinística das FPGAs, o que significa que, para uma dada entrada e configuração lógica, o comportamento do sistema é invariável e previsível. Essa característica contrasta com arquiteturas baseadas em software, como CPUs e GPUs, nas quais fatores como gerenciamento de interrupções, escalonamento de tarefas e dependência de instruções podem introduzir variações indesejadas no tempo de resposta. No contexto de experimentos de física de partículas, onde a precisão temporal é crítica, esse comportamento previsível torna as FPGAs especialmente valiosas.

Adicionalmente, a capacidade de reconfiguração das FPGAs oferece uma flexibilidade única. Ao contrário dos ASICs, cujo circuito é fixo após a fabricação, as FPGAs permitem atualizações e modificações do projeto lógico mesmo após a instalação em campo, viabilizando correções de bugs, ajustes de desempenho e até a substituição completa do algoritmo implementado. Essa propriedade reduz significativamente o tempo de resposta entre o desenvolvimento e a aplicação, além de minimizar custos com refabricação.

No escopo de experimentos científicos de larga escala, como aqueles conduzidos no CERN, o uso de FPGAs é amplamente difundido. No detector ATLAS, por exemplo, esses dispositivos são empregados em sistemas de disparo (*trigger*) e leitura de dados, onde a latência mínima e a confiabilidade são requisitos inegociáveis. As FPGAs são responsáveis por realizar, em tempo real, a filtragem de milhões de eventos por segundo, permitindo que apenas aqueles com maior potencial físico sejam armazenados para análise posterior. Essa filtragem é feita por circuitos especializados, programados para reconhecer padrões específicos de interesse físico com base na forma e na energia dos pulsos registrados.

Em síntese, as FPGAs combinam alto desempenho computacional com flexibilidade de implementação e previsibilidade temporal, consolidando-se como a plataforma preferencial para sistemas embarcados que operam sob requisitos severos de tempo real. Sua aplicabilidade em ambientes como o TileCal reforça sua relevância como tecnologia de base para o desenvolvimento de simuladores, filtros digitais e algoritmos de reconstrução de energia em experimentos de fronteira.

4 METODOLOGIA

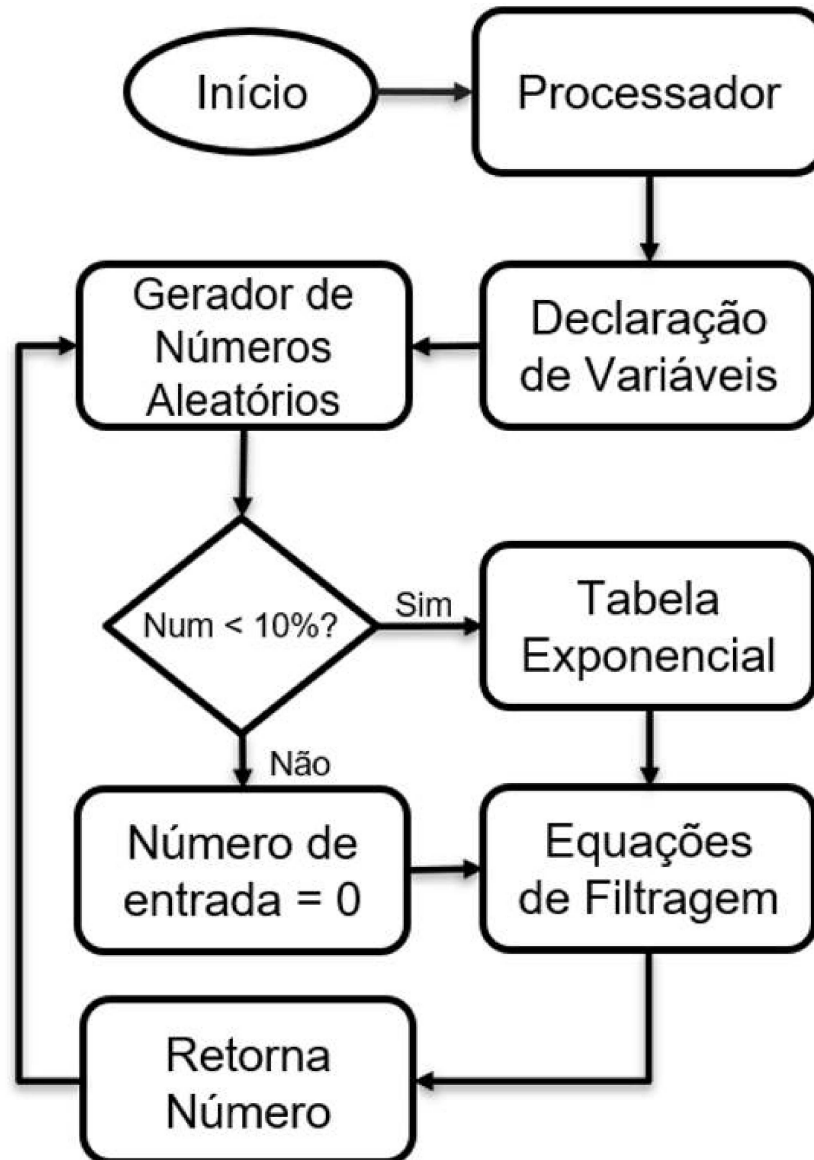
Este capítulo descreve em detalhes a metodologia adotada para o desenvolvimento do simulador de pulsos hadrônicos proposto neste trabalho. O objetivo principal do simulador é gerar, de forma determinística e em tempo compatível com os requisitos do calorímetro TileCal, formas de pulso que representem o sinal eletrônico típico de eventos físicos detectados nos módulos hadrônicos do experimento ATLAS. Para isso, foi adotada uma abordagem modular, em que cada bloco funcional do sistema foi projetado, testado e validado separadamente, antes de ser integrado na versão final do simulador.

Diferentemente de uma abordagem puramente teórica ou baseada em simulações genéricas, o foco deste trabalho esteve na construção de um sistema executável que pudesse, futuramente, ser implementado em hardware reconfigurável, como FPGAs. Assim, foram consideradas desde as limitações de desempenho e precisão numérica até os custos computacionais de cada etapa da simulação.

O simulador completo foi estruturado com base em blocos lógicos bem definidos, cada um responsável por uma parte do processo de geração do sinal. O primeiro deles é o gerador de números aleatórios, responsável por fornecer variação controlada aos parâmetros do pulso, simulando a aleatoriedade dos eventos reais. Em seguida, a tabela exponencial permite calcular de forma eficiente o decaimento da curva de pulso. A modelagem matemática do pulso define a forma geral da curva, com base em parâmetros ajustáveis como amplitude, tempo de subida e tempo de decaimento. Por fim, diferentes arquiteturas de simuladores foram implementadas, cada uma explorando um conjunto distinto de recursos da linguagem C e da arquitetura SAPHO, incluindo chamadas de sub-rotina, uso de pilha, ponteiros e armazenamento dinâmico.

A Figura 17 ilustra, de forma esquemática, o fluxo geral de funcionamento do simulador. Cada etapa representada no diagrama será abordada individualmente nas próximas seções deste capítulo, com destaque para os aspectos computacionais, decisões de projeto e estratégias de otimização adotadas.

Figura 17 – Fluxograma base do Simulador.



Fonte: (Elaborado pelo autor (2025).)

A metodologia aqui descrita tem como objetivo garantir que os sinais gerados apresentem não apenas fidelidade em relação aos modelos físicos, mas também viabilidade computacional para implementação futura em FPGA. A abordagem escolhida busca, portanto, um equilíbrio entre precisão, desempenho e simplicidade arquitetural, características fundamentais em sistemas embarcados aplicados à instrumentação de experimentos de física de altas energias.

4.1 GERADOR DE NÚMEROS ALEATÓRIOS

A noção de aleatoriedade acompanha a humanidade desde a Antiguidade, sendo inicialmente associada a práticas religiosas, jogos de azar e especulações filosóficas. Para os gregos clássicos, por exemplo, o conceito de acaso estava intimamente ligado à im-

previsibilidade dos eventos naturais e ao debate sobre determinismo versus liberdade (HACKING, 2001). Aristóteles distinguia entre fenômenos necessários, contingentes e fortuitos, destacando que o acaso representava aquilo que escapava à causalidade estrita (ARISTOTLE, 1984).

Do ponto de vista científico, a formalização do conceito de aleatoriedade surge de maneira mais clara no século XVII, com o desenvolvimento da teoria das probabilidades por matemáticos como Blaise Pascal e Pierre de Fermat. Esse avanço esteve diretamente relacionado à análise dos jogos de azar, marcando a primeira tentativa sistemática de quantificar o acaso (HACKING, 2006). A partir desse momento, a aleatoriedade deixa de ser vista apenas como um fenômeno metafísico ou intuitivo e passa a ser objeto de estudo rigoroso, integrando-se gradualmente à estatística e à matemática aplicada.

Nos séculos XIX e XX, com o fortalecimento da estatística matemática e a aplicação de métodos probabilísticos em áreas como física, engenharia e ciências sociais, a aleatoriedade consolidou-se como ferramenta fundamental de modelagem científica.

Antes do advento dos computadores, a obtenção de números aleatórios para aplicações científicas e estatísticas era feita de forma manual ou mecânica. Entre os métodos mais antigos estão o uso de dados, moedas e roletas, empregados tanto em jogos de azar quanto em experimentos probabilísticos elementares. Esses instrumentos, apesar de rudimentares, forneceram as primeiras tentativas de quantificação empírica do acaso.

Com o crescimento da estatística no início do século XX, surgiram iniciativas formais de sistematização da aleatoriedade. Um marco importante foi a publicação das *Tables of Random Sampling Numbers*, organizadas por Kendall e Smith em 1939, que disponibilizaram sequências pré-computadas de números aleatórios a serem utilizadas em experimentos estatísticos, pesquisas operacionais e simulações (KENDALL; SMITH, 1939). Essas tabelas representaram um avanço significativo, pois permitiram aos pesquisadores acesso a grandes amostras de números com características estatísticas próximas da uniformidade, sem depender de dispositivos físicos sujeitos a vieses.

4.1.1 Introdução da geração computacional de números aleatórios

A transição para a geração computacional ocorreu na década de 1950, quando os primeiros algoritmos matemáticos foram executados em computadores digitais de grande porte. Em 1951, Derrick H. Lehmer propôs um método baseado em recorrências lineares modulares, conhecido posteriormente como *Linear Congruential Generator* (LCG) (LEHMER, 1951). Esse algoritmo pode ser expresso pela recorrência:

$$X_{n+1} = (a \cdot X_n + c) \bmod m \quad (4.1)$$

onde a é o multiplicador, c o incremento e m o módulo. Apesar de sua simplicidade,

o método de Lehmer estabeleceu as bases para a maioria dos geradores pseudoaleatórios modernos, oferecendo longos períodos, uniformidade razoável e facilidade de implementação em computadores digitais.

Esse desenvolvimento marcou uma mudança de paradigma: em vez de depender de instrumentos físicos ou tábuas estáticas, a aleatoriedade passou a ser produzida de forma determinística e reproduzível, abrindo caminho para o uso extensivo de simulações em áreas como física nuclear, estatística aplicada e engenharia.

Após a formulação inicial de Lehmer, diversos aprimoramentos foram propostos ao longo das décadas seguintes, consolidando o uso de geradores congruenciais e estabelecendo critérios mais rigorosos para sua qualidade estatística. Donald Knuth, em sua obra clássica *The Art of Computer Programming* publicada originalmente em 1969 e ampliada em edições posteriores, dedicou um volume inteiro à análise de algoritmos semi-numéricos, incluindo geradores de números pseudoaleatórios. Knuth enfatizou propriedades como período máximo, distribuição uniforme e ausência de correlações de curto alcance, parâmetros essenciais para garantir a confiabilidade das simulações (KNUTH, 1997).

Na década de 1980, Park e Miller (PARK; MILLER, 1988) introduziram o chamado *Minimal Standard Generator*, um gerador congruencial linear que se destacou pela simplicidade e pela qualidade estatística satisfatória quando comparado a implementações da época. O trabalho destacou a importância da escolha cuidadosa dos parâmetros (a, c, m) , demonstrando que geradores mal configurados poderiam exibir sérias deficiências, como períodos curtos e padrões previsíveis. Esse artigo tornou-se referência ao propor um conjunto mínimo de requisitos para que um gerador pudesse ser considerado confiável em aplicações práticas.

Esses desenvolvimentos estabeleceram um corpo teórico sólido para a geração de números pseudoaleatórios, que serviu de base para a evolução de métodos mais sofisticados nas décadas seguintes, incluindo os algoritmos baseados em *shifts*, transformadas e combinações múltiplas.

4.1.2 Geração Física vs. Pseudoaleatória

A distinção entre geradores de números verdadeiramente aleatórios e geradores pseudoaleatórios é fundamental para compreender as escolhas de implementação em sistemas digitais embarcados. Em termos conceituais, a **aleatoriedade verdadeira** é aquela originada de fenômenos físicos intrinsecamente imprevisíveis, enquanto a **pseudoaleatoriedade** corresponde à geração determinística de sequências numéricas que, apesar de previsíveis a partir de uma semente inicial, apresentam propriedades estatísticas semelhantes às da aleatoriedade natural.

4.1.2.1 Fontes de aleatoriedade física

Os geradores de números verdadeiramente aleatórios (TRNG – *True Random Number Generators*) baseiam-se em processos físicos cuja evolução não pode ser prevista com exatidão. Entre as principais fontes exploradas destacam-se:

- **Ruído térmico:** resulta da agitação aleatória de portadores de carga em resistores ou diodos semicondutores. É uma das fontes mais antigas e amplamente utilizadas em circuitos eletrônicos para geração de entropia (SCHINDLER, 2001).
- **Jitter de osciladores:** pequenas variações aleatórias no período de osciladores digitais ou analógicos. Esse método tornou-se popular em projetos de TRNG em FPGA e microcontroladores devido à simplicidade de implementação (SUNAR; MARTIN; STINSON, 2007).
- **Fenômenos radioativos:** o decaimento nuclear e a detecção de partículas são processos inerentemente probabilísticos e constituem fontes confiáveis de aleatoriedade, embora impraticáveis em sistemas embarcados comuns.
- **Flutuações quânticas:** aproveitadas em dispositivos modernos de geração quântica de números aleatórios (QRNG – *Quantum Random Number Generators*), exploram a natureza probabilística de processos quânticos, como emissão de fótons ou superposição de estados (HERRERO-COLLANTES; GARCIA-ESCARTIN, 2017).

Embora garantam uma aleatoriedade mais “pura”, todos esses métodos requerem conversão analógica-digital, filtragem e pós-processamento para eliminar vieses estatísticos e ajustar a distribuição de saída. Além disso, demandam circuitos especializados, calibração frequente e podem apresentar problemas de latência ou instabilidade sob determinadas condições ambientais.

4.1.2.2 Geradores pseudoaleatórios

Por outro lado, os geradores pseudoaleatórios (PRNG – *Pseudo-Random Number Generators*) são algoritmos determinísticos implementados em software ou hardware. A partir de uma semente inicial, produzem sequências com características estatísticas semelhantes às dos números aleatórios verdadeiros, mas de forma totalmente reprodutível. Sua grande vantagem é a simplicidade estrutural: exigem apenas operações aritméticas e lógicas elementares, podendo ser implementados de forma eficiente mesmo em arquiteturas restritas.

Apesar de previsíveis, PRNGs bem projetados podem apresentar longos períodos, baixa correlação entre valores consecutivos e boa uniformidade, sendo adequados para a maioria das aplicações científicas e de engenharia. Diferentemente dos TRNGs, não sofrem

com ruído ambiental ou limitações físicas, e sua saída pode ser reproduzida fielmente para fins de validação e depuração.

4.1.2.3 *Vantagens e limitações*

A Tabela 3 resume as principais vantagens e limitações de cada abordagem:

Tabela 3 – Comparação entre geradores verdadeiramente aleatórios e pseudoaleatórios.

Característica	TRNG (físico)	PRNG (algorítmico)
Fonte	Fenômenos físicos imprevisíveis (ruído, jitter, quântico)	Algoritmos determinísticos (LCG, MLCG, Xorshift, etc.)
Entropia	Alta (intrínseca ao processo físico)	Dependente da semente, limitada pelo período do algoritmo
Custo de implementação	Elevado: requer circuitos analógicos, ADCs, calibração	Baixo: operações aritméticas e lógicas simples
Latência	Pode ser significativa (digitalização e filtragem)	Muito baixa, compatível com clock do sistema
Reprodutibilidade	Não reprodutível	Totalmente reprodutível a partir da semente
Adequação a FPGA	Limitada: alto custo de hardware e difícil integração	Excelente: fácil síntese, baixo consumo de recursos

No escopo deste trabalho, a adoção de TRNGs físicos se mostra inviável por três motivos principais: (i) a complexidade e o custo de implementação em FPGA ou em um processador minimalista como o SAPHO, que não dispõe de unidades analógicas; (ii) a necessidade de reprodutibilidade nos testes, fundamental para a validação e comparação de resultados; e (iii) a maior latência e consumo de recursos associados a circuitos físicos de coleta de entropia.

Dessa forma, optou-se por geradores pseudoaleatórios, em particular da classe congruencial multiplicativa com incremento, que oferecem um balanço adequado entre simplicidade estrutural, qualidade estatística e integração com a arquitetura do simulador.

4.1.3 Métodos Clássicos de Geração Pseudoaleatória

A geração pseudoaleatória em sistemas digitais baseia-se em recorrências determinísticas que produzem sequências com propriedades estatísticas semelhantes às de uma fonte verdadeiramente aleatória. Entre as famílias clássicas, destacam-se os geradores congruenciais (LCG/MLCG), os baseados em operações bit a bit (Xorshift), os de grande período e equidistribuição (Mersenne Twister) e os lineares sobre $GF(2)$ (LFSR). Nesta subseção, descrevem-se os princípios, parâmetros e implicações práticas de implementação, com foco nas restrições de hardware encontradas em FPGA e no processador SAPHO.

4.1.3.1 Geradores Congruenciais Lineares (LCG e MLCG)

Os *Linear Congruential Generators* (LCG) têm a forma clássica (KNUTH, 1997; PARK; MILLER, 1988):

$$X_{n+1} = (a \cdot X_n + c) \bmod m, \quad (4.2)$$

em que a é o multiplicador, c o incremento, m o módulo e X_0 a semente. Quando $c = 0$, obtém-se o caso multiplicativo (**MCG**); quando $c \neq 0$, tem-se o LCG aditivo. Na prática de hardware digital, é comum escolher $m = 2^w$ (por exemplo, $w = 32$), de modo que a redução modular seja implementada automaticamente pelo *wrap-around* aritmético de w bits, o que elimina divisões custosas.

Nesse modelo, alguns pontos devem ser destacados:

- **Período e escolha de parâmetros:** para $m = 2^w$ e $c \neq 0$, condições de Hull–Dobell garantem período máximo m se: (i) $\gcd(c, m) = 1$; (ii) $a - 1$ é múltiplo de todos os fatores primos de m ; (iii) se m é múltiplo de 4, então $a - 1$ é múltiplo de 4 (KNUTH, 1997). Para MCG ($c = 0$), o período máximo é $m/4$ quando $m = 2^w$ e a é apropriado. Além do período, o *spectral test* avalia a qualidade geométrica da sequência em dimensões maiores, sendo crítico na seleção de a (KNUTH, 1997; L'ECUYER, 1990).
- **Saída e descarte de bits baixos:** em LCG/MCG, bits menos significativos tendem a exibir correlações; por isso, recomenda-se extrair bits mais altos (p.ex., $\text{out} = (X_{n+1} \gg s) \& \text{mask}$), como empregado neste trabalho, reduzindo artefatos estatísticos em aplicações de simulação.
- **Custo em hardware:** para $m = 2^w$, a implementação requer uma multiplicação $w \times w$, uma soma e um deslocamento/máscara — operações bem suportadas em FPGA e viáveis no SAPHO. Trata-se de uma relação vantajosa entre custo e qualidade para simulações com requisitos de reprodutibilidade e baixa latência.

4.1.3.2 Xorshift

Os geradores *Xorshift* aplicam uma sequência de deslocamentos e operações XOR sobre um estado interno de largura fixa (MARSAGLIA, 2003). Um esquema típico (32 ou 64 bits) é:

$$x \leftarrow x \oplus (x \ll a); \quad x \leftarrow x \oplus (x \gg b); \quad x \leftarrow x \oplus (x \ll c), \quad (4.3)$$

com tripla (a, b, c) escolhida para maximizar o período (tipicamente $2^w - 1$) e a qualidade. Extensões como *xorshift** e *xorshift+* aplicam uma multiplicação por constante ou soma de estados para melhorar testes estatísticos modernos (*BigCrush*), porém aumentam ligeiramente o custo em hardware.

Xorshift é extremamente leve (apenas registradores, shifts e XOR), com latência mínima e alta taxa de amostragem — um forte candidato em FPGA. Contudo, variantes simples podem apresentar falhas em testes rigorosos e correlações lineares, exigindo *scrambling* adicional para aplicações mais sensíveis (MARSAGLIA, 2003).

4.1.3.3 *Mersenne Twister (MT19937)*

O *Mersenne Twister* (MT19937) (MATSUMOTO; NISHIMURA, 1998) é um gerador com período colossal $2^{19937} - 1$ e equidistribuição em alta dimensão (623 para saídas de 32 bits). Baseia-se em uma recorrência linear sobre GF(2) com *twist* e *tempering* da saída.

É excelente para simulações em software geral pela combinação de grande período e boa equidistribuição. Em contrapartida, exige um estado relativamente grande (tipicamente 624 palavras de 32 bits) e uma lógica mais complexa (acessos a vetor, *tempering*), o que aumenta o uso de memória e a área em FPGA. Além disso, por ser linear sobre GF(2), pode apresentar fraquezas em certos bits baixos e não é apropriado para criptografia.

4.1.3.4 *Geradores baseados em LFSR*

Linear Feedback Shift Registers (LFSR) implementam recorrências lineares em GF(2) utilizando realimentação por XOR segundo um polinômio primitivo (TAUSWORTHE, 1965; GOLOMB, 1967). Um LFSR de n bits com polinômio primitivo alcança período máximo $2^n - 1$.

São extremamente simples e rápidos em hardware: requerem apenas *flip-flops* e algumas portas XOR. São onipresentes em *built-in self-test*, *scramblers*, codificadores de canal e aplicações de verificação.

Por serem estritamente lineares em GF(2), exibem padrões e correlações detectáveis em diversas métricas; em geral, *sozinhos* não são recomendados para simulações que exigem alta fidelidade estatística. Combinações (p.ex., somas, *decimations*, filtros não lineares) podem mitigar limitações, ao custo de maior complexidade.

4.1.3.5 *Comparação e implicações práticas em hardware*

A Tabela 4 resume características relevantes para escolha em ambientes embarcados e de FPGA.

Tabela 4 – Comparação geral entre métodos clássicos de PRNG (visão de alto nível).

Método	Período típico	Qualidade e Uniformidade	Custo em hardware (FPGA)
LCG/MLCG ($m = 2^w$)	até 2^w (LCG com $c \neq 0$); menor em MCG	Adequada se parâmetros bem escolhidos; bits baixos mais fracos; passa em testes básicos; pode falhar nos mais severos	Baixo: 1 mult + 1 soma + shifts/máscara; ótima latência e área
Xorshift	até $2^w - 1$	Muito rápido; versões simples podem falhar em testes modernos; variantes * e + melhoram	Muito baixo: registradores + XOR + shifts; variantes com * usam 1 mult
Mersenne Twister (MT19937)	$2^{19937} - 1$	Excelente equidistribuição; boa para simulação em software; linear sobre GF(2)	Moderado/alto: grande estado (624 palavras); mais lógica e memória
LFSR (polinômio primitivo)	$2^n - 1$	Linear; correlações evidentes; requer não-linearidades adicionais para melhor qualidade	Muito baixo: FF + XOR; escalável com pouquíssimos recursos

No cenário do simulador do TileCal, a escolha recai sobre geradores com (i) baixo custo lógico, (ii) alta taxa de amostragem, (iii) reprodutibilidade e (iv) qualidade suficiente para modelagem de amplitude e ocorrência de pulsos. Embora MT19937 ofereça período e equidistribuição superiores, seu custo em memória e lógica não se justifica em FPGA. LFSR é leve, mas linearidade excessiva pode introduzir vieses indesejados. Xorshift é muito atrativo em hardware, mas versões mínimas exigem cuidado adicional para evitar falhas estatísticas. O LCG/MLCG com parâmetros bem escolhidos e descarte de bits baixos apresenta o melhor equilíbrio entre simplicidade e qualidade para esta aplicação específica (KNUTH, 1997; PARK; MILLER, 1988).

4.1.4 Justificativa do Método Adotado neste Trabalho

A escolha de um gerador pseudoaleatório para o simulador desenvolvido neste trabalho deve considerar, além da qualidade estatística, as limitações impostas pela plataforma de hardware alvo. O processador SAPHO não implementa ponto flutuante no padrão IEEE-754 (23 bits de mantissa e 8 bits de expoente), mas utiliza uma representação customizável em que o número de bits do expoente e da mantissa pode ser ajustado, desde que a soma não exceda 31. Essa flexibilidade permite otimizar a síntese em FPGA, reduzindo área e consumo, porém limita a precisão e dificulta a implementação de algoritmos que dependem de ponto flutuante completo. Além disso, a lógica disponível deve ser preservada para outras partes críticas do sistema, como filtros digitais e unidades de reconstrução de energia, o que impõe severas restrições de área e consumo energético.

Entre os métodos avaliados, destacam-se três candidatos viáveis em hardware: LFSR, Xorshift e MLCG.

- **LFSR**: é o mais simples de implementar em FPGA, exigindo apenas *flip-flops* e algumas portas XOR. Apesar da eficiência, sua linearidade intrínseca sobre GF(2) produz padrões detectáveis em análises estatísticas e limita a qualidade da aleatoriedade para simulações de fenômenos físicos. O uso isolado de LFSR poderia introduzir artefatos nos sinais simulados.
- **Xorshift**: também muito eficiente em hardware, pois utiliza apenas operações de deslocamento e XOR. Entretanto, variantes mínimas tendem a falhar em baterias de testes mais rigorosas (MARSAGLIA, 2003), exigindo pós-processamento (como multiplicações adicionais ou combinações) para atingir qualidade estatística comparável à de métodos congruenciais bem parametrizados.
- **MLCG (Multiplicative Linear Congruential Generator com incremento)**: requer apenas uma multiplicação de 32 bits, uma soma e um deslocamento/máscara. Embora não apresente período colossal como o Mersenne Twister, fornece uniformidade adequada, boa dispersão dos bits mais significativos e simplicidade de implementação. Além disso, a escolha de parâmetros apropriados, fundamentada na literatura, garante que o gerador satisfaça os critérios de qualidade estatística relevantes para este trabalho (PARK; MILLER, 1988; L'ECUYER, 1990).

Diante dessas alternativas, o **MLCG** mostrou-se a opção mais equilibrada. Diferentemente dos LFSR, não sofre de linearidade excessiva, e, ao contrário do Xorshift, não depende de extensões para corrigir falhas estatísticas conhecidas. Sua estrutura aritmética é compatível tanto com a ISA simplificada do SAPHO quanto com síntese eficiente em FPGA, preservando área e tempo de execução.

Além disso, um requisito importante para a validação científica é a capacidade de reproduzir os resultados obtidos. Por se tratar de um algoritmo determinístico, o MLCG garante que, a partir de uma semente fixa, a sequência gerada será sempre idêntica. Essa propriedade permite repetir simulações com condições iniciais controladas, comparar resultados entre execuções distintas e validar o funcionamento em diferentes plataformas (software e FPGA). Além disso, facilita a depuração e o rastreamento de falhas durante a etapa de desenvolvimento.

A parametrização adotada neste trabalho segue recomendações da literatura, utilizando:

- $a = 2891336453$ como multiplicador, valor amplamente citado por gerar sequências de alta qualidade em espaço de 32 bits (L'ECUYER, 1990);

- $c = 12345$ como incremento constante, garantindo independência em relação ao módulo 2^{32} ;
- $seed = 4123$ como semente inicial, assegurando reprodutibilidade dos experimentos;
- mod não será incluído na equação pois, em um circuito digital, a variável de saída já possui limitação pelo seu número de bits.

Após cada iteração, aplica-se uma etapa de deslocamento e máscara sobre os bits mais significativos, eliminando correlações nos bits inferiores e restringindo a saída ao intervalo desejado, conforme a Equação 4.4. Essa técnica é simples, mas suficiente para assegurar a dispersão adequada dos valores em aplicações de simulação.

$$out = (rnd_{n+1} \gg shift) \& mask \quad (4.4)$$

Do ponto de vista de implementação, o MLCG aproveita recursos nativos das FPGAs modernas, que oferecem multiplicadores de 18×18 ou 32×32 bits otimizados em blocos dedicados (DSP slices). A operação de soma e as máscaras são triviais em lógica combinacional. Assim, o custo em área é mínimo, a latência por número gerado é baixa e a escalabilidade do sistema é mantida.

A adoção do MLCG neste trabalho equilibra simplicidade, qualidade estatística e compatibilidade com hardware restrito. Esse gerador atende aos requisitos do simulador do TileCal ao permitir a emulação de sinais com variação natural de amplitude e ocorrência de pulsos, sem comprometer os recursos do SAPHO ou da FPGA utilizada.

4.1.5 Implementação no Simulador do TileCal

Uma vez definido o algoritmo pseudoaleatório mais adequado, sua integração ao simulador do TileCal foi realizada de modo a reproduzir características fundamentais dos sinais observados experimentalmente. O gerador desempenha papéis centrais em diferentes etapas do fluxo de dados, influenciando a dinâmica de geração de pulsos, sua variabilidade e a construção de cenários complexos como *pile-up*.

O gerador pseudoaleatório baseado em MLCG é utilizado em duas funções principais:

- **Controle probabilístico da ocorrência de pulsos:** o primeiro uso do gerador está associado à determinação da presença ou ausência de um pulso em cada ciclo de simulação. Essa lógica, detalhada na seção seguinte, consiste em comparar a saída pseudoaleatória com um limiar configurável, de modo a ajustar a taxa de ocorrência de pulsos. Esse mecanismo introduz estocasticidade compatível com as condições experimentais do LHC, em que nem todos os cruzamentos de feixe resultam em interações detectáveis.

- **Variação de amplitude:** a energia depositada em cada célula do TileCal, convertida em sinal elétrico, apresenta variações naturais. Para emular esse comportamento, os números pseudoaleatórios modulam a amplitude dos pulsos gerados, garantindo dispersão estatística em torno de valores médios. Porém, como a saída do MLCG é uniforme, ela não pode ser utilizada diretamente, uma vez que a resposta real do calorímetro segue distribuições de natureza exponencial. Assim, os números uniformes produzidos pelo gerador são transformados em valores de distribuição exponencial por meio de técnicas de conversão probabilística, discutidas em detalhe na Seção 4.2. Esse procedimento garante que a variabilidade da energia depositada em cada célula do TileCal seja reproduzida de forma realista.

A integração do gerador ao fluxo de dados foi projetada para se adaptar a diferentes cenários de síntese em hardware. Em algumas implementações, o módulo é incorporado diretamente ao processador SAPHO, fornecendo um valor aleatório a cada amostra processada. Em outras configurações, pode operar como um coprocessador dedicado, gerando um valor pseudoaleatório a cada ciclo de clock e disponibilizando-o aos blocos que necessitam de entrada estocástica. Essa flexibilidade permite balancear a ocupação de recursos lógicos da FPGA e a taxa de geração de números de acordo com os requisitos de desempenho da simulação.

4.1.5.1 *Impacto na fidelidade das simulações*

A fidelidade do simulador depende diretamente da qualidade estatística da sequência pseudoaleatória empregada. Caso a distribuição não fosse uniforme ou apresentasse padrões previsíveis, efeitos artificiais poderiam surgir nos sinais gerados, comprometendo a confiabilidade dos resultados.

Ao utilizar o MLCG com parâmetros adequados e pós-processamento dos bits mais significativos, garante-se dispersão satisfatória e ausência de correlações de curto alcance. Com isso, as flutuações de amplitude e a probabilidade de ocorrência de pulsos são reproduzidas de forma realista, permitindo que o simulador represente adequadamente as condições operacionais do TileCal.

Portanto, a adoção de um gerador pseudoaleatório eficiente, determinístico e de baixo custo de implementação foi essencial para assegurar que os resultados obtidos possuam validade científica, oferecendo uma base confiável para a avaliação de desempenho e para o desenvolvimento de técnicas de reconstrução de energia em ambientes de alta taxa de eventos.

4.2 TABELA EXPONENCIAL

A modelagem estatística da energia depositada em calorímetros de altas energias é essencial para compreender e reproduzir, em simulações, o comportamento real dos sinais eletrônicos. Ao atravessarem o Tile Calorimeter, as partículas provenientes das colisões no LHC interagem com a matéria por meio de processos de ionização, excitação e produção de chuveiros hadrônicos. Esses mecanismos apresentam forte caráter estocástico, de modo que a energia depositada em cada célula não é fixa, mas distribuída segundo leis probabilísticas (FABJAN; GIANOTTI, 2003; THOMSON, 2013).

De forma geral, a distribuição de energia observada em calorímetros apresenta caudas longas, nas quais eventos de baixa energia são muito mais frequentes que eventos de alta energia. Essa assimetria estatística torna inadequado o uso de modelos gaussianos simples, embora estes sejam úteis para descrever o ruído eletrônico do sistema de leitura. Para modelar a amplitude dos pulsos de energia, algumas distribuições são comumente consideradas:

- **Distribuição Exponencial:** descreve processos cuja probabilidade decai rapidamente com o aumento da variável de interesse. É simples, analiticamente tratável e captura a assimetria fundamental da deposição de energia em calorímetros. Por essa razão, é frequentemente utilizada como aproximação de primeira ordem.
- **Distribuição Gama:** generaliza a exponencial e permite ajustar melhor a soma de várias contribuições independentes de energia, característica dos chuveiros hadrônicos. Embora mais realista, sua implementação em hardware é mais custosa, pois exige cálculos adicionais de funções especiais.
- **Distribuição Gaussiana:** aplicável principalmente à modelagem do ruído eletrônico associado à instrumentação. Não é adequada para a distribuição da energia depositada, pois não representa corretamente a assimetria nem a cauda de altas energias.
- **Histogramas de Monte Carlo:** gerados a partir de simulações detalhadas, como as do *Geant4*, fornecem uma descrição altamente fiel da distribuição de energia. Contudo, sua implementação direta em FPGA exigiria grande quantidade de memória e não garantiria a reprodutibilidade e eficiência desejadas em arquiteturas embarcadas (JAMES, 1980).

Nesse contexto, a distribuição exponencial apresenta-se como um compromisso adequado entre realismo físico e viabilidade computacional. Ela reproduz a tendência estatística dominante das amplitudes de pulso no TileCal, ao mesmo tempo em que permite uma implementação eficiente por meio de tabelas discretizadas (*lookup tables*), como será detalhado nas próximas subseções.

4.2.1 Formulação da Distribuição Exponencial Contínua

A escolha da distribuição exponencial para modelar amplitudes de pulsos baseia-se em duas ideias fundamentais: (i) em muitos processos físicos com eventos “raros” ou com decaimento rápido de probabilidade nas caudas, a chance de observar valores grandes diminui aproximadamente de forma exponencial; (ii) quando apenas se sabe que a variável é não negativa e possui média finita, a exponencial é a distribuição de *máxima entropia* sob essas restrições, isto é, a menos informativa dentre as que respeitam tais condições. No contexto do TileCal, isso captura bem a predominância de pulsos pequenos e a raridade de amplitudes elevadas.

Se os incrementos de energia detectada podem ser pensados como ocorrências elementares aproximadamente independentes, com uma taxa média constante por unidade de escala (tempo, comprimento de caminho ou até uma grandeza ligada à energia), então o tempo até a próxima ocorrência segue uma **distribuição exponencial**. Em notação usual, diz-se que $X \sim \text{Exp}(\lambda)$, onde $\lambda > 0$ é o parâmetro de taxa. Essa interpretação ajuda a compreender por que a probabilidade de amplitudes grandes cai tão rapidamente: valores elevados requerem muitas contribuições acumuladas, um evento menos provável.

A **função densidade de probabilidade (f.d.p.)** descreve a distribuição de probabilidade de uma variável aleatória contínua. Em termos simples, ela mostra a probabilidade de a variável aleatória assumir um valor dentro de um determinado intervalo (ROSS, 2014). A área sob a curva da f.d.p. em um intervalo específico representa a probabilidade de a variável cair nesse intervalo. Sua descrição matemática é dada por:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0, \quad (4.5)$$

com $f(x) = 0$ para $x < 0$. Nessa expressão:

- x representa a variável contínua — aqui associada à amplitude de um pulso idealizado;
- $\lambda > 0$ é o parâmetro da distribuição, também chamado de *taxa de decaimento*. Ele controla a inclinação da curva: quanto maior λ , mais “curta” e concentrada próxima de zero é a distribuição; valores pequenos de λ espalham mais a probabilidade, permitindo pulsos de maior amplitude;
- o termo $e^{-\lambda x}$ é responsável pela cauda exponencial descendente, que garante que a probabilidade decai de forma suave à medida que x cresce;
- o fator multiplicativo λ assegura a **normalização**, ou seja, que a soma total das probabilidades seja igual a 1:

$$\int_0^{\infty} \lambda e^{-\lambda x} dx = 1.$$

A segunda função importante é a **função de distribuição acumulada (f.d.a.)**, que expressa a probabilidade de a variável assumir um valor até x :

$$F(x) = P(X \leq x) = \int_0^x f(t) dt. \quad (4.6)$$

No caso da distribuição exponencial, substituindo $f(t) = \lambda e^{-\lambda t}$, obtemos:

$$F(x) = \int_0^x \lambda e^{-\lambda t} dt. \quad (4.7)$$

A resolução da integral é simples: a primitiva de $e^{-\lambda t}$ é $-\frac{1}{\lambda}e^{-\lambda t}$. Assim,

$$F(x) = \left[-e^{-\lambda t} \right]_0^x = 1 - e^{-\lambda x}, \quad x \geq 0. \quad (4.8)$$

Esse resultado mostra que valores pequenos de x são altamente prováveis (pois $F(x)$ cresce rapidamente perto da origem), enquanto a probabilidade de valores elevados é cada vez menor. Em termos físicos, $F(x)$ quantifica a fração de pulsos com amplitude inferior a um determinado limiar.

Por fim, para transformar números pseudoaleatórios uniformes em valores que sigam a distribuição exponencial, utiliza-se a **função inversa da f.d.a.**. Dado um número aleatório u uniformemente distribuído no intervalo $(0,1)$, o valor correspondente de x pode ser obtido por:

$$x = F^{-1}(u) = -\frac{1}{\lambda} \ln(1 - u). \quad (4.9)$$

Essa fórmula é central para a simulação: ela mapeia a saída de um gerador uniforme (como os discutidos na Seção 4.1) para uma variável com distribuição exponencial, respeitando o comportamento estatístico observado nos pulsos reais do TileCal.

Portanto, as três equações fundamentais — f.d.p. (4.5), f.d.a. (4.8) e inversa (4.9) — formam a base matemática que sustenta a modelagem exponencial. O próximo passo consiste em adaptar essa formulação contínua para um contexto discreto, compatível com tabelas de valores inteiros implementadas em FPGA.

4.2.2 Discretização da Distribuição

Na prática, embora a formulação da distribuição exponencial seja contínua, a implementação em hardware digital exige que ela seja representada de forma **discreta e quantizada**. Isso ocorre porque o gerador pseudoaleatório utilizado no simulador produz valores binários de 9 bits, o que corresponde a $N = 512$ possíveis índices de acesso. Esses índices são usados diretamente para endereçar uma **tabela exponencial**, que contém os valores discretizados da distribuição. Cada posição da tabela armazena uma amplitude de 12 bits, representando a saída quantizada correspondente.

4.2.2.1 Aproximação por intervalos discretos

A primeira etapa da discretização é dividir o eixo contínuo de possíveis amplitudes em N intervalos iguais, cada um de largura Δx . Assim, cada índice inteiro $k = 0, 1, \dots, N-1$ representa o intervalo

$$[k\Delta x, (k+1)\Delta x].$$

Assim, sendo, a probabilidade associada a cada valor discreto é obtida pela integral da f.d.p. contínua nesse intervalo:

$$P(X = k) = \int_{k\Delta x}^{(k+1)\Delta x} \lambda e^{-\lambda x} dx.$$

Essa integral pode ser resolvida de forma exata:

$$P(X = k) = e^{-\lambda k\Delta x} (1 - e^{-\lambda\Delta x}).$$

O termo $e^{-\lambda k\Delta x}$ assegura a queda exponencial da distribuição, enquanto o fator $(1 - e^{-\lambda\Delta x})$ corrige o valor para o tamanho finito do intervalo.

4.2.2.2 Construção da tabela exponencial

Com base nessa formulação, é possível construir uma **tabela exponencial discreta**, na qual cada índice k é associado a uma amplitude quantizada A_k . Uma forma prática de obter esses valores é utilizar a função inversa da distribuição acumulada (método da inversa), definida por:

$$F^{-1}(u) = -\frac{1}{\lambda} \ln(1 - u), \quad u \in (0, 1).$$

Ao associar cada valor discreto k à fração $u = \frac{k}{N}$, obtém-se:

$$A_k \approx F^{-1}\left(\frac{k}{N}\right) = -\frac{1}{\lambda} \ln\left(1 - \frac{k}{N}\right).$$

Os valores A_k calculados dessa forma são então arredondados e armazenados como inteiros de 12 bits, prontos para serem acessados em hardware.

4.2.2.3 Vantagens da discretização

A adoção da versão discretizada da exponencial apresenta três vantagens principais:

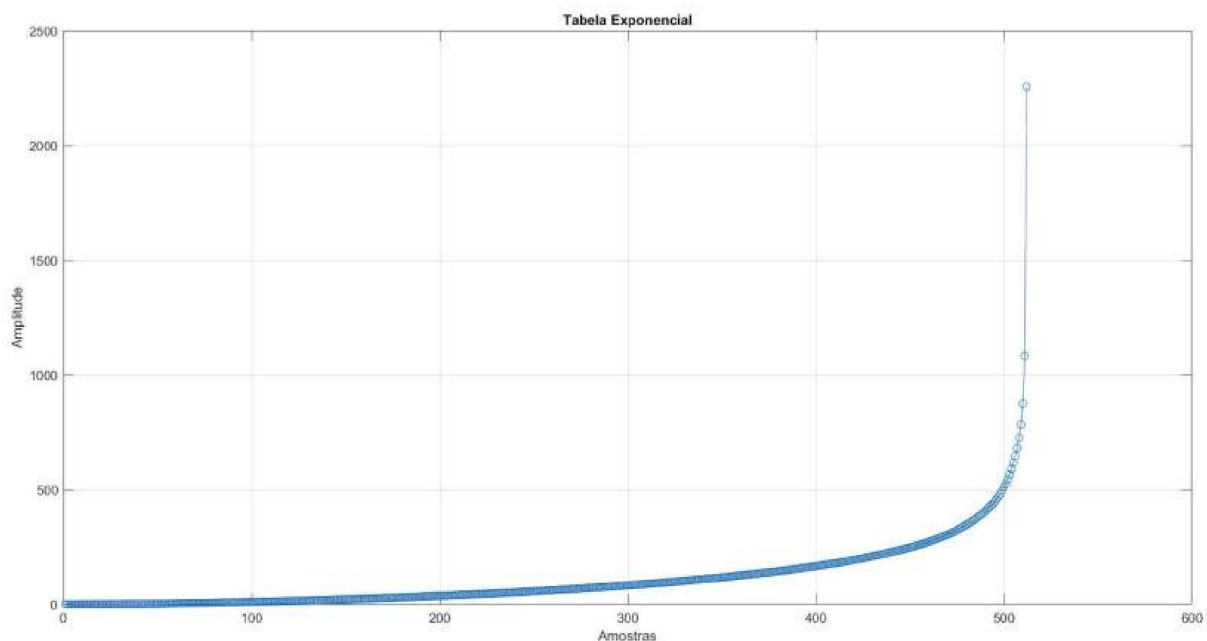
- **Eficiência computacional:** evita o cálculo de funções logarítmicas ou exponenciais em tempo de execução, reduzindo drasticamente o custo em FPGA.
- **Controle estatístico:** permite definir de maneira explícita o conjunto de amplitudes possíveis e sua frequência relativa, garantindo a coerência com a distribuição contínua.

- **Flexibilidade:** a tabela pode ser recalculada com diferentes valores de λ ou resoluções de bits, adaptando-se facilmente a novos cenários de simulação.

Assim, a discretização viabiliza a implementação realista da distribuição exponencial dentro do ambiente SAPHO, conciliando fidelidade estatística com as restrições de arquitetura e síntese em FPGA.

A Figura 18 ilustra um gráfico da tabela calculada, destacando visualmente a característica exponencial da distribuição e a concentração de valores mais baixos nas primeiras posições. Nessa tabela, o valor máximo é 2255 e o número de posições na tabela é 512, ou seja, 9 bits.

Figura 18 – Amplitudes e valores na tabela exponencial.



Fonte: (Elaborado pelo autor (2025).)

4.2.3 Taxa de Ocupação e Fluxo Geral do Bloco Exponencial

A implementação da tabela exponencial no simulador não é utilizada de forma isolada: é necessário também reproduzir o padrão estatístico de ocorrência de eventos característico do TileCal. Nos dados experimentais, observa-se que a grande maioria dos instantes de amostragem não contém deposição de energia relevante em uma célula. Aproximadamente 90% das amostras são nulas, ou seja, não há pulso registrado. Apenas em cerca de 10% dos casos ocorre a geração de um pulso mensurável. Essa proporção — chamada de **taxa de ocupação** — é um parâmetro essencial para que o simulador reproduza com fidelidade o comportamento real do detector.

Assim, o bloco da tabela exponencial foi estruturado em duas etapas principais, ambas dependentes de números pseudoaleatórios independentes (gerados conforme a Seção 4.1). Essas etapas são:

1. **Decisão de ocorrência do evento (pulso ou zero).** O primeiro número pseudoaleatório, R , é utilizado para decidir se naquele ciclo haverá ou não um pulso. Considerando que R é um valor inteiro gerado em 10 bits ($0 \leq R \leq 1023$), calcula-se a razão:

$$\frac{R}{R_{\max}},$$

onde $R_{\max} = 1023$ é o valor máximo possível. Essa razão é um número entre 0 e 1, que pode ser comparado a um limiar α que representa a taxa de ocupação. No caso deste trabalho, adotou-se $\alpha = 0,1$, correspondente a uma taxa de 10%. Dessa forma:

$$\frac{R}{R_{\max}} < \alpha \quad \Rightarrow \quad \text{evento gerado},$$

$$\frac{R}{R_{\max}} \geq \alpha \quad \Rightarrow \quad \text{saída nula}.$$

Isso significa que, em média, 90% dos ciclos terão saída igual a zero, e apenas 10% dos ciclos prosseguirão para a geração de um pulso. Essa etapa garante que o simulador não produza pulsos em excesso, preservando a mesma densidade estatística observada em condições reais de operação do TileCal.

2. **Determinação da amplitude do evento não nulo.** Caso a condição anterior indique que um pulso deve ser gerado, um segundo número pseudoaleatório é produzido. Esse número, agora com 9 bits ($0 \leq k \leq 511$), é utilizado como índice de acesso direto à `tabela_exponencial`, que contém 512 endereços e valores pré-calculados, conforme a Subsubseção 4.2.2.2.

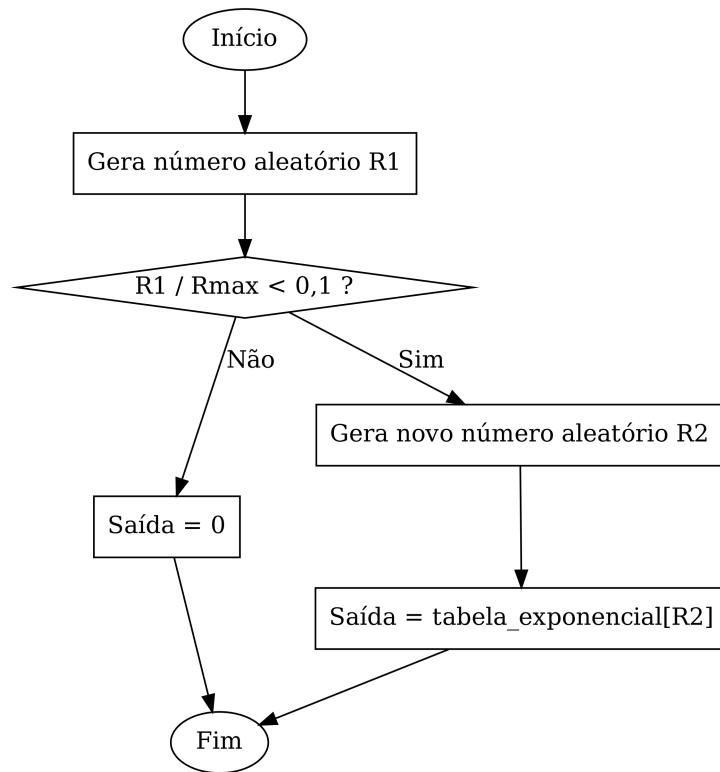
Cada posição k da tabela armazena um valor inteiro de 12 bits, A_k , correspondente a uma amplitude de pulso coerente com a distribuição exponencial discretizada discutida na Seção anterior. O acesso à memória é imediato, de modo que a operação consiste apenas em selecionar o valor armazenado e atribuí-lo à saída. Essa estratégia evita cálculos em tempo de execução e assegura que as amplitudes simuladas sigam rigorosamente a distribuição estatística desejada.

Um exemplo numérico ilustra o funcionamento: suponha que o primeiro número aleatório seja $R = 72$. O cálculo fornece $72/1023 \approx 0.070$, que é menor que $\alpha = 0.1$. Portanto, o simulador decide que um evento deve ser gerado. Em seguida, o segundo número pseudoaleatório resulta em $k = 32$. O simulador então acessa a posição `tabela_exponencial[32]`, cujo conteúdo (um valor inteiro de 12 bits) é atribuído como a amplitude final do pulso naquele ciclo. Se, por outro lado, o primeiro número tivesse

resultado em $R = 700$, a condição $700/1023 \approx 0.684 > 0.1$ levaria a uma saída nula, encerrando o processo.

A Figura 19 apresenta o fluxograma correspondente, que resume graficamente esse processo. O diagrama destaca as etapas de decisão: a comparação do primeiro número com a taxa de ocupação, a bifurcação entre saída nula e evento aceito, e finalmente o acesso à tabela exponencial para definição da amplitude. Essa sequência evidencia como o simulador combina a **estatística de ocupação** com a **distribuição de amplitudes**, resultando em uma representação fiel dos sinais eletrônicos observados no TileCal.

Figura 19 – Fluxo geral de decisão do bloco exponencial.



Fonte: (Elaborado pelo autor (2025).)

A implementação da tabela exponencial permitiu unir duas características estatísticas fundamentais dos sinais observados no TileCal: a predominância de amostras nulas e a distribuição assimétrica das amplitudes não nulas. Por meio da introdução de uma etapa inicial de verificação da taxa de ocupação, garante-se que cerca de 90% dos ciclos correspondam a valores iguais a zero, refletindo a baixa probabilidade de deposição de energia significativa em uma célula. Já para os 10% de casos em que um pulso é aceito, a amplitude é determinada diretamente a partir de uma tabela discreta construída com base na distribuição exponencial contínua, preservando a assimetria natural entre pulsos

pequenos e raros pulsos de maior intensidade.

Essa abordagem equilibra rigor estatístico e eficiência digital: as propriedades matemáticas da exponencial são incorporadas à lógica de hardware sem necessidade de cálculos complexos em tempo de execução, sendo reduzidas a operações simples de comparação e acesso à memória. Além disso, a utilização de números pseudoaleatórios com *seed* fixa assegura reprodutibilidade dos resultados, condição indispensável para simulações de Física de Altas Energias.

Dessa forma, o bloco da tabela exponencial cumpre papel central no simulador, fornecendo amplitudes compatíveis com a realidade do detector e garantindo que os cenários de ruído e ocupação implementados sejam representativos do ambiente de altas taxas de eventos que será encontrado na Fase 2 do LHC.

4.3 MODELAGEM DE PULSOS

A forma de pulso observada no Tile Calorimeter não é simplesmente o sinal bruto proveniente dos tubos fotomultiplicadores (PMTs), mas sim o resultado do processamento realizado pela eletrônica de *front-end*. Quando uma partícula atravessa o calorímetro e deposita energia em uma célula, a luz coletada pelo PMT é convertida em um pulso de corrente extremamente rápido, com largura de poucos nanossegundos. Esse pulso, ao ser aplicado ao circuito de *shaping*, é transformado em uma resposta temporal bem definida, característica do TileCal.

Do ponto de vista matemático, essa forma de onda corresponde à **resposta ao impulso** do sistema de modelagem analógica. O circuito de *shaping*, implementado como uma rede de resistores, capacitores e indutores (RLC), define os polos e zeros que moldam a resposta temporal. O resultado é um sinal com três regiões distintas: uma **subida rápida**, associada às constantes de tempo mais curtas; um **pico em torno de 50–75 ns**, que representa a máxima transferência de carga; e uma **cauda exponencial lenta**, ligada às constantes de tempo mais longas do circuito. Essa resposta é a assinatura típica dos pulsos do TileCal e a base para todas as etapas posteriores de reconstrução de energia.

A modelagem matemática desse comportamento é, portanto, essencial para qualquer simulador realista. No contexto deste trabalho, em que se busca implementar uma versão digital da resposta do calorímetro em um processador embarcado, é necessário representar a função de transferência equivalente do shaper e traduzi-la para estruturas discretas compatíveis com FPGA. Dessa forma, torna-se possível reproduzir não apenas a forma individual de cada pulso, mas também efeitos mais complexos, como o *pile-up*, resultante da sobreposição de múltiplos sinais em intervalos curtos de tempo.

Assim, o objetivo desta seção é apresentar o modelo matemático que descreve a resposta do TileCal, partindo da função de transferência analógica global, de ordem

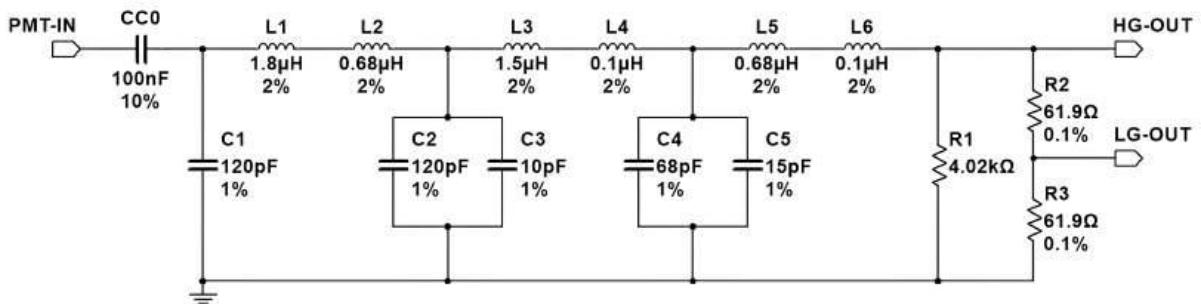
elevada, até sua decomposição e implementação em filtros digitais discretos de primeira e segunda ordem, utilizados no simulador desenvolvido.

4.3.1 Circuito equivalente do Shaper (FENICS)

O condicionamento do sinal no TileCal é realizado pelo circuito de *shaping* presente nas placas FENICS. Esse estágio eletrônico tem a função de transformar o pulso ultrarrápido de corrente, produzido pelo tubo fotomultiplicador (PMT), em uma forma de onda mais larga e estável, adequada para digitalização e processamento posterior. A modelagem matemática desse circuito pode ser feita a partir de uma rede equivalente composta por resistores, capacitores e indutores (RLC), cuja topologia foi projetada para reproduzir a forma típica dos pulsos observados experimentalmente (BONNEFOY; TEAM, 2021).

A Figura 20 apresenta o diagrama esquemático simplificado do circuito equivalente do shaper. Embora cada componente tenha papel específico no comportamento dinâmico do sinal, é a combinação dos elementos que define a resposta global do sistema. Os **capacitores** são responsáveis por controlar a largura do pulso, armazenando e liberando carga em diferentes escalas de tempo. Os **indutores** introduzem efeitos de atraso e amortecimento da cauda, suavizando transições rápidas. Já os **resistores** ajustam o ganho, determinam o fator de amortecimento da rede e implementam a técnica de *pole-zero cancellation* (PZC), fundamental para suprimir componentes indesejadas e estabilizar a resposta. Em conjunto, esses elementos configuram um sistema de ordem elevada, capaz de moldar o pulso em três regiões distintas: subida rápida, pico em torno de 50–75 ns e cauda exponencial lenta.

Figura 20 – Circuito equivalente RLC do shaper implementado no FENICS.



Fonte: (LUNA *et al.*, 2024).

O comportamento do circuito RLC pode ser descrito por equações diferenciais lineares que relacionam a entrada $x(t)$ (corrente do PMT) e a saída $y(t)$. Resolver essas equações diretamente no domínio do tempo é impraticável devido à ordem elevada do sistema. Para simplificar a análise, aplica-se a **transformada de Laplace**, que converte derivadas temporais em multiplicações pela variável complexa s :

$$\mathcal{L} \left\{ \frac{d^n y(t)}{dt^n} \right\} = s^n Y(s) \quad (4.10)$$

onde $Y(s)$ é a transformada de Laplace da saída $y(t)$. Esse procedimento permite reescrever o sistema de equações diferenciais como uma relação algébrica entre polinômios em s .

Define-se então a **função de transferência** como:

$$H(s) = \frac{Y(s)}{X(s)}, \quad (4.11)$$

que representa a resposta em frequência do circuito para qualquer entrada. Essa função encapsula completamente o comportamento dinâmico do sistema, incluindo polos (associados às constantes de tempo do circuito) e zeros (resultantes de cancelamentos implementados no projeto).

4.3.1.1 Função de transferência global do shaper

A aplicação da análise em Laplace ao circuito RLC do FENICS resulta em uma função de transferência de ordem elevada, típica de sistemas analógicos de modelagem de pulsos. No caso do shaper do TileCal, obtém-se uma função racional com polinômio numerador de ordem 7 e denominador de ordem 8, refletindo a presença de múltiplos polos e zeros:

$$H(s) = \frac{8,59 \cdot 10^{27} s^7 + 1,84 \cdot 10^{30} s^6 + 1,83 \cdot 10^{32} s^5 + 1,09 \cdot 10^{34} s^4 + 4,05 \cdot 10^{35} s^3 + 8,9 \cdot 10^{36} s^2 + 1,01 \cdot 10^{38} s + 4,67 \cdot 10^{38}}{s^8 + 2,33 \cdot 10^8 s^7 + 2,59 \cdot 10^{16} s^6 + 1,84 \cdot 10^{24} s^5 + 8,55 \cdot 10^{31} s^4 + 2,7 \cdot 10^{39} s^3 + 5,57 \cdot 10^{46} s^2 + 6,88 \cdot 10^{53} s + 3,67 \cdot 10^{60}} \quad (4.12)$$

Essa equação descreve a resposta global do shaper analógico. A ordem elevada garante fidelidade ao comportamento físico real do TileCal, mas ao mesmo tempo impõe desafios para implementação direta em hardware digital, exigindo técnicas de simplificação, como a decomposição em filtros de ordem reduzida e a aplicação da transformada bilinear, discutidas nas subseções seguintes.

4.3.2 Decomposição em filtros parciais

A função de transferência global $H(s)$ do circuito de *shaping* do TileCal, apresentada na Eq. 4.12, descreve de forma completa a resposta temporal do sistema. Essa representação de ordem elevada é precisa, mas abre espaço para diferentes estratégias de implementação, que foram exploradas em simuladores distintos desenvolvidos neste trabalho. Em alguns

casos, optou-se pela utilização direta da equação combinada em sua forma discreta (Simuladores 2 e 3). Em outros, buscou-se reduzir a complexidade aparente da equação por meio da **expansão em frações parciais**, resultando em filtros de baixa ordem que, combinados, reproduzem a mesma resposta global (Simuladores 1 e 4).

A expansão em frações parciais é uma técnica clássica que permite reescrever uma função racional de alta ordem como a soma de termos simples associados a polos reais e complexos. No caso de $H(s)$, essa decomposição leva naturalmente à seguinte forma geral:

$$H(s) = \sum_{i=1}^3 \frac{\alpha_i}{s + p_i} + \sum_{j=1}^2 \frac{\beta_j s + \gamma_j}{s^2 + a_j s + b_j}, \quad (4.13)$$

onde:

- os termos de primeira ordem $\frac{\alpha_i}{s + p_i}$ representam filtros simples, associados a polos reais do sistema;
- os termos de segunda ordem $\frac{\beta_j s + \gamma_j}{s^2 + a_j s + b_j}$ representam pares de polos complexos conjugados.

Aplicando esse procedimento à função global do TileCal, obtêm-se cinco filtros independentes, listados a seguir:

$$H_1(s) = \frac{2,3 \times 10^7}{s + 4 \times 10^7}, \quad (4.14)$$

$$H_2(s) = \frac{-3,1 \times 10^7 \cdot s - 8,4 \times 10^{14}}{s^2 + 7,2 \times 10^7 \cdot s + 5,1 \times 10^{15}}, \quad (4.15)$$

$$H_3(s) = \frac{-7,7 \times 10^2}{s + 1,1 \times 10^9}, \quad (4.16)$$

$$H_4(s) = \frac{8,2 \times 10^6 \cdot s - 8,7 \times 10^{13}}{s^2 + 5 \times 10^7 \cdot s + 1,4 \times 10^{16}}, \quad (4.17)$$

$$H_5(s) = \frac{-3,5 \times 10^4}{s + 8,8 \times 10^4}. \quad (4.18)$$

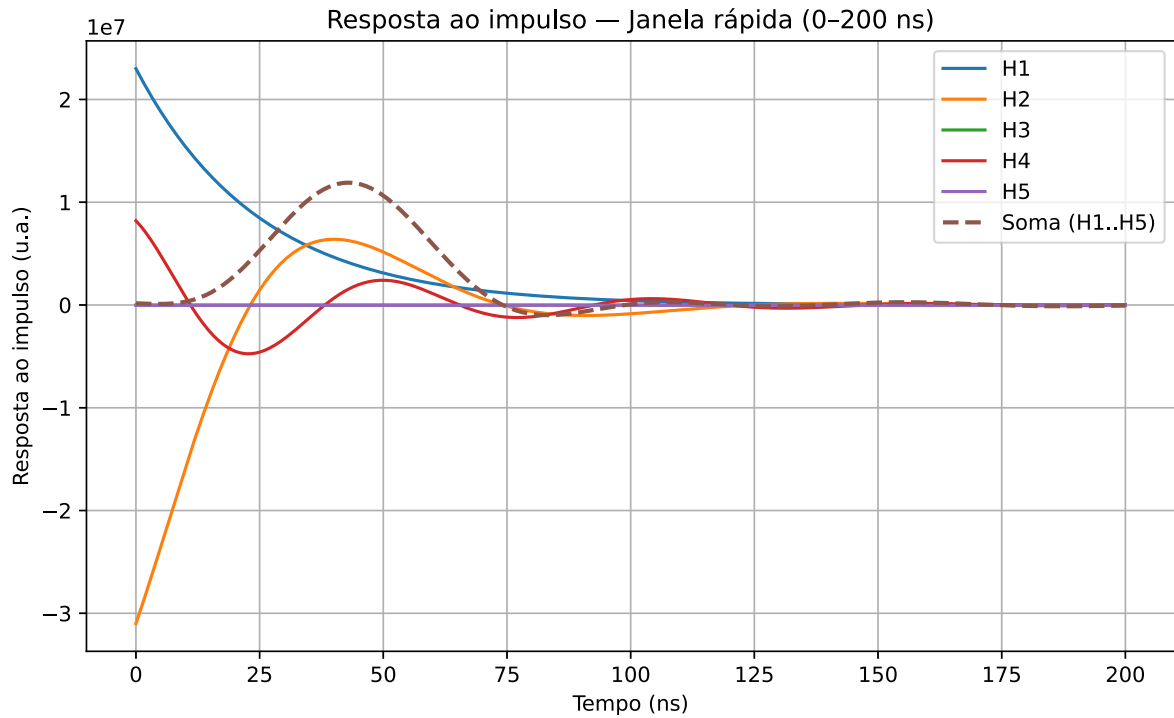
Cada um desses filtros parciais corresponde a uma componente distinta da resposta temporal:

- $H_1(s)$ e $H_3(s)$: filtros de primeira ordem responsáveis pela **subida rápida** e pelas componentes mais instantâneas do pulso;
- $H_2(s)$ e $H_4(s)$: filtros de segunda ordem que introduzem memória adicional, modelando a região do **platô** e parte da **cauda**;

- $H_5(s)$: filtro de primeira ordem associado à **descarga lenta**, caracterizando a **cauda exponencial**.

A combinação das saídas desses cinco filtros reconstitui a forma de pulso típica observada experimentalmente no TileCal, como pode ser visto na Figura 21. Essa decomposição facilita tanto a análise matemática quanto a implementação modular em arquiteturas digitais. Nos simuladores em que se optou pelo uso direto da equação global (Simuladores 2 e 3), manteve-se a integridade da função $H(s)$ em uma única estrutura, ao passo que nos casos com frações parciais (Simuladores 1 e 4), o modelo foi dividido em blocos elementares, cada qual traduzido individualmente para o domínio discreto.

Figura 21 – Resposta ao Impulso das 5 equações de filtro.



Fonte: Elaborado pelo Autor (2025)

4.3.3 Discretização dos filtros

A implementação digital do modelo requer a conversão do domínio analógico (s) para o domínio discreto (z), preservando a estabilidade e a forma espectral útil do *shaper*. Nesta dissertação adotou-se a **transformada bilinear** (método de Tustin), cuja correspondência é

$$s \longleftarrow \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}, \quad (4.19)$$

com período de amostragem $T = 25 \text{ ns}$ ($f_s = 40 \text{ MHz}$). A bilinear mapeia todo o semiplano esquerdo de s para o interior do círculo unitário em z , garantindo que **sistemas estáveis no contínuo permaneçam estáveis no discreto**. O preço é a *distorção de frequência* (*frequency warping*), descrita pela relação

$$\Omega = \frac{2}{T} \tan\left(\frac{\omega}{2}\right) \quad \Longleftrightarrow \quad \omega = 2 \arctan\left(\frac{\Omega T}{2}\right),$$

onde Ω é frequência analógica (rad/s) e ω é frequência digital (rad/amostra). Quando necessário, emprega-se pré-distorção (prewarping) em frequências de interesse para casar pontos-chave do espectro.

Após a decomposição (Seção 4.3.2), obtêm-se cinco termos:

$$H_1(s), H_2(s), H_3(s), H_4(s), H_5(s),$$

com três filtros de primeira ordem e dois de segunda ordem. Cada termo é discretizado individualmente por (4.19), resultando em IIRs curtos (biquads). A seguir apresentamos fórmulas fechadas úteis para derivação/checagem e, em seguida, o procedimento em MATLAB.

(i) Termos de primeira ordem.

Para um termo genérico:

$$H_{1a}(s) = \frac{K}{s + p},$$

a bilinear com $K_T \equiv 2/T$ leva a:

$$H_{1a}(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}, \quad \text{com} \quad a_1 = \frac{\alpha - 1}{\alpha + 1}, \quad b_0 = b_1 = \frac{K}{K_T(\alpha + 1)},$$

onde $\alpha = \frac{pT}{2} = \frac{p}{K_T}$. Aplicando aos seus termos:

$$H_1(s) = \frac{2,3 \times 10^7}{s + 4 \times 10^7}, \quad H_3(s) = \frac{-7,7 \times 10^2}{s + 1,1 \times 10^9}, \quad H_5(s) = \frac{-3,5 \times 10^4}{s + 8,8 \times 10^4}.$$

Para cada um, compute α , depois a_1 e $b_0 = b_1$ como acima.

(ii) Termos de segunda ordem.

Para um termo genérico

$$H_{2a}(s) = \frac{b_1 s + b_0}{s^2 + a_1 s + a_0},$$

a substituição $s = K_T \frac{1-z^{-1}}{1+z^{-1}}$ e a multiplicação por $(1 + z^{-1})^2$ conduzem ao biquad

$$H_{2a}(z) = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2}}{D_0 + D_1 z^{-1} + D_2 z^{-2}},$$

com coeficientes analítico-fechados:

$$\begin{aligned} D_0 &= a_0 + a_1 K_T + K_T^2, & D_1 &= 2(a_0 - K_T^2), & D_2 &= a_0 - a_1 K_T + K_T^2, \\ B_0 &= b_0 + b_1 K_T, & B_1 &= 2(b_0 - b_1 K_T), & B_2 &= b_0 - b_1 K_T. \end{aligned}$$

Normalizando por D_0 (i.e., dividindo todos por D_0), obtêm-se

$$\boxed{\begin{aligned} b_0^{(z)} &= \frac{B_0}{D_0}, & b_1^{(z)} &= \frac{B_1}{D_0}, & b_2^{(z)} &= \frac{B_2}{D_0}, \\ a_1^{(z)} &= \frac{D_1}{D_0}, & a_2^{(z)} &= \frac{D_2}{D_0}, \end{aligned}}$$

definindo o biquad $\frac{b_0^{(z)} + b_1^{(z)} z^{-1} + b_2^{(z)} z^{-2}}{1 + a_1^{(z)} z^{-1} + a_2^{(z)} z^{-2}}$. Aplique essas fórmulas aos seus $H_2(s)$ e $H_4(s)$:

$$\begin{aligned} H_2(s) &= \frac{-3,1 \times 10^7 s - 8,4 \times 10^{14}}{s^2 + 7,2 \times 10^7 s + 5,1 \times 10^{15}}, \\ H_4(s) &= \frac{8,2 \times 10^6 s - 8,7 \times 10^{13}}{s^2 + 5 \times 10^7 s + 1,4 \times 10^{16}}. \end{aligned}$$

Procedimento em MATLAB (termo a termo).

```
fs = 40e6;    % Hz
```

```
% Exemplo para H2(s):
```

```
b2 = [-3.1e7, -8.4e14];
```

```
a2 = [1, 7.2e7, 5.1e15];
```

```
[b2z, a2z] = bilinear(b2, a2, fs);
```

```
% Para H1, H3, H5 (1a ordem):
```

```
b1 = [2.3e7]; a1 = [1, 4e7];
```

```
[b1z, a1z] = bilinear(b1, a1, fs);
```

```
% Repetir para H3, H4, H5 e depois somar as respostas:
```

```
% H_total(z) = H1(z)+H2(z)+H3(z)+H4(z)+H5(z)
```

No simulador, cada $H_i(z)$ é implementado como um bloco IIR independente (1^a ou 2^a ordem). A **soma das saídas** produz a resposta total do canal.

Aplicando a transformada bilinear, conforme apresentado, as funções de transferência analógicas $H_i(s)$ foram convertidas para o domínio digital $H_i(z)$. As equações a seguir já estão normalizadas com o denominador na forma $1 + a_1 z^{-1} + a_2 z^{-2}$:

$$H_1(z) = \frac{0.1917 + 0.1917 z^{-1}}{1 - 0.3333 z^{-1}} \quad (4.20)$$

$$H_2(z) = \frac{-0.1924 - 0.0973 z^{-1} + 0.0950 z^{-2}}{1 - 0.1506 z^{-1} + 0.3326 z^{-2}} \quad (4.21)$$

$$H_3(z) = \frac{-6.53 \times 10^{-7} - 6.53 \times 10^{-7} z^{-1}}{1 + 0.8644 z^{-1}} \quad (4.22)$$

$$H_4(z) = \frac{0.0233 - 0.0071 z^{-1} - 0.0305 z^{-2}}{1 + 0.6230 z^{-1} + 0.6721 z^{-2}} \quad (4.23)$$

$$H_5(z) = \frac{-4.37 \times 10^{-4} - 4.37 \times 10^{-4} z^{-1}}{1 - 0.9978 z^{-1}} \quad (4.24)$$

Cada filtro $H_i(z)$ pode ser implementado diretamente como uma equação de diferenças no processador SAPHO ou em FPGA, utilizando apenas multiplicações e somas com os coeficientes apresentados. Os filtros H_1 , H_3 e H_5 são de primeira ordem, enquanto H_2 e H_4 correspondem a filtros de segunda ordem (biquads).

Para a obtenção da função unificada discretizada, da forma como é descrita na Equação 4.26, é necessário realizar a soma das 5 funções transformadas previamente:

$$H(z) = H_1(z) + H_2(z) + H_3(z) + H_4(z) + H_5(z). \quad (4.25)$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-m}}{1 + a_1 z^{-1} + \dots + a_n z^{-n}}. \quad (4.26)$$

Logo, dessa forma, mostra-se possível a obtenção dos coeficientes das equações utilizadas para modelagem do pulso do TileCal, etapa essencial para o simulador como um todo.

4.3.4 Equações de diferenças e implementação no simulador

Cada filtro é definido por uma equação de diferenças no tempo discreto, envolvendo coeficientes no numerador (b_0, b_1, b_2) e no denominador (a_1, a_2), calculados diretamente no MATLAB a partir das funções apresentadas anteriormente. Esses coeficientes definem a influência dos valores atuais e passados da entrada e da saída sobre o valor presente do sinal simulado.

O sinal de entrada $x[n]$, obtido por sorteio pseudoaleatório conforme descrito na Seção 4.1, é processado individualmente por cada filtro, gerando saídas parciais $y_1[n]$ a $y_5[n]$. A composição dessas saídas resulta em $y_6[n]$, que representa a forma temporal completa do pulso, já incluindo efeitos de sobreposição (*pile-up*).

Os filtros obtidos após discretização correspondem a três sistemas de primeira ordem (H_1, H_3, H_5) e dois de segunda ordem (H_2, H_4). Suas equações de diferenças são:

$$y_1[n] = b_{10} \cdot x[n] + b_{11} \cdot x[n-1] - a_{11} \cdot y_1[n-1], \quad (4.27)$$

$$y_2[n] = b_{20} \cdot x[n] + b_{21} \cdot x[n-1] + b_{22} \cdot x[n-2] - a_{21} \cdot y_2[n-1] - a_{22} \cdot y_2[n-2], \quad (4.28)$$

$$y_3[n] = b_{30} \cdot x[n] + b_{31} \cdot x[n-1] - a_{31} \cdot y_3[n-1], \quad (4.29)$$

$$y_4[n] = b_{40} \cdot x[n] + b_{41} \cdot x[n-1] + b_{42} \cdot x[n-2] - a_{41} \cdot y_4[n-1] - a_{42} \cdot y_4[n-2], \quad (4.30)$$

$$y_5[n] = b_{50} \cdot x[n] + b_{51} \cdot x[n-1] - a_{51} \cdot y_5[n-1]. \quad (4.31)$$

Onde:

- $x[n]$ representa o valor atual da entrada;
- $x[n-1], x[n-2]$ são as amostras anteriores da entrada;
- $y_k[n-1], y_k[n-2]$ são os estados passados do filtro k ;
- b_{ki} e a_{kj} são coeficientes do numerador e denominador, obtidos via discretização.

Ademais, pode-se escrever a saída global do sistema como a soma algébrica das contribuições parciais:

$$y[n] = y_1[n] + y_2[n] + y_3[n] + y_4[n] + y_5[n]. \quad (4.32)$$

Uma alternativa equivalente é a implementação direta da forma global $H(z)$, já combinada em um único filtro de ordem superior. Nesse caso, a equação de diferenças envolve mais termos de atraso, tanto de entrada quanto de saída:

$$\begin{aligned} y[n] = & b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] + b_3 \cdot x[n-3] \\ & + b_4 \cdot x[n-4] + b_5 \cdot x[n-5] + b_6 \cdot x[n-6] + b_7 \cdot x[n-7] \\ & - (a_1 \cdot y[n-1] + a_2 \cdot y[n-2] + a_3 \cdot y[n-3] + a_4 \cdot y[n-4] \\ & + a_5 \cdot y[n-5] + a_6 \cdot y[n-6] + a_7 \cdot y[n-7]). \end{aligned} \quad (4.33)$$

Essa formulação concentra todo o efeito do *shaper* em uma única equação, reduzindo o número de blocos lógicos, mas aumentando a ordem e, conseqüentemente, a complexidade numérica. Por outro lado, a decomposição em filtros parciais facilita a implementação em FPGA ou em processadores com recursos limitados, já que cada equação envolve apenas dois ou três atrasos. Os coeficientes específicos para as implementações são detalhados no Capítulo 5.

Na prática, as variáveis $x[n]$, $x[n-1]$, $y_k[n-1]$, etc., são mantidas em registradores e atualizadas a cada ciclo. Cada multiplicação e soma representa uma operação elementar que pode ser paralelizada no FPGA ou sequencializada no SAPHO. Assim:

- A forma global concentra mais multiplicações em uma única equação;
- A forma parcial distribui o custo entre filtros independentes de 1^a e 2^a ordem, permitindo uso eficiente de biquads.

Em ambos os casos, o modelo é capaz de reproduzir não apenas a forma típica de um único pulso, mas também a sobreposição de múltiplos sinais em sequência, validando algoritmos de reconstrução de energia em condições realistas de *pile-up*.

O processo de modelagem de pulsos do TileCal, iniciado a partir da descrição analógica do circuito de *shaping*, permitiu obter a função de transferência global $H(s)$ e suas representações equivalentes em frações parciais. A aplicação da transformada bilinear resultou em versões discretizadas adequadas para implementação digital, tanto na forma unificada $H(z)$ quanto como um conjunto de filtros parciais de primeira e segunda ordem. As respectivas equações de diferenças estabelecem o elo entre a teoria contínua e a execução prática no processador SAPHO ou em FPGA. Assim, a seção fornece a base matemática e computacional necessária para o desenvolvimento dos simuladores apresentados no capítulo seguinte.

5 SIMULADORES

A etapa de simulação desempenha um papel central na presente pesquisa, sendo o elo entre a concepção metodológica apresentada no capítulo anterior e a avaliação prática da viabilidade de um sistema completo capaz de gerar, processar e reconstruir sinais compatíveis com os observados no Tile Calorimeter (TileCal). Os simuladores descritos neste capítulo integram todos os blocos funcionais desenvolvidos — como o gerador de números pseudoaleatórios, o módulo de filtragem exponencial com taxa de ocupação controlada e a modelagem matemática do pulso — a fim de formar estruturas completas, testáveis e potencialmente implementáveis em dispositivos reconfiguráveis como FPGAs.

Cada simulador representa uma abordagem distinta para o problema da simulação em ambientes de alta taxa de eventos. A evolução entre as versões reflete decisões de projeto que envolvem desempenho computacional, ocupação de recursos lógicos, paralelismo e viabilidade de síntese. Desde versões mais simplificadas, com foco em funcionalidade básica e validação da arquitetura conceitual, até configurações otimizadas para compatibilidade com restrições reais de tempo e espaço — como aquelas exigidas por aplicações no trigger de nível 1 do experimento ATLAS —, o desenvolvimento progressivo dos simuladores permitiu analisar e balancear os compromissos envolvidos.

No contexto da simulação de sinais eletrônicos do TileCal, a amplitude do pulso de saída não segue uma distribuição uniforme, mas sim exponencial, com forte tendência a valores baixos. Adicionalmente, há uma predominância significativa de eventos com amplitude nula, o que representa fisicamente a ausência de interação nas células do calorímetro em determinada janela temporal. Para refletir esse comportamento estatístico realista, foi implementado um controle de taxa de ocupação, fixado em 10%. Isso significa que apenas 10% dos eventos gerados prosseguem para a etapa de filtragem, enquanto os demais resultam em valor nulo. Tal abordagem foi fundamental para garantir a aderência estatística dos dados simulados às características físicas do TileCal, principalmente no que se refere ao impacto do ruído eletrônico e à estrutura temporal dos pulsos.

Com base nos blocos apresentados no Capítulo 4, o fluxo funcional completo dos simuladores pode ser descrito em quatro estágios: (1) geração de número pseudoaleatório uniforme, (2) aplicação de critério de ocupação, (3) mapeamento exponencial via tabela precomputada, e (4) modelagem da forma de pulso com múltiplos filtros digitais. A forma final do pulso é gerada pela sobreposição da resposta de cada filtro, permitindo simular não apenas a morfologia individual de um evento isolado, mas também o fenômeno de empilhamento temporal (*pile-up*) — comum em ambientes com alta densidade de eventos.

Neste capítulo, cada uma das versões dos simuladores será descrita detalhadamente. As subseções a seguir (5.1 a 5.4) apresentam a motivação por trás de cada arquitetura, seus principais blocos constituintes, os recursos utilizados, os modos de operação e os principais

parâmetros ajustáveis. Sempre que pertinente, serão incluídos fluxogramas, representações esquemáticas, trechos de código e observações sobre a viabilidade de síntese e execução em tempo real. Ao final, no Capítulo 6, os resultados obtidos com cada uma das versões são comparados, permitindo avaliar o impacto das decisões de projeto no desempenho global do sistema.

A organização e descrição dos simuladores, portanto, não apenas consolida os elementos desenvolvidos na metodologia, como também proporciona uma análise crítica das possibilidades e limitações envolvidas na construção de sistemas digitais para instrumentação científica de alta complexidade. Este estudo, ao integrar simulação estatística, processamento digital de sinais e design em FPGA, contribui para o avanço de soluções compactas e eficientes voltadas a ambientes de aquisição de dados com elevadas taxas de eventos, como os encontrados no LHC.

5.1 SIMULADOR 1

O desenvolvimento do primeiro simulador teve como principal objetivo validar, de forma modular e incremental, os blocos funcionais fundamentais que compõem o sistema completo de simulação de pulsos compatíveis com os sinais observados no Tile Calorimeter (TileCal). Por se tratar da primeira versão funcional do simulador, esta implementação foi concebida com foco na simplicidade estrutural, facilitando a verificação individual de cada etapa da cadeia de processamento e servindo como prova de conceito para as fases subsequentes do projeto.

A motivação para a construção desta versão inicial está diretamente associada à necessidade de verificar a compatibilidade entre os blocos projetados na metodologia — como o gerador pseudoaleatório, a tabela exponencial e os filtros digitais — e sua integração no processador dedicado SAPHO. Ao adotar uma abordagem com apenas um núcleo de processamento e com suporte exclusivo à aritmética inteira, o Simulador 1 se propõe a demonstrar que é possível, mesmo com recursos computacionais limitados, gerar pulsos com características estatísticas e temporais coerentes com o ambiente experimental do ATLAS.

Outro fator determinante para a escolha desse arranjo simplificado foi a viabilidade de síntese e simulação. Utilizando os arquivos de saída gerados pela IDE do SAPHO, foi possível descrever o sistema completo em *Verilog* e sintetizá-lo com ferramentas amplamente utilizadas em ambientes de FPGA, como o Quartus. Com isso, foi possível avaliar diretamente o comportamento da implementação em termos de latência, funcionalidade e compatibilidade com ferramentas de simulação digital, como o ModelSim.

Além de representar um passo inicial na construção do simulador final, esta versão permitiu estabelecer uma linha de base para comparação com as demais arquiteturas. Dessa forma, o Simulador 1 cumpre o papel fundamental de servir como referência para

avaliar ganhos de desempenho, eficiência e paralelismo à medida que o sistema é aprimorado nas versões seguintes.

5.1.1 Arquitetura Geral

A arquitetura desse simulador foi concebida de forma simples e didática, com o objetivo de permitir a validação completa dos blocos desenvolvidos e sua integração dentro do processador SAPHO. Esta primeira versão do sistema é composta por apenas um núcleo de processamento, cuja operação é exclusivamente com aritmética inteira, com palavra de dados de 32 bits, entre outras características, conforme a Tabela 5. Tal escolha foi orientada pela intenção de reduzir a complexidade da lógica gerada, simplificar o processo de síntese e facilitar a depuração dos resultados.

Tabela 5 – Características do Simulador 1

Parâmetro	Valor
Tipo de Representação Numérica	Ponto Fixo (0)
Número de Bits	32
Tamanho da Pilha de Dados	10
Tamanho da Pilha de Instruções	10
Número de Portas de Entrada	2
Número de Portas de Saída	2
Ganho Aplicado	256
Número Equações de Filtragem	5

O gerador de números pseudoaleatórios foi implementado diretamente dentro do processador, utilizando instruções personalizadas no código C e sem a necessidade de circuitos auxiliares externos. A cada ciclo, um novo valor pseudoaleatório uniforme é calculado, sendo este o ponto de partida para as etapas subsequentes de decisão (taxa de ocupação), mapeamento exponencial e modelagem temporal do pulso.

Todos os filtros digitais, descritos em 4.3 e utilizados para simular a forma temporal do pulso foram implementados diretamente como equações separadas, dentro desse mesmo processador. Dessa forma, o processador executa, de forma sequencial, os cálculos referentes a cada filtro, armazenando em registradores intermediários os valores parciais até a composição do sinal final.

Os coeficientes das equações dos filtros foram obtidos previamente através do MATLAB, e, por se tratar de valores fixos, puderam ser implementados diretamente no código do processador no SAPHO. Tais equações podem ser vistas abaixo:

$$y_1[n] = 0.1917 \cdot x[n] + 0.1917 \cdot x[n-1] + 0.3333 \cdot y_1[n-1] \quad (5.1)$$

$$y_2[n] = -0.1924 \cdot x[n] - 0.0973 \cdot x[n-1] + 0.0950 \cdot x[n-2] \\ + 0.1506 \cdot y_2[n-1] - 0.3326 \cdot y_2[n-2] \quad (5.2)$$

$$y_3[n] = -6.5254 \times 10^{-7} \cdot x[n] - 6.5254 \times 10^{-7} \cdot x[n-1] - 0.8644 \cdot y_3[n-1] \quad (5.3)$$

$$y_4[n] = 0.0233 \cdot x[n] - 0.0071 \cdot x[n-1] - 0.0305 \cdot x[n-2] \\ - 0.6230 \cdot y_4[n-1] - 0.6721 \cdot y_4[n-2] \quad (5.4)$$

$$y_5[n] = -4.3702 \times 10^{-4} \cdot x[n] - 4.3702 \times 10^{-4} \cdot x[n-1] + 0.9978 \cdot y_5[n-1] \quad (5.5)$$

$$y_{\text{total}}[n] = (y_1[n] + y_2[n] + y_3[n] + y_4[n] + y_5[n]) \cdot 1000 \quad (5.6)$$

1

É importante destacar que a equação final foi multiplicada por um fator de 1000 com o objetivo de realizar a **quantização da saída**. Isso se deve ao fato de o processador utilizado neste simulador operar com aritmética inteira. Dessa forma, a multiplicação permite representar de forma aproximada os valores reais dos pulsos com maior granularidade, mesmo utilizando apenas inteiros, facilitando a reconstrução da forma de onda no pós-processamento.

A estrutura geral do sistema é sintetizada em um fluxograma ilustrativo, apresentado na Figura 22. A sequência de operações realizadas pelo processador, desde a geração do número aleatório até a formação de uma amostra final, pode ser descrita da seguinte forma:

- **Início do Simulador:**

- Execução de um laço infinito `while(1)`, simulando continuamente amostras.

- **Geração de Número Aleatório:**

- Utiliza um gerador congruente linear: `rnd = rnd · a + c`.

- **Shift e Máscara:**

- Calcula o `rnd_out` com deslocamento e máscara: `rnd_out = (rnd >> bits) & 127`.

¹ Na implementação em C no SAPHO, os atrasos temporais foram representados por variáveis como `rx31`, `ry42`, etc., correspondentes a $x[n-1]$, $y[n-2]$, devido à limitação de não se poder usar diretamente índices temporais.

- **Verificação da Taxa de Ocupação:**

- Se `rnd_out < ccc`, um novo `rnd` é gerado e o valor de $x[n]$ é obtido da tabela:
 $x[n] = \text{mem_exp}[\text{rnd_out}]$.
- Caso contrário, $x[n] = 0$.

- **Filtro 1 (1ª Ordem):** A equação está apresentada na Equação 5.1.

Atualização das variáveis:

$$x_1[n-1] = x[n]; \quad y_1[n-1] = y_1[n]$$

- **Filtro 2 (2ª Ordem):** A equação está apresentada na Equação 5.2.

Atualização das variáveis:

$$x_2[n-2] = x_2[n-1]; \quad x_2[n-1] = x[n]; \quad y_2[n-2] = y_2[n-1]; \quad y_2[n-1] = y_2[n]$$

- **Filtro 3 (1ª Ordem):** A equação está apresentada na Equação 5.3.

Atualização das variáveis:

$$x_3[n-1] = x[n]; \quad y_3[n-1] = y_3[n]$$

- **Filtro 4 (2ª Ordem):** A equação está apresentada na Equação 5.4.

Atualização das variáveis:

$$x_4[n-2] = x_4[n-1]; \quad x_4[n-1] = x[n]; \quad y_4[n-2] = y_4[n-1]; \quad y_4[n-1] = y_4[n]$$

- **Filtro 5 (1ª Ordem):** A equação está apresentada na Equação 5.5.

Atualização das variáveis:

$$x_5[n-1] = x[n]; \quad y_5[n-1] = y_5[n]$$

- **Soma e Saída:** A equação final está apresentada na Equação 5.6.

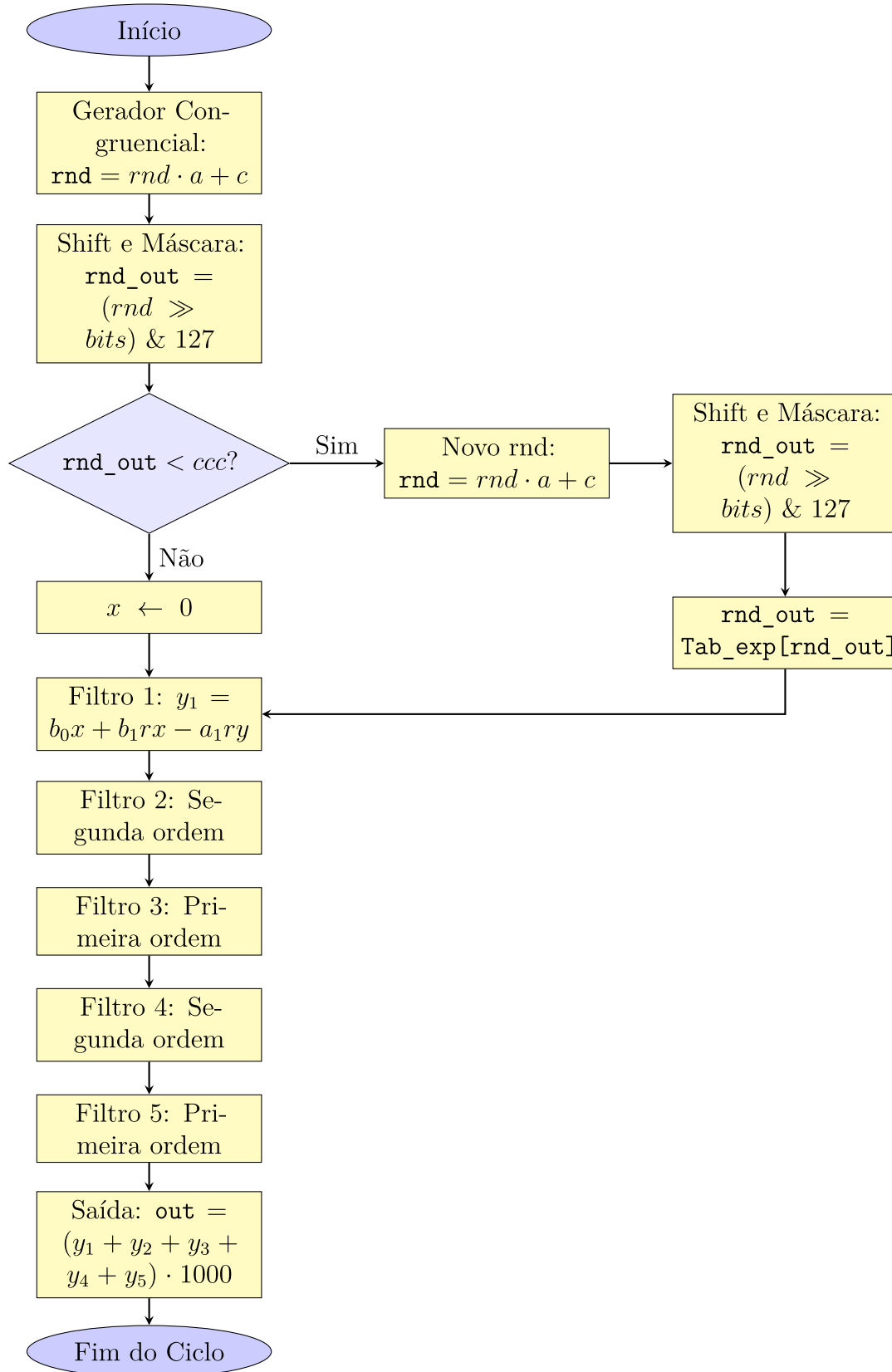
$$y_{\text{total}}[n] = (y_1[n] + y_2[n] + y_3[n] + y_4[n] + y_5[n]) \cdot 1000$$

- **Retorno ao Início do Laço:**

- Repete-se todo o processo para a próxima amostra.

Apesar da arquitetura do processador ser configurada como **inteira**, os coeficientes dos filtros digitais foram definidos como **float** no código-fonte. Isso implica na necessidade de conversão automática entre representações numéricas dentro do pipeline de execução.

Figura 22 – Fluxograma do Simulador 1: execução de um ciclo para a geração de uma amostra.



Fonte: Elaborado pelo autor (2025).

No SAPHO, essa operação é realizada por meio dos blocos **Float to Int** ($F \rightarrow I$) ou **Int to Float** ($I \rightarrow F$), inseridos automaticamente pelo compilador, conforme a configuração da ULA e a natureza das instruções. Esses blocos realizam a conversão entre valores com ponto flutuante e inteiros sempre que necessário, possibilitando a operação mista entre dados inteiros e coeficientes em ponto flutuante mesmo em arquiteturas com suporte restrito.

Tal característica reforça a flexibilidade do SAPHO para aplicações híbridas e permite que simuladores mais simples, como o apresentado neste caso, possam explorar recursos típicos de arquiteturas mais complexas, com sobrecarga mínima de hardware.

Esse modelo é funcional e coerente com os objetivos iniciais do projeto, ainda que apresente limitações em termos de desempenho, devido à ausência de paralelismo.

Essas representações visuais ajudam a contextualizar os aspectos técnicos da implementação e fornecem uma visão clara da composição estrutural do Simulador 1. Elas também servem como base de comparação com as versões posteriores, nas quais o grau de paralelismo, modularização e desempenho foram significativamente aprimorados.

5.2 SIMULADOR 2

O Simulador 2 representa um avanço conceitual em relação ao Simulador 1, adotando uma arquitetura mais compacta e integrada para o processamento dos sinais simulados, com foco em desempenho computacional e otimização da lógica de controle. Enquanto o simulador anterior executava as cinco equações de filtragem de forma isolada e sequencial, esta nova arquitetura utiliza uma única equação resultante — derivada da combinação matemática dos cinco filtros originais — para realizar a simulação completa da resposta do TileCal. A principal motivação por trás dessa modificação é a busca por redução no uso de recursos computacionais, simplificação da lógica de controle e otimização do desempenho do sistema embarcado, resultando em um número menor de clocks para a geração de cada amostra.

Do ponto de vista estrutural, o processador segue o mesmo padrão de arquitetura de ponto fixo configurado no SAPHO, com os mesmos parâmetros de largura de dados, profundidade das pilhas e portas de entrada e saída, características que podem ser visualizadas na Tabela 6. No entanto, ao unificar os filtros em uma única operação, foi possível simplificar o código C gerado, minimizar atualizações redundantes de registradores e compactar o ciclo de processamento por amostra.

Tabela 6 – Características do Simulador 2

Parâmetro	Valor
Tipo de Representação Numérica	Ponto Fixo (0)
Número de Bits	32
Tamanho da Pilha de Dados	10
Tamanho da Pilha de Instruções	10
Número de Portas de Entrada	2
Número de Portas de Saída	2
Ganho Aplicado	256
Número Equações de Filtragem	1

De acordo com a Seção 4.3, a resposta discreta do *shaper* do TileCal pode ser representada tanto pela soma de filtros parciais quanto por uma única equação de diferenças global. Para a implementação no simulador, adotou-se a forma combinada, expressa pela Eq. 5.7, cujos coeficientes foram obtidos a partir da discretização bilinear do modelo analógico (Equação 4.26). Essa equação define a relação direta entre as amostras de entrada $x[n]$ e as saídas passadas $y[n - k]$, servindo como base para a execução digital do algoritmo no processador SAPHO:

$$\begin{aligned}
y[n] = & 0,0222 \cdot x[n] + 0,1068 \cdot x[n-1] + 0,1894 \cdot x[n-2] + 0,1037 \cdot x[n-3] \\
& + 0,1276 \cdot y[n-2] + 0,4181 \cdot y[n-3] + 0,4352 \cdot y[n-4] + 0,1524 \cdot y[n-6] \\
& - \left(0,1063 \cdot x[n-4] + 0,1890 \cdot x[n-5] + 0,1053 \cdot x[n-6] + 0,0215 \cdot x[n-7] \right. \\
& \left. + 0,0056 \cdot y[n-1] + 0,0709 \cdot y[n-5] + 0,0643 \cdot y[n-7] \right)
\end{aligned} \tag{5.7}$$

Essa abordagem oferece vantagens em termos de eficiência, mas também impõe desafios: a equação resultante apresenta maior complexidade algébrica e maior número de termos, exigindo maior número de etapas de memória e somadores. Ainda assim, o modelo provou-se funcional e representa um passo importante na compactação do simulador, alinhando-se ao objetivo de desenvolver arquiteturas viáveis para implementação em FPGA sob restrições de área e consumo.

A seguir, é apresentada a descrição das etapas executadas pelo Simulador 2, com base na equação unificada e na estrutura geral do código implementado:

- **Início do Simulador:**

- Execução contínua via `while(1)`, garantindo a geração ininterrupta de amostras.

- **Geração do Número Aleatório:**

- Geração de `rnd` por meio do gerador congruente: $\text{rnd} = \text{rnd} \cdot a + c$.

- **Shift e Máscara:**

- Cálculo de `rnd_out` com operação de deslocamento e máscara: `rnd_out = (rnd \gg bits) & 127`.

- **Verificação da Taxa de Ocupação:**

- Se `rnd_out < ccc`, um novo número aleatório é gerado e o valor de entrada x é obtido da tabela exponencial.
- Caso contrário, $x \leftarrow 0$.

- **Aplicação da Equação Unificada:**

- A entrada $x[n]$ é processada segundo a equação 5.7, utilizando valores anteriores da entrada e da saída.

- **Atualização das Variáveis:**

- Os valores anteriores são atualizados conforme o deslocamento temporal:

$$\begin{aligned} x[n-7] &\leftarrow x[n-6], & x[n-6] &\leftarrow x[n-5], & \dots, & x[n-1] &\leftarrow x[n] \\ y[n-7] &\leftarrow y[n-6], & y[n-6] &\leftarrow y[n-5], & \dots, & y[n-1] &\leftarrow y[n] \end{aligned}$$

- **Saída do Processador:**

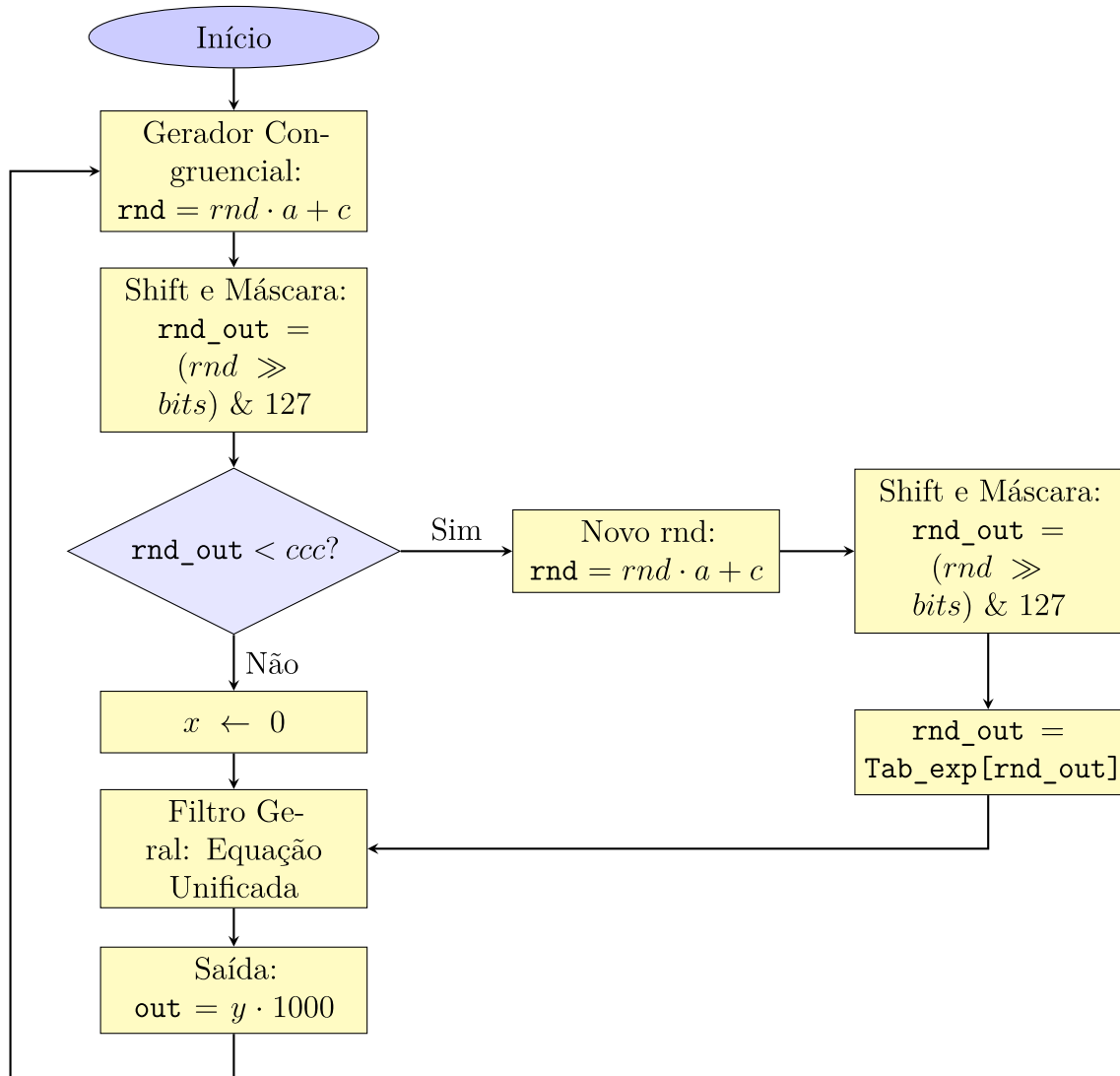
- A saída final é obtida por: $y[n] \cdot 1000$, garantindo compatibilidade com representação inteira.

O processo de simulação descrito anteriormente é executado de forma contínua, como ilustrado na Figura 23, garantindo que uma nova amostra seja gerada a cada ciclo do processador. Isso reflete o comportamento desejado em sistemas embarcados de tempo real, nos quais a aquisição, o processamento e a saída dos dados ocorrem de forma ininterrupta. O laço `while(1)`, implícito na estrutura do código, assegura essa execução cíclica, característica essencial para aplicações de alta taxa de eventos como as enfrentadas no experimento ATLAS.

Embora a implementação por meio de uma única equação de filtragem reduza a quantidade de operações de controle e atualizações de registradores intermediários, essa estratégia exige mais do compilador C e do otimizador do SAPHO. O número de operandos envolvidos, somado à profundidade da memória necessária para armazenar os valores anteriores das entradas e saídas, aumenta a complexidade dos caminhos de dados internos e a ocupação lógica do FPGA. Ainda assim, o ganho obtido em eficiência de tempo, com a redução significativa no número de ciclos por amostra, compensa esse aumento moderado de recursos.

Um ponto importante é que, mesmo mantendo a arquitetura em ponto fixo, os coeficientes da equação de filtragem são representados em ponto flutuante. Isso exige

Figura 23 – Fluxograma do Simulador 2 com ciclo contínuo de execução.



Fonte: Elaborado pelo autor (2025).

que o compilador SAPHO gere automaticamente os blocos de conversão entre formatos — por exemplo, de inteiro para ponto flutuante na entrada, e de ponto flutuante para inteiro na saída — conforme o tipo de dado atribuído a cada variável. Tais conversões são realizadas por blocos dedicados, que aumentam o consumo de recursos, mas garantem compatibilidade e precisão na operação, permitindo que o desenvolvedor aproveite a simplicidade da arquitetura inteira sem perder a funcionalidade dos coeficientes em ponto flutuante.

A multiplicação da saída por um fator de ganho (neste caso, 1000) compensa a limitação de precisão imposta pela quantização da saída do processador, e permite o uso de inteiros como representação final da amplitude simulada. Essa técnica é comum em aplicações de filtragem digital embarcada e permite evitar a propagação de erros de arredondamento, preservando a fidelidade da forma de onda simulada.

Dessa forma, o Simulador 2 representa uma alternativa arquitetural ao Simulador 1, concebida com o objetivo de reduzir o número de ciclos de clock necessários para a geração de cada amostra. Ao unificar as equações de filtragem em uma única expressão matemática, busca-se otimizar o fluxo de execução, simplificando a lógica de controle e diminuindo o número de operações envolvidas no processamento de sinais.

Tal proposta é particularmente relevante em sistemas embarcados com restrições de área e tempo, como os que visam operação em tempo real em ambientes de alta densidade de eventos. Embora os resultados comparativos sejam discutidos posteriormente, a implementação do Simulador 2 já demonstra o potencial de reorganizar a estrutura de simulação para viabilizar alternativas mais compactas e eficientes.

Além disso, essa arquitetura ressalta a versatilidade do ambiente de desenvolvimento SAPHO, que permite explorar diferentes estratégias de simulação baseadas em um mesmo modelo conceitual, promovendo maior liberdade de experimentação e adaptação às exigências do sistema final.

5.3 SIMULADOR 3

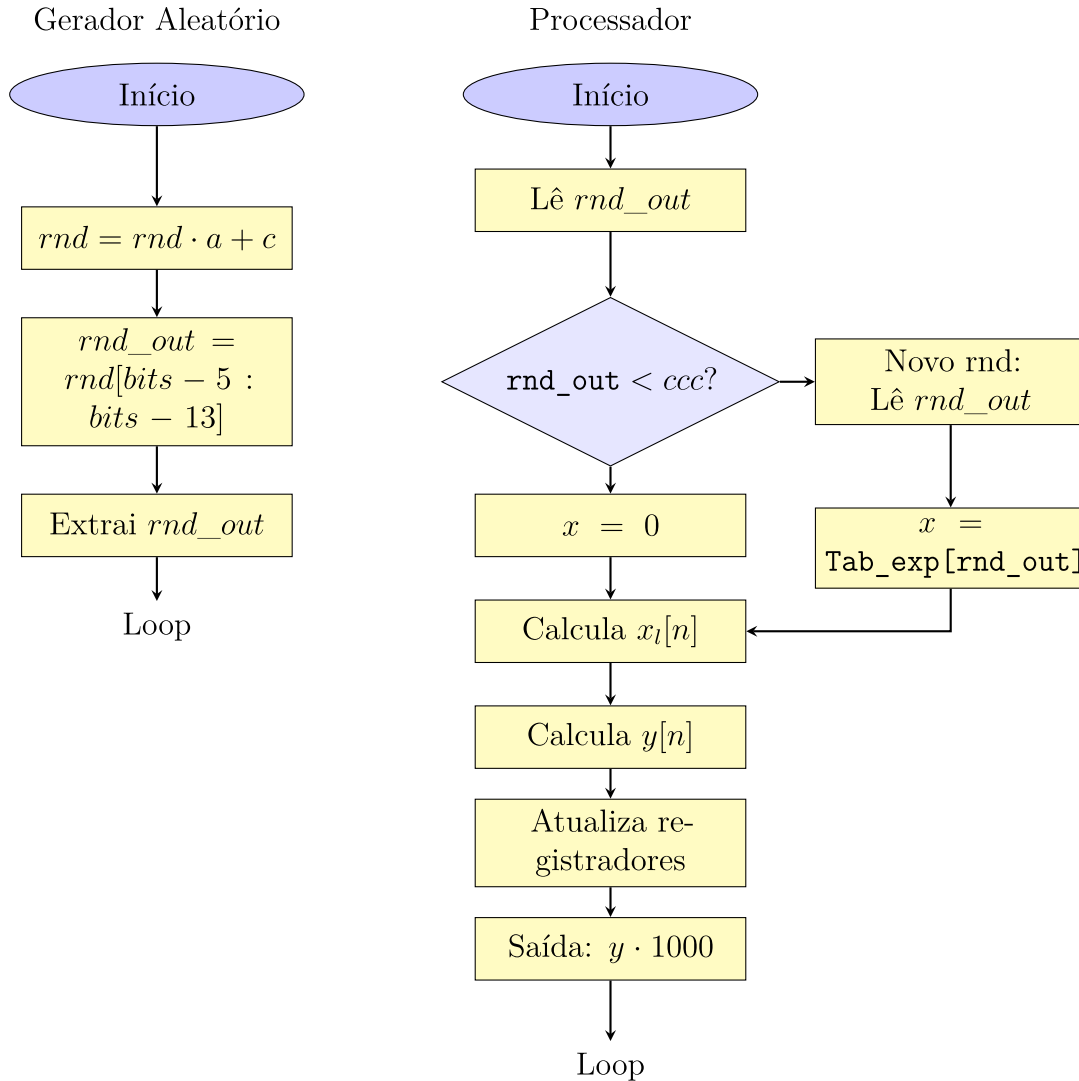
O Simulador 3 representa uma alternativa arquitetural fundamentada na utilização de processadores em ponto flutuante. Essa escolha foi motivada pela possibilidade de representar uma faixa dinâmica de valores mais ampla, eliminando a necessidade de etapas explícitas de normalização ou ganho como nos simuladores anteriores. Ao contrário dos simuladores 1 e 2, que utilizam representação em ponto fixo e exigem multiplicações por fatores de escala como 256 ou 1000, o Simulador 3 opera diretamente com números reais, o que facilita o mapeamento dos pulsos a valores energéticos mais realistas e reduz as perdas associadas à quantização e ao arredondamento. Além disso, a precisão mantida na representação dos coeficientes e dos resultados permite um maior controle sobre o perfil temporal do pulso simulado, tornando-o ideal para testes com algoritmos sensíveis a pequenas variações.

Entretanto, o uso do ponto flutuante também impõe restrições funcionais no SAPHO. De acordo com a Tabela 1, a ULA em ponto flutuante não oferece suporte a algumas instruções essenciais disponíveis na versão ponto fixo, como operações de *shift* (deslocamento de bits), inversão lógica, ou máscara de bits. Essa limitação impede que certos blocos de manipulação binária — utilizados, por exemplo, na geração de números pseudoaleatórios com compressão de bits — sejam implementados diretamente no processador. Em função disso, o gerador congruencial linear foi implementado como um módulo externo, escrito diretamente em Verilog e instanciado no nível superior do sistema, fora do SAPHO. Por outro lado, a verificação da taxa de ocupação e a aplicação da tabela exponencial permaneceram dentro do processador, já que envolvem operações aritméticas compatíveis com a arquitetura de ponto flutuante.

A integração entre os blocos externos em Verilog e o processador gerado pelo SAPHO foi feita por meio de um arquivo `top_level`, denominado `pulse_sim.v`. Nesse módulo de integração, a saída do gerador externo (`rand_out`) é concatenada com bits fixos de controle e conectada à porta de entrada do processador (`proc_sim.v`). Essa conexão permite que os valores aleatórios gerados fora do processador sejam utilizados internamente como índices de acesso à tabela exponencial carregada na memória de dados (`Proc_Sim_data.mif`), dentro do SAPHO. Esse fluxo garante a continuidade da lógica de simulação e torna possível a reutilização do mesmo código C da camada interna sem alterações profundas. O código em Verilog que realiza essa interconexão também é responsável por mapear as saídas do processador e decodificá-las em sinais utilizáveis no ambiente de simulação e controle.

A Figura 24 apresenta o fluxograma detalhado do Simulador 3, evidenciando a separação funcional entre o gerador de números pseudoaleatórios — implementado em Verilog e executado continuamente no módulo superior — e o processador SAPHO, que realiza o cálculo da filtragem apenas quando um novo dado de entrada é requerido. Diferentemente do Simulador 2, onde todos os blocos pertencem a uma cadeia sequencial única, o Simulador 3 adota uma arquitetura desacoplada e reativa: o processador somente consome os valores gerados quando há evento, permitindo maior paralelismo e eficiência no uso de recursos.

Figura 24 – Fluxograma do Simulador 3, detalhando os diferentes loops paralelos entre processador e gerador.



Fonte: Elaborado pelo autor (2025).

Com essa configuração híbrida — parte em HDL manual e parte gerada automaticamente — o Simulador 3 consegue contornar as limitações funcionais da arquitetura flutuante do SAPHO, mantendo desempenho adequado e compatibilidade com os modelos matemáticos anteriormente definidos. Essa abordagem modular reforça a flexibilidade da plataforma, permitindo testar diferentes arquiteturas de maneira incremental e adaptável conforme as restrições do hardware-alvo.

As principais características do Simulador 3 estão resumidas na Tabela 7, sendo possível observar diferenças importantes em relação às versões anteriores, tanto na arquitetura do processador quanto na estratégia de integração dos blocos lógicos.

Tabela 7 – Características do Simulador 3

Parâmetro	Valor
Tipo de Representação Numérica	Ponto Flutuante (1)
Bits da Mantissa	16
Bits do Expoente	6
Número de Bits Totais	23
Tamanho da Pilha de Dados	10
Tamanho da Pilha de Instruções	10
Número de Portas de Entrada	1
Número de Portas de Saída	2
Ganho Aplicado	128
Número de Equações de Filtragem	1
Gerador de Números Aleatórios	Módulo externo em Verilog
Verificação de Ocupação	Interna ao processador
Tabela Exponencial	Interna ao processador

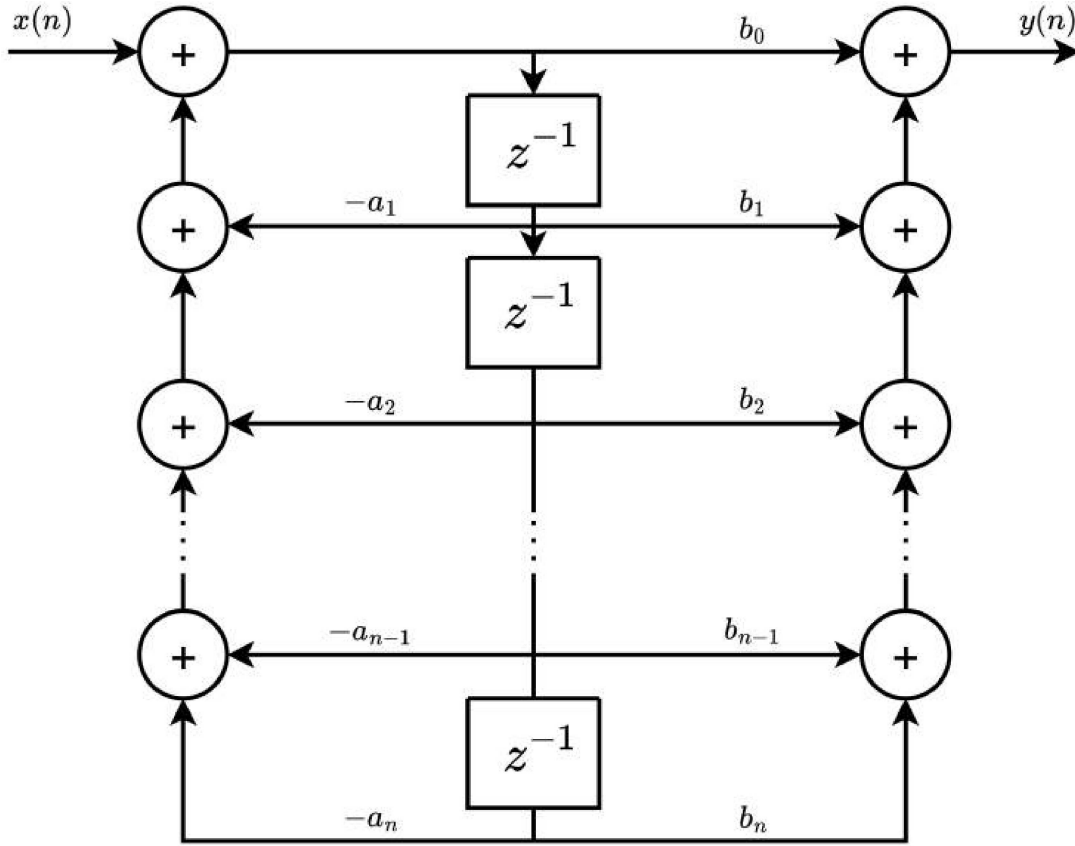
A modelagem matemática do Simulador 3 foi baseada na equação de diferenças resultante da discretização da função de transferência global $H(s)$, obtida a partir da soma das cinco funções de transferência analógicas apresentadas na Equação 4.12. Essa soma foi realizada no domínio contínuo e posteriormente discretizada utilizando a transformada bilinear, conforme detalhado na Equação 5.7. O objetivo foi unificar os cinco filtros do sistema original em uma única equação de filtragem IIR de ordem superior, reduzindo a complexidade estrutural do código, ainda que mantendo a mesma resposta temporal global.

No entanto, a estrutura utilizada para implementar essa equação é diferente, denominada **Forma Direta II**, uma arquitetura clássica para implementação de filtros IIR. Nessa configuração, os mesmos registradores de estado são compartilhados para aplicar os coeficientes do numerador e do denominador, permitindo maior eficiência em termos de uso de memória e simplicidade no fluxo de dados.

A Forma Direta II permite a realização de filtros utilizando um número mínimo de registradores de atraso, ao reutilizar os mesmos estados para o numerador e o denominador da função de transferência. No caso específico do Simulador 3, mesmo tratando-se de um filtro do tipo FIR (*Finite Impulse Response*), a estrutura adotada se beneficia dessa topologia para reduzir a complexidade de hardware.

Na Figura 25, apresenta-se a estrutura conceitual da Forma Direta II, conforme referência clássica da literatura.

Figura 25 – Estrutura da Forma Direta II para filtros digitais



Fonte: Elaborado pelo autor (2025)

A modelagem matemática do Simulador 3 é baseada na combinação de duas equações principais: uma equação de pré-processamento (estado intermediário) e uma equação de saída. Ambas utilizam registradores de atraso r_k que armazenam os estados internos do filtro.

A primeira etapa realiza um ajuste do sinal de entrada $x[n]$, incorporando efeitos de estados passados, conforme a equação:

$$x_1[n] = x[n] + 0,1267 r_2 + 0,4181 r_3 + 0,4352 r_4 + 0,1524 r_6 - (0,0056 r_1 + 0,0709 r_5 + 0,0643 r_7) \quad (5.8)$$

A saída do sistema é uma combinação linear do estado atual e de seus estados passados:

$$y[n] = 0,0222 x_1[n] + 0,1068 r_1 + 0,1894 r_2 + 0,1037 r_3 - (0,1063 r_4 + 0,1890 r_5 + 0,1053 r_6 + 0,0215 r_7) \quad (5.9)$$

Após o cálculo de $y[n]$, os registradores de estado são atualizados em cascata, conforme segue:

$$\begin{aligned}
r_7 &\leftarrow r_6 \\
r_6 &\leftarrow r_5 \\
r_5 &\leftarrow r_4 \\
r_4 &\leftarrow r_3 \\
r_3 &\leftarrow r_2 \\
r_2 &\leftarrow r_1 \\
r_1 &\leftarrow x_l[n]
\end{aligned} \tag{5.10}$$

Apesar de não haver realimentação direta da saída $y[n]$, a manipulação interna dos estados na equação (5.8) simula um comportamento parcialmente recursivo. No entanto, como a saída depende exclusivamente de valores passados do estado intermediário, o filtro permanece classificado como FIR. A forma direta II, nesse contexto, reduz a quantidade de registradores de atraso necessários, o que é essencial para implementação eficiente em FPGA.

A escolha dessa estrutura para o simulador 3 permite:

- Redução no número de blocos de memória utilizados (menos registradores);
- Maior velocidade de processamento devido à simplificação estrutural;
- Facilidade de *pipeline* para execução paralela;
- Economia de recursos lógicos na FPGA, permitindo uso de LUTs e blocos DSP com maior eficiência.

Além disso, essa abordagem é especialmente útil quando a reconstrução de energia deve ocorrer em tempo real, como no contexto de altas taxas de eventos do experimento ATLAS, no LHC. O simulador é capaz de realizar a filtragem com baixa latência, reproduzindo fielmente a forma de pulso do TileCal, mesmo em ambientes com ruído e sobreposição de eventos.

A modelagem matemática apresentada para o Simulador 3 demonstra a adequação da Forma Direta II como estrutura eficiente para filtros digitais implementados em FPGA. Ao combinar desempenho, economia de recursos e fidelidade ao sinal físico, esta abordagem representa uma solução robusta para aplicações em instrumentação de física de altas energias.

5.4 SIMULADOR 4

O Simulador 4 representa a versão final e mais refinada da linha de simuladores desenvolvidos ao longo deste trabalho, incorporando as lições aprendidas nas versões

anteriores e visando maximizar o desempenho, a modularidade e a compatibilidade com ambientes de alta taxa de eventos. Esta versão foi concebida com o objetivo de atender aos requisitos funcionais e arquiteturais observados nas aplicações do TileCal.

Anteriormente, diferentes estratégias foram exploradas: o Simulador 1 testou uma abordagem sequencial baseada em ponto fixo, o Simulador 2 introduziu a unificação da equação de filtragem e lógica de controle, enquanto o Simulador 3 adotou ponto flutuante e separação funcional entre os blocos de geração e processamento. No entanto, cada uma dessas versões ainda apresentava limitações específicas, como o acoplamento excessivo entre os blocos internos, a ausência de paralelismo entre etapas de cálculo e a utilização de estruturas de controle parcialmente centralizadas.

A principal diferença do Simulador 4 em relação ao Simulador 3 está na estratégia adotada para a filtragem. Em vez de consolidar todas as respostas temporais em uma única equação de filtragem global, a proposta será a abordagem de múltiplas equações individuais, semelhantes às utilizadas no Simulador 1. No entanto, ao contrário das versões anteriores, essas equações não são executadas sequencialmente dentro de um único processador: cada equação de filtragem é compilada em um código C separado e instanciada como um processador SAPHO independente. Com isso, o sistema final consiste em uma arquitetura paralela composta por múltiplos processadores operando simultaneamente, cada um responsável por processar uma das equações de filtragem. As saídas desses processadores são então somadas no nível superior do sistema, no arquivo `top level` escrito em Verilog, formando a saída final do simulador. Essa abordagem distribui a carga computacional entre vários núcleos e possibilita uma execução altamente paralela, reduzindo a latência total e aumentando a taxa de processamento sem comprometer a fidelidade da simulação.

Uma motivação central para esta versão foi alcançar uma execução próxima da taxa real de eventos do TileCal, permitindo simulações realistas em 40 MHz com fidelidade temporal preservada. Para isso, a lógica foi reescrita com ênfase em paralelismo de instruções, redução do número de ciclos por operação e minimização do número de instruções redundantes na pilha de cada processador. O processador SAPHO foi explorado ao máximo, aproveitando suas instruções aritméticas em ponto flutuante e otimizando a organização do código em C para se adequar à arquitetura da ULA.

Como resultado, o Simulador 4 consolida a proposta iniciada nas versões anteriores, incorporando o mesmo modelo matemático de resposta de pulso, porém com uma implementação muito mais eficiente. A estrutura modular e paralela, combinada com a integração transparente entre os blocos em Verilog e os arquivos de dados dos processadores, tornam essa versão apta a ser utilizada em testes reais de aquisição e simulação de pulsos do TileCal, reproduzindo com precisão as formas de sinal que chegam aos canais de leitura, com desempenho compatível com os requisitos de operação do ATLAS.

A versão final do simulador foi concebida com uma arquitetura distribuída, na qual múltiplos processadores SAPHO operam de forma paralela, cada um dedicado à execução de uma equação de filtragem individual. Esse arranjo modular permite a divisão da carga computacional entre diferentes núcleos, reduzindo a latência geral do sistema e aumentando sua escalabilidade. Além disso, a independência entre os blocos favorece a reutilização de código e simplifica o processo de depuração e validação funcional.

Cada processador é configurado com uma tabela exponencial interna e lógica de verificação de ocupação local, preservando a autonomia das instâncias. As saídas resultantes das cinco equações implementadas são somadas externamente por um módulo adicional em Verilog, garantindo a recomposição correta da forma de pulso. A Tabela 8 apresenta os principais parâmetros adotados nesta implementação.

Tabela 8 – Características da Implementação Final com Múltiplos Processadores SAPHO

Parâmetro	Valor
Tipo de Representação Numérica	Ponto Flutuante (1)
Bits da Mantissa	16
Bits do Expoente	6
Número de Bits Totais	23
Número de Processadores SAPHO	5
Número de Equações de Filtragem	5 (uma por processador)
Ganho Aplicado por Canal	128
Tamanho da Pilha de Dados	10
Tamanho da Pilha de Instruções	10
Número de Portas de Entrada por Núcleo	1
Número de Portas de Saída por Núcleo	2
Gerador de Números Aleatórios	Módulo externo em Verilog
Tabela Exponencial	Interna ao Gerador de Números Aleatórios
Verificação de Ocupação	Interna ao Gerador de Números Aleatórios
Somador Final	Módulo externo em Verilog

5.4.1 Equações de Filtragem

Nesta versão final, a filtragem do sinal é realizada por meio de cinco núcleos SAPHO operando em paralelo, cada um responsável por aplicar uma equação distinta associada a uma componente específica da resposta ao pulso. Diferentemente da abordagem unificada adotada anteriormente, a presente configuração distribui o processamento entre diferentes instâncias, permitindo a execução simultânea de equações com diferentes ordens e características.

As expressões utilizadas foram extraídas do modelo analógico original, e discretizadas com base nas características de cada componente. Nos casos dos filtros de primeira ordem — implementados nos processadores 1, 3 e 5 —, optou-se por uma estrutura simples e direta, uma vez que o número de operações por ciclo é reduzido. Já para os filtros de

segunda ordem (processadores 2 e 4), foi empregada a Forma Direta II, como no Simulador 3, visando minimizar o número de ciclos por saída e otimizar o uso dos recursos internos do SAPHO.

As equações discretas implementadas por cada núcleo são apresentadas a seguir.

Processador 1 (Filtro de 1ª ordem):

$$y_1[n] = 0,1917 \cdot x[n] + 0,1917 \cdot r_x + 0,3333 \cdot r_y \quad (5.11)$$

$$r_x \leftarrow x[n], \quad r_y \leftarrow y_1[n]$$

Processador 2 (Filtro de 2ª ordem — Forma Direta II):

$$x_1[n] = x[n] + 0,1506 \cdot r_1 - 0,3326 \cdot r_2 \quad (5.12)$$

$$y_2[n] = 0,9500 \cdot r_2 - (0,1924 \cdot x_1[n] + 0,0973 \cdot r_1) \quad (5.13)$$

$$r_2 \leftarrow r_1, \quad r_1 \leftarrow x_1[n]$$

Processador 3 (Filtro de 1ª ordem — resposta lenta):

$$y_3[n] = - \left(6,5254 \times 10^{-7} \cdot x[n] + 6,5254 \times 10^{-7} \cdot r_x + 0,8644 \cdot r_y \right) \quad (5.14)$$

$$r_x \leftarrow x[n], \quad r_y \leftarrow y_3[n]$$

Processador 4 (Filtro de 2ª ordem — Forma Direta II):

$$x_1[n] = x[n] - (0,6230 \cdot r_1 + 0,6721 \cdot r_2) \quad (5.15)$$

$$y_4[n] = 0,0233 \cdot x_1[n] - (0,0071 \cdot r_1 + 0,0305) \quad (5.16)$$

$$r_2 \leftarrow r_1, \quad r_1 \leftarrow x_1[n]$$

Processador 5 (Filtro de 1ª ordem):

$$y_1[n] = -0,000437 \cdot (x[n] + r_x) + 0,9978 \cdot r_y \quad (5.17)$$

$$r_x \leftarrow x[n], \quad r_y \leftarrow y_1[n]$$

A saída total do sistema é obtida a partir da soma das saídas individuais de cada processador, realizada no nível superior do projeto em Verilog:

$$y[n] = y_1[n] + y_2[n] + y_3[n] + y_4[n] + y_5[n] \quad (5.18)$$

Para garantir a sincronização entre os núcleos, todos os processadores são acionados simultaneamente e compartilham a mesma entrada pseudoaleatória em cada ciclo de

simulação. Como alguns filtros exigem menos instruções para serem executados (em especial os de primeira ordem), foi necessário incluir instruções artificiais de espera, como somas nulas do tipo $x = x + 0$, com o objetivo de igualar o número de ciclos entre os diferentes núcleos. Essa técnica assegura que todas as saídas estejam disponíveis no mesmo instante e evita desalinhamento no somador final. Tal abordagem é particularmente útil quando se deseja garantir comportamento determinístico na simulação em tempo real, facilitando a validação do modelo.

5.4.2 Arquitetura Geral

A estrutura adotada na implementação final segue uma abordagem paralela e segmentada, organizada em três níveis principais: um módulo gerador externo de números pseudoaleatórios, cinco processadores SAPHO operando em paralelo, e um bloco somador que agrega as saídas individuais para produzir o sinal final.

O componente responsável pela geração dos eventos é o módulo `generate_random_32`, desenvolvido diretamente em Verilog. Ele realiza não apenas a geração de números congruenciais, mas também aplica internamente a verificação da taxa de ocupação e o mapeamento exponencial — ambos incorporados ao próprio código do gerador. Dessa forma, a saída do bloco externo já corresponde ao valor final de entrada que será fornecido aos núcleos SAPHO, dispensando qualquer processamento adicional dentro dos processadores.

Cada processador SAPHO executa uma equação distinta de filtragem, conforme descrito na Seção 5.4.1, e opera de maneira independente, porém sincronizada. Essa sincronização é garantida pelo fato de que todos os núcleos solicitam uma nova entrada exatamente no mesmo ciclo de clock, por meio do comando `in(0)`. Como o valor de entrada (`rand_out`) está em constante atualização no nível superior, todos os processadores recebem o mesmo dado simultaneamente, mantendo coerência no alinhamento temporal das saídas.

O sinal `rand_out`, com largura de 9 bits, é conectado às entradas dos cinco núcleos através de uma concatenação de bits fixos no topo do módulo `pulse_sim (top_level)`. Cada processador é instanciado com seu respectivo código em C (convertido em Verilog pelo SAPHO), representando uma equação distinta do modelo analógico discretizado.

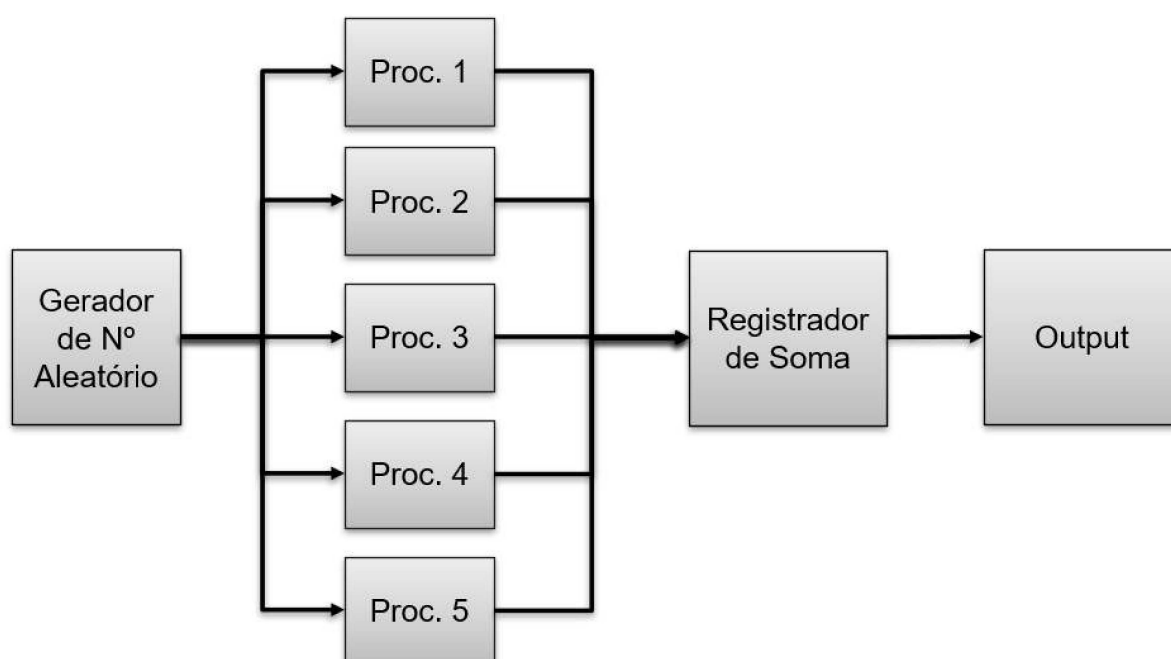
As saídas parciais dos processadores (`out1` a `out5`) são monitoradas individualmente, mas são combinadas dentro de um bloco somador central, também em Verilog. Essa agregação é feita de forma síncrona no `pulse_sim`, por meio de um `always @ (posedge clk)`, utilizando o sinal de habilitação `out_en1` como referência para efetuar a soma:

```
always @ (posedge clk)
    if (out_en1) out <= out1 + out2 + out3 + out4 + out5;
```

O sinal de habilitação é uma forma de verificar, através da simulação, se os processadores estão sincronizados ou não. Isso se faz necessário pois saída final `out` só deve ser modificada quando for expelida uma saída pelo processador, ou seja, após o processador terminar as equações e atualização de variáveis. Assim, para a estrutura de soma, pode-se utilizar qualquer um dos 5 sinais de habilitação dos processadores, pelo fato de haver sincronização entre eles.

A Figura 26 ilustra a arquitetura geral do sistema para geração de uma amostra, destacando a conexão entre os blocos externos, os núcleos SAPHO e o somador final.

Figura 26 – Arquitetura geral do Simulador 4.

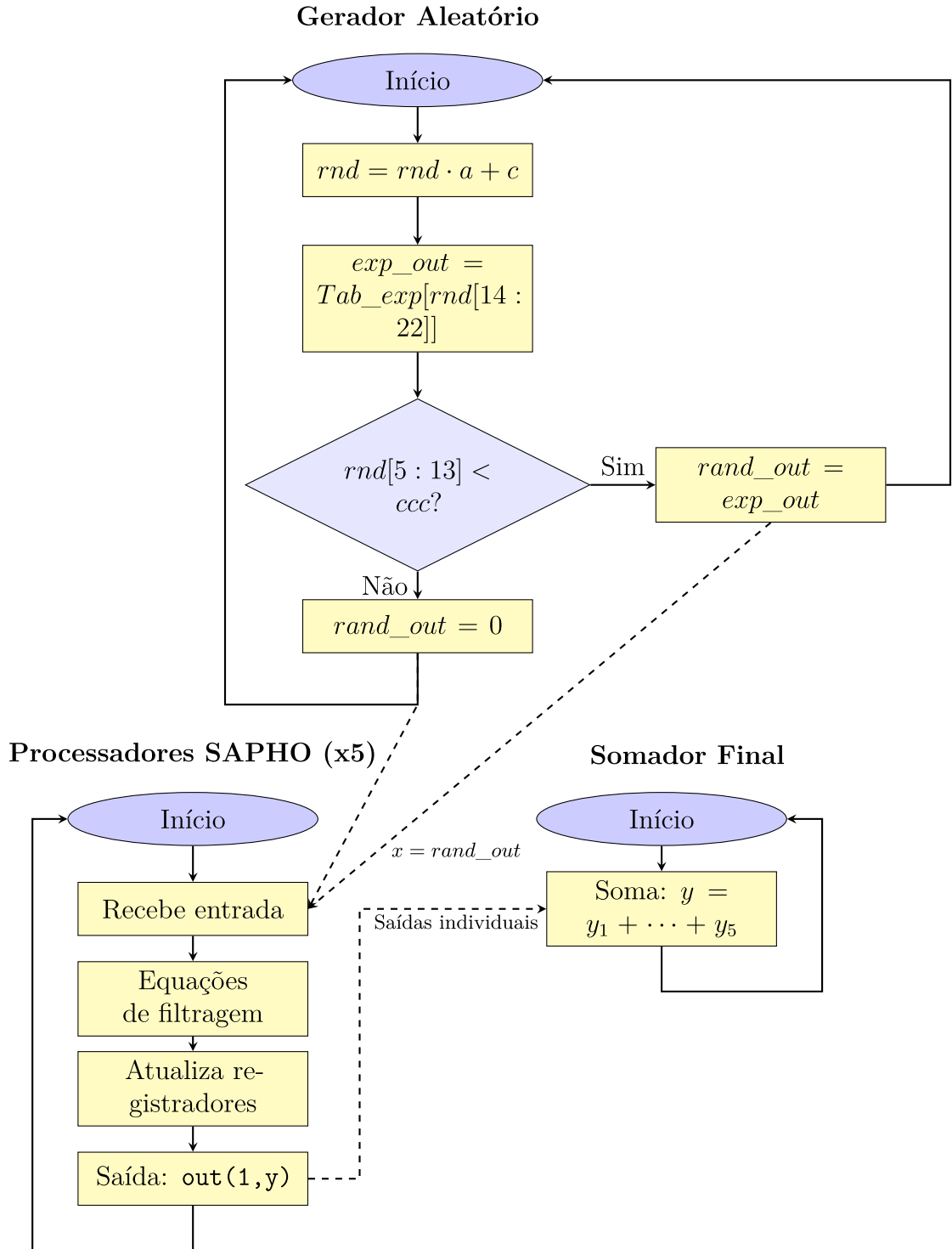


Fonte: Elaborado pelo autor (2025).

Devido à natureza paralela e distribuída do Simulador 4, torna-se fundamental representar graficamente a interação entre os diferentes blocos funcionais que compõem a arquitetura. A Figura 27 apresenta o fluxograma de execução dessa versão, dividida em três componentes principais: o gerador pseudoaleatório, os cinco processadores SAPHO e o somador final em Verilog.

Na parte superior do fluxograma, encontra-se o bloco responsável pela geração dos valores de entrada. Esse gerador implementa um algoritmo congruencial linear para produzir números pseudoaleatórios e aplica diretamente duas etapas: a verificação da taxa de ocupação e a transformação pela tabela exponencial. A variável `rand`, que possui 32 bits, é utilizada tanto para o mapeamento de seu respectivo valor na tabela exponencial (`exp_out = Tab_exp[rand[14:22]]`), quanto para verificação da taxa de ocupação (`rand[5:13] < ccc?`). Apesar de serem utilizados 9 bits em ambas as etapas, são bits distintos, para

Figura 27 – Fluxograma de execução do Simulador 4, com divisão entre gerador pseudoaleatório, múltiplos processadores SAPHO e somador final.



Fonte: Elaborado pelo autor (2025).

que não sejam limitados valores de saída em posição na tabela exponencial superiores a ccc , por conta da condição de ocupação. O valor final, chamado **rand_out**, é atualizado continuamente a cada ciclo de clock. Quando a condição de ocupação é satisfeita, o valor gerado é atualizado para **exp_out**; caso contrário, é zerado.

Na porção inferior esquerda do diagrama, os cinco processadores são representados de forma unificada, já que compartilham o mesmo comportamento estrutural: recebem como entrada o valor `rand_out`, aplicam a equação de filtragem correspondente, atualizam seus registradores internos e produzem uma saída parcial y_i . Essa execução ocorre de forma síncrona, com todos os processadores acessando a entrada no mesmo ciclo, o que garante que as saídas estejam alinhadas no tempo.

À direita do bloco de processamento, encontra-se o somador final, responsável por agregar as cinco saídas individuais em uma única amostra de saída $y[n]$. Essa soma é realizada quando o primeiro processador habilita seu sinal de saída (`out_en1`), o que indica que todos os núcleos já finalizaram seu processamento para aquele ciclo. O valor combinado é, então, armazenado para posterior análise e validação.

Essa representação da Figura 27 ilustra claramente a independência entre os blocos, ao mesmo tempo em que destaca os pontos de comunicação e sincronização necessários para garantir a consistência do processamento. O modelo paralelo adotado nesta versão permite simular em tempo real pulsos com fidelidade e desempenho compatíveis com aplicações de alta taxa de eventos, como as encontradas no TileCal.

A quarta e última versão do simulador consolida as melhorias progressivas desenvolvidas ao longo deste trabalho, ao incorporar paralelismo real entre núcleos, utilizar de forma otimizada os recursos do SAPHO e adotar uma estrutura modular compatível com execução em tempo real. A equalização do número de ciclos entre os processadores garante o alinhamento das saídas, mesmo diante de equações de diferentes ordens, enquanto a separação entre gerador, filtros e somador promove escalabilidade e organização do projeto.

6 RESULTADOS

Este capítulo apresenta uma análise detalhada dos quatro simuladores desenvolvidos ao longo deste trabalho, com foco na avaliação do desempenho, eficiência e adequação para ambientes com alta taxa de eventos, como o TileCal no experimento ATLAS. O principal objetivo é comparar, de forma quantitativa, o custo computacional de cada versão e sua viabilidade para operação em tempo real, considerando o desafio de simular pulsos compatíveis com uma frequência de 40 MHz, correspondente a um intervalo de apenas 25 ns entre eventos no LHC.

Para isso, um dos principais indicadores utilizados é a quantidade de ciclos de clock necessários para gerar uma única amostra em cada versão do simulador. Com esse valor, calcula-se a frequência mínima necessária para que o sistema consiga operar em tempo real, utilizando a seguinte equação:

$$f_{\text{necessária}} = N_{\text{clocks}} \times 40 \text{ MHz} \quad (6.1)$$

Esse cálculo indica a frequência de operação exigida para que um simulador específico consiga gerar amostras em tempo compatível com a taxa de eventos do LHC. Assim, quanto menor o número de ciclos por amostra, menor será a frequência exigida da FPGA, aumentando a viabilidade da implementação física.

De modo análogo, adota-se como referência uma frequência típica de operação de 500 MHz para dispositivos FPGA modernos. A partir desse valor, é possível estimar a frequência efetiva de simulação de cada versão do sistema, considerando o número de ciclos de clock consumidos por amostra. Esse cálculo fornece uma medida prática de até que ponto o simulador pode ser utilizado em experimentos de validação de metodologias e testes de reconstrução em tempo real. A equação correspondente é:

$$f_{\text{sim}} = \frac{500 \text{ MHz}}{N_{\text{clocks}}} \quad (6.2)$$

A Tabela 9 resume as principais características de cada simulador, servindo como referência para as análises que serão desenvolvidas nas próximas seções.

Tabela 9 – Resumo comparativo entre os simuladores desenvolvidos

Simulador	Tipo de Proc.	Nº de Equações	Paralelismo	Ger. Aleatória
1	Inteiro	5 eq. Separadas	Não	Interna
2	Inteiro	Combinadas em 1	Não	Interna
3	Ponto flutuante	Combinadas em 1	Não	Externa
4	Ponto flutuante	5 (1 por proc.)	Sim (5 proc.)	Externa

Nos tópicos seguintes, cada simulador será analisado individualmente, com gráficos e medições obtidas via simulação no ambiente ModelSim e MATLAB. As comparações

finais destacarão a evolução obtida ao longo das versões, culminando com a proposta final de um simulador em tempo real apto a operar em condições próximas às do TileCal.

6.1 SIMULADOR 1

O Simulador 1 foi o primeiro modelo funcional implementado neste trabalho e serviu como base inicial para os experimentos posteriores. Sua estrutura foi desenvolvida com foco em simplicidade, utilizando ponto fixo como forma de representação numérica e incorporando toda a lógica de controle e geração de dados diretamente no código C compilado via SAPHO.

Neste modelo, tanto o gerador de números pseudoaleatórios quanto a verificação de ocupação e a aplicação da tabela exponencial foram implementados internamente ao próprio processador. Isso implica que a geração de cada valor de entrada envolvia múltiplas instruções aritméticas e condicionais, executadas em tempo de simulação, o que afeta diretamente o número de ciclos consumidos por amostra.

Além disso, o modelo utilizava coeficientes de filtragem com valores inferiores a 1, os quais exigiam operações do tipo divisão e multiplicação por constantes fracionárias. Como a ULA do SAPHO não realiza operações diretamente em ponto flutuante nesta configuração, essas instruções implicam conversões entre inteiro e ponto flutuante em tempo de execução. Tais conversões impõem um custo adicional não desprezível, aumentando a latência do simulador.

Outro fator que impacta na variação do número de ciclos por amostra é a instrução condicional do tipo `if`, utilizada para verificar se o número gerado satisfaz o critério de ocupação. Caso a condição seja atendida, o código executa blocos adicionais para realizar o mapeamento exponencial; caso contrário, zera a entrada. Como consequência, o tempo total para gerar cada amostra não é constante e depende da quantidade de vezes que o fluxo entra ou não na condição condicional.

Com base em simulações executadas no ambiente ModelSim, observou-se que o número médio de ciclos de clock por amostra foi de aproximadamente **5104 ciclos**. Considerando a necessidade de simular pulsos com uma frequência de **40 MHz** (intervalo de 25 ns), a frequência mínima necessária para uma FPGA executar esse modelo em tempo real seria:

$$f_{\text{necessária}} = 5104 \cdot 40 \text{ MHz} = \mathbf{204,16 \text{ GHz}} \quad (6.3)$$

Esse valor está muito acima da frequência máxima operável por qualquer FPGA disponível no mercado, tornando esta versão inviável para aplicações em tempo real com a frequência padrão do LHC. A frequência que esse simulador seria capaz de gerar uma amostra em uma FPGA que seja capaz de operar a 500MHz, é:

$$F_{\text{sim1}} = \frac{500 \text{ MHz}}{5104} \approx \mathbf{97,96 \text{ kHz}} \quad (6.4)$$

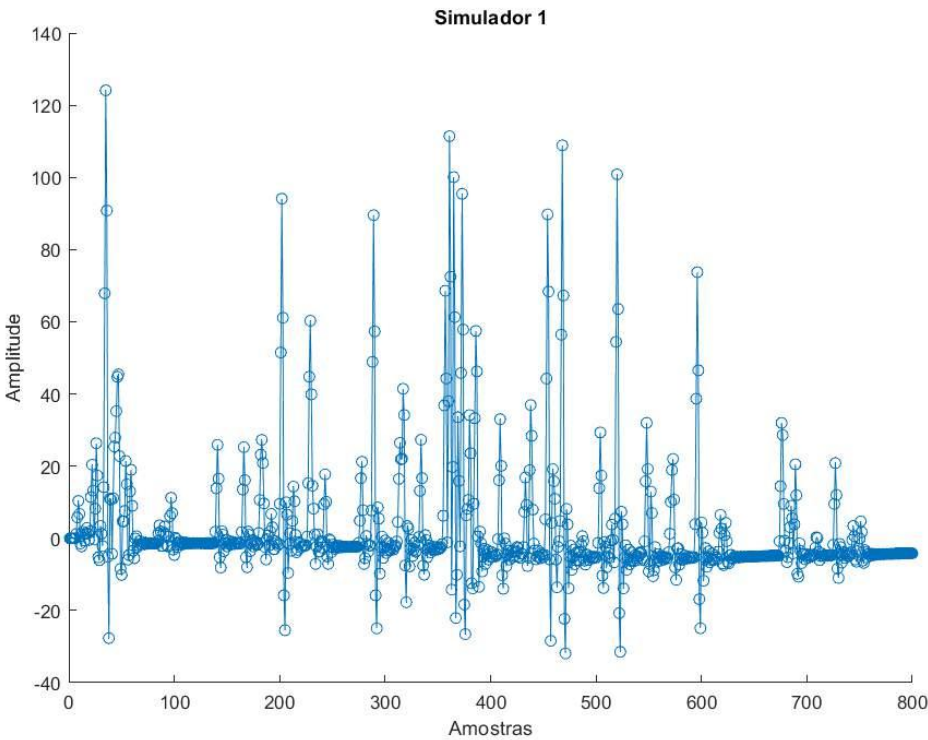
Nota-se, portanto, que esse simulador seria limitado a aplicações e testes com geração de pulsos em uma frequência inferior a 100 kHz, de acordo com a frequência definida de 500 MHz para a operação do módulo. A Tabela 10 resume o desempenho obtido para este modelo.

Tabela 10 – Desempenho do Simulador 1

Parâmetro	Valor
Número médio de ciclos por amostra	5104
Frequência necessária para 40 MHz	204,16 GHz
Representação numérica	Ponto fixo (inteiro)
Ganho aplicado na saída	1000 (ajustável no <code>top level</code>)

A Figura 28 mostra a forma de onda gerada pelo Simulador 1, com base em aproximadamente 800 amostras. A forma apresenta o perfil característico dos pulsos do TileCal, embora sem normalização direta.

Figura 28 – Saída do Simulador 1: forma de onda resultante com base em 800 amostras.

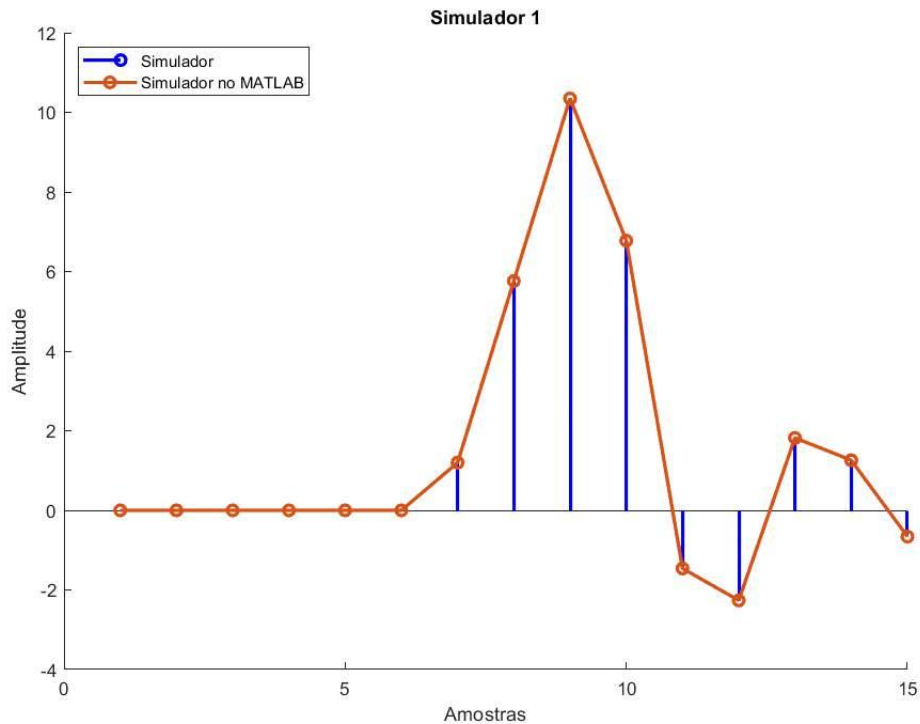


Fonte: Elaborado pelo autor (2025).

Para fins de comparação, a mesma sequência pseudoaleatória foi aplicada em um script MATLAB com os mesmos coeficientes discretizados, e mesmos valores aleatórios, como mostrado na Figura 29. O objetivo dessa comparação é validar, para as primeiras amostras, o simulador quanto a precisão dos resultados. A boa concordância entre os dois

modelos, logo, valida a consistência do filtro, mesmo com as limitações de desempenho da versão em hardware.

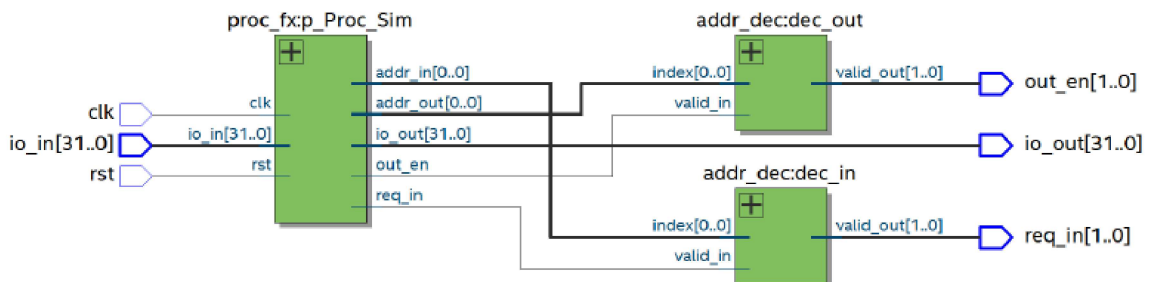
Figura 29 – Amostras geradas pelo Simulador 1 replicado em MATLAB.



Fonte: Elaborado pelo autor (2025).

Cabe destacar que a multiplicação final por 1000 aplicada na saída do simulador é meramente um fator de escala, utilizado para ampliar o intervalo de saída no ambiente de simulação. Esse fator pode, se desejado, ser removido ou compensado no nível superior do sistema (`top level`) sem afetar o número de ciclos ou o desempenho geral do modelo. Portanto, esse ganho não compromete a análise de desempenho, nem interfere na fidelidade relativa das formas de pulso.

Figura 30 – Estrutura RTL gerada para o Simulador 1, visualizada no ambiente Quartus.



Fonte: Elaborado pelo autor, com base na geração automática do RTL Viewer (2025).

A Figura 30 apresenta a estrutura RTL gerada automaticamente pelo Quartus para o Simulador 1. É possível observar uma densidade considerável de interconexões e blocos internos, consequência da execução sequencial de múltiplas equações de filtragem dentro de um único núcleo SAPHO, somada à presença do gerador de números pseudoaleatórios e das operações condicionais. Essa organização monolítica dificulta a reutilização de blocos, limita a escalabilidade e contribui diretamente para o alto número de ciclos por amostra. A simplificação e a modularização dessa estrutura são justamente os objetivos perseguidos nas versões subsequentes.

Apesar da elevada exigência de clock, o Simulador 1 cumpriu seu papel como estrutura de partida. Foi a partir dessa versão que se iniciou o refinamento estrutural e arquitetural dos modelos seguintes, com foco na redução da latência, no desacoplamento dos blocos internos e na adoção de formas mais eficientes de filtragem e controle. Os simuladores seguintes, como será visto a seguir, exploram outras estratégias justamente para mitigar os problemas de desempenho observados nesta primeira abordagem.

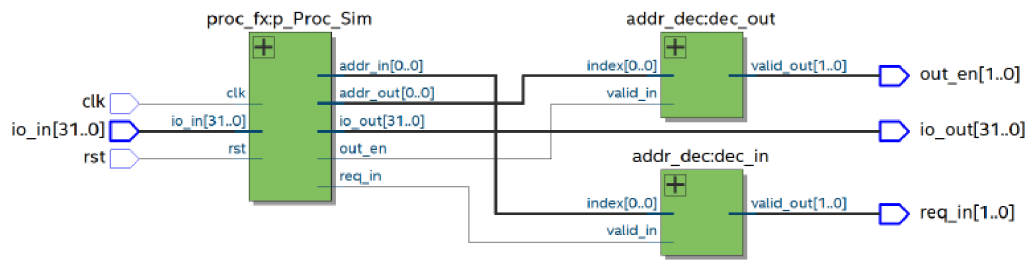
6.2 SIMULADOR 2

A segunda versão do simulador marca uma evolução em relação à estrutura adotada inicialmente, ao introduzir a unificação das equações de filtragem em uma única expressão combinada. Diferentemente do Simulador 1, que implementava as equações de forma separada, aqui o objetivo principal foi a redução do número de ciclos necessários para gerar uma amostra, mantendo a fidelidade da resposta simulada. O processador permanece operando em ponto fixo, e o gerador de números pseudoaleatórios continua embutido internamente ao núcleo SAPHO.

Essa escolha arquitetural traz implicações diretas na execução do código. Apesar da lógica de geração aleatória e filtragem ainda ocorrer dentro do mesmo núcleo, a compactação das operações em uma única função de transferência permitiu uma significativa redução na quantidade de instruções executadas por amostra, o que se refletiu na queda do número de ciclos necessários.

A Figura 31 apresenta o RTL Viewer gerado pelo Quartus para o Simulador 2. Como não houve alterações na estrutura de blocos do processador, o esquema permanece visualmente idêntico ao da primeira versão (Figura 30). A principal diferença reside no conteúdo do código em linguagem C utilizado como base para a conversão automática, localizado no bloco do processador `Proc_Sim`.

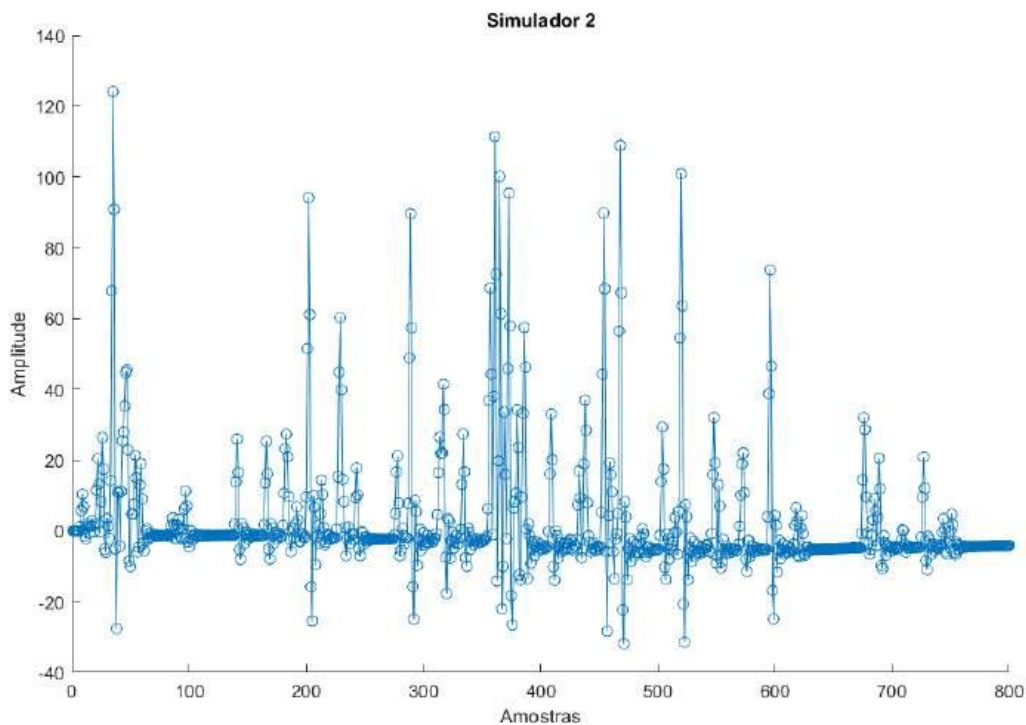
Figura 31 – RTL Viewer do Simulador 2, com estrutura idêntica ao do Simulador 1.



Fonte: Elaborado pelo autor (2025).

A seguir, apresenta-se o resultado da simulação temporal do sistema:

Figura 32 – Saída temporal gerada pelo Simulador 2.



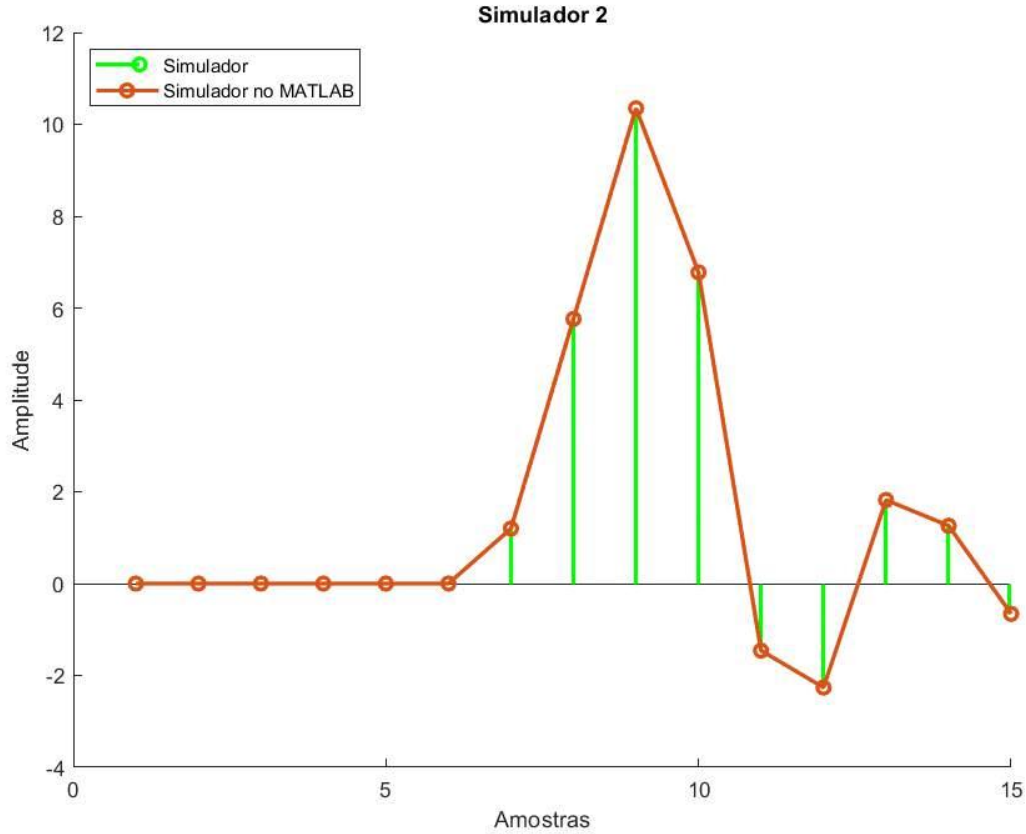
Fonte: Elaborado pelo autor (2025).

Observa-se na Figura 32 que o comportamento das amostras é idêntico ao obtido com o primeiro simulador, o que evidencia que a lógica de geração pseudoaleatória e de resposta ao pulso foi preservada. Esse resultado era esperado, visto que o gerador interno opera de forma determinística e a estrutura de filtragem implementa a mesma resposta total, ainda que por meio de uma única equação.

Ainda assim, foi realizado a comparação das primeiras amostras do simulador 2 com relação ao simulador criado em software, através do MATLAB, com o objetivo de

validação, como pode ser visto na Figura 33.

Figura 33 – Saída temporal gerada pelo Simulador 2.



Fonte: Elaborado pelo autor (2025).

No entanto, em termos de desempenho, houve um avanço considerável. A quantidade de ciclos de clock necessária para produzir uma nova amostra foi reduzida de 5104 para **3390 ciclos**. Essa economia de 1714 ciclos representa uma redução de aproximadamente 33,6% na carga de processamento. A nova frequência mínima necessária para executar o simulador com taxa de eventos de 40 MHz passa a ser:

$$F_{\text{sim2}} = 3390 \times 40 \text{ MHz} = \mathbf{135,6 \text{ GHz}} \quad (6.5)$$

Embora esse valor ainda esteja muito acima da capacidade real das FPGAs atuais, a queda significativa em relação aos 204,16 GHz do Simulador 1 reforça a eficácia da estratégia adotada. Ainda assim, o simulador não é viável para operação em tempo real com taxa de eventos do LHC, sendo mais adequado para análise de desempenho e estudos estruturais preliminares. A frequência a que esse modelo poderia operar, portanto, é:

$$F_{\text{sim2}} = \frac{500 \text{ MHz}}{3390} \approx \mathbf{147,49 \text{ kHz}} \quad (6.6)$$

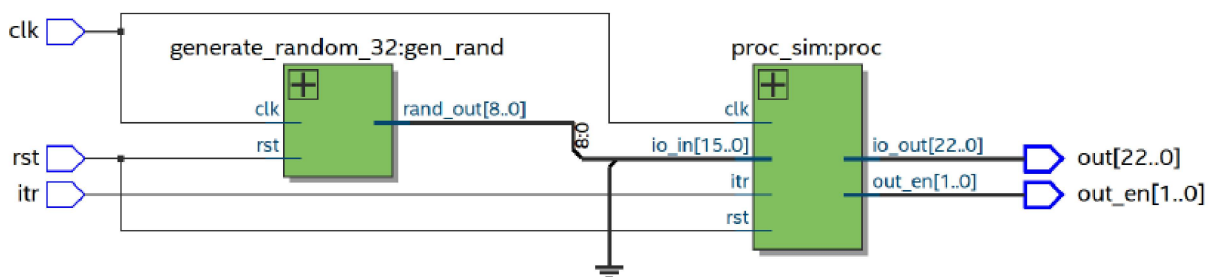
Por fim, é importante destacar que, apesar da mudança na estrutura algorítmica, as amostras geradas continuam sendo exatamente as mesmas do primeiro simulador. Isso ocorre porque o gerador pseudoaleatório permanece embutido no núcleo e a sequência numérica produzida segue a mesma lógica. Dessa forma, a equivalência nas formas de onda simuladas reforça a validade do Simulador 2 como um sucessor funcional e otimizado do Simulador 1.

6.3 SIMULADOR 3

A terceira versão do simulador representa um ponto de transição fundamental entre as arquiteturas sequenciais e a abordagem paralela da versão final. Nela, a principal mudança estrutural está na separação entre os blocos de geração de números pseudoaleatórios e os blocos de processamento. O gerador congruencial linear foi implementado externamente, em Verilog, enquanto o processador SAPHO passou a executar apenas a lógica de filtragem e verificação de ocupação, operando exclusivamente em ponto flutuante.

Essa reorganização teve como principal objetivo reduzir o número total de instruções dentro do processador e, com isso, diminuir o número de ciclos de clock por amostra. A arquitetura também passou a utilizar uma equação unificada de filtragem IIR, discretizada a partir da soma das cinco funções de transferência contínuas, conforme discutido nos Capítulos 3 e 5. Essa equação foi implementada por meio da estrutura de filtragem em *Direct Form II*, como detalhado na Seção 5.3.

Figura 34 – Estrutura RTL do Simulador 3, com separação entre gerador externo e processador SAPHO.



Fonte: Elaborado pelo autor (2025).

A Figura 34 evidencia a estrutura mais modular do Simulador 3. O gerador externo fornece diretamente a entrada ao processador, que executa a filtragem com base no valor recebido. Essa abordagem modular é mais próxima da configuração utilizada no Simulador 4, permitindo flexibilidade e potencial de paralelismo.

Na simulação realizada, o número médio de ciclos de clock para gerar uma nova amostra foi de apenas **78 ciclos**, representando uma redução drástica em relação às versões

anteriores. A frequência mínima necessária para simular a taxa de eventos de 40 MHz, correspondente à frequência do LHC, passa a ser:

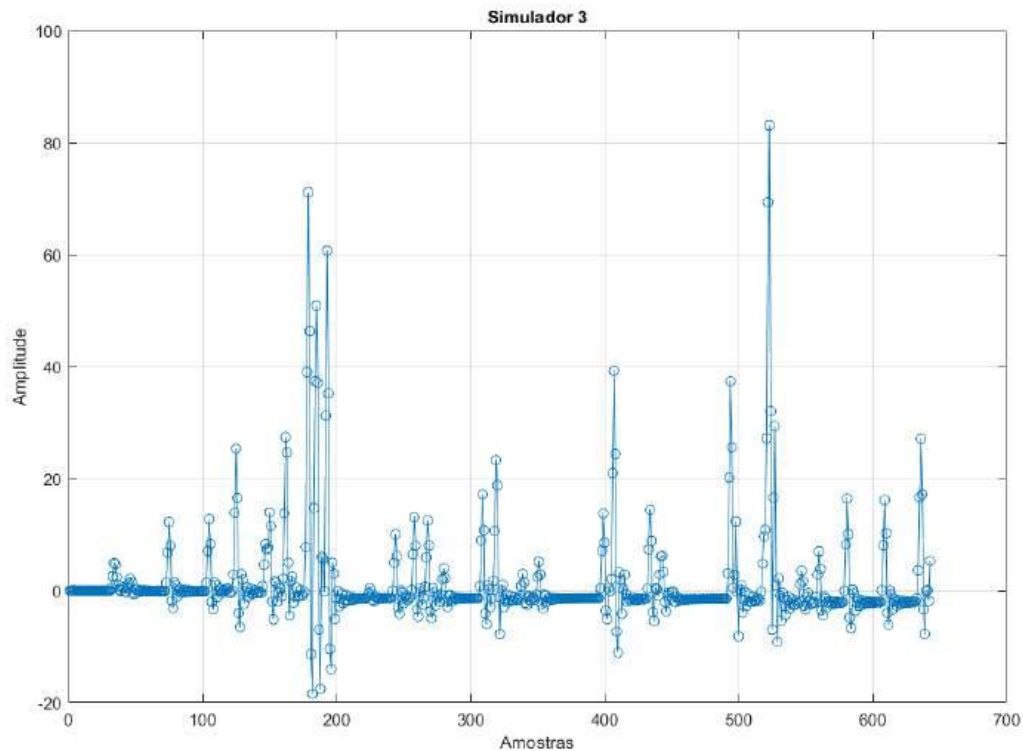
$$F_{\text{sim3}} = 78 \times 40 \text{ MHz} = \mathbf{3,12 \text{ GHz}} \quad (6.7)$$

Esse valor, embora ainda superior às frequências típicas de operação de FPGAs comerciais (geralmente abaixo de 1 GHz), já se encontra em uma ordem de grandeza mais próxima da viabilidade prática. Em comparação ao Simulador 1, cuja frequência exigida ultrapassava 200 GHz, esta versão apresenta um ganho superior a **65 vezes**, o que demonstra a eficiência da nova organização modular e da substituição de operações inteiras por ponto flutuante diretamente manipulável pela ULA. Portanto, esse simulador seria capaz de operar em:

$$F_{\text{sim3}} = \frac{500 \text{ MHz}}{78} \approx \mathbf{6,41 \text{ MHz}} \quad (6.8)$$

Esse valor bem superior permitiria testes e validação de metodologias mais complexas, devido a melhora na rapidez de geração de amostra. Na Figura 35 observa-se a saída gerada em tempo de simulação.

Figura 35 – Saída temporal do Simulador 3: forma de pulso gerada com arquitetura modular.



Fonte: (Elaborado pelo autor, com base na simulação em ModelSim).

Apesar da diferença estrutural em relação aos simuladores anteriores, a forma de pulso permanece coerente com o modelo físico representado. No entanto, nesta versão não foi possível realizar uma comparação direta com o modelo em MATLAB. A razão para isso está no comportamento não determinístico introduzido pela nova organização. Como o gerador de números aleatórios está localizado fora do processador, ele opera de forma contínua e atualiza seu valor em cada ciclo de clock. Por outro lado, o processador solicita esse valor somente após concluir uma amostra, através do comando `in(0)`. Adicionalmente, a presença da verificação condicional de ocupação interna ao processador (via `if (rnd_out < ccc)`) altera o número de instruções executadas dependendo da condição satisfeita, fazendo com que os instantes de leitura do gerador variem a cada evento.

Essa característica torna a replicação da sequência exata de valores e da forma de onda correspondente inviável em software, dificultando a validação cruzada com scripts MATLAB. No entanto, o comportamento geral do pulso e o desempenho em termos de tempo de execução validam a arquitetura como um avanço relevante na direção de uma simulação eficiente e escalável.

O Simulador 3, portanto, estabelece uma base sólida sobre a qual se construiu a versão paralela e final apresentada na próxima seção. A separação dos blocos, o uso de ponto flutuante, a modularidade e a redução expressiva da latência tornam essa versão um elo intermediário essencial na trajetória evolutiva dos simuladores.

6.4 SIMULADOR 4

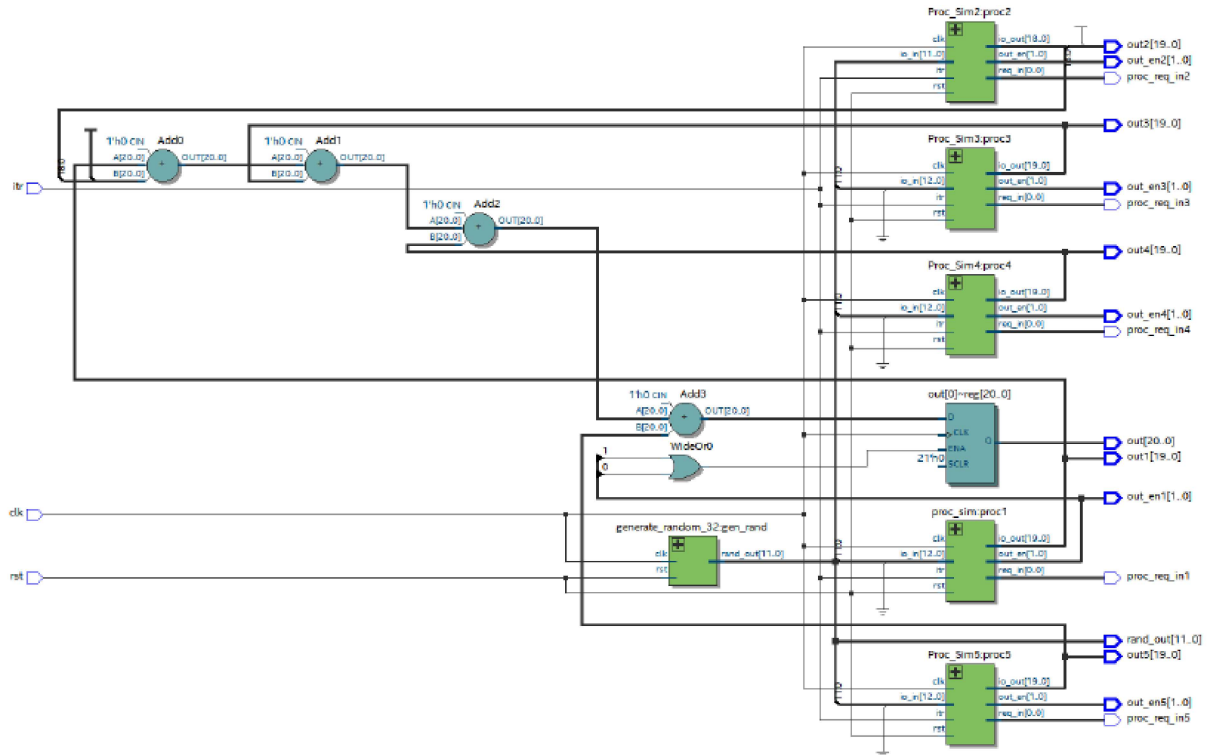
A versão final deste projeto adota uma organização explicitamente paralela: cinco processadores SAPHO operam simultaneamente, cada um executando uma equação de filtragem discreta do modelo (vide Seção 5.4.1). Essa nova arquitetura deriva das limitações observadas nas versões anteriores: no Simulador 1 e 2, a natureza sequencial, aliada ao processador em ponto fixo e à geração interna de números pseudoaleatórios, elevou o custo temporal por amostra; no Simulador 3, o ponto flutuante e a migração do gerador para o *top level* reduziram drasticamente os *clocks*/amostra, mas ainda havia margem para avanços. Aqui, ao distribuir as equações em cinco núcleos e somar as saídas no *top level*, busca-se maximizar o paralelismo de dados mantendo a fidelidade de forma e o sincronismo entre vias.

No nível superior (`pulse_sim`), o gerador pseudoaleatório externo produz continuamente `rand_out`, já com verificação de ocupação e mapeamento exponencial embutidos. Todos os núcleos SAPHO requisitam a entrada no mesmo ciclo (via `in(0)`), de tal modo que consomem o mesmo valor de `rand_out` e evoluem de forma alinhada. Como filtros de ordens diferentes possuem latências distintas, foram adicionadas operações neutras (`x = x + 0`) nos caminhos mais curtos, equalizando o número de ciclos entre processadores. Ao final de cada iteração, um somador síncrono (acionado por `out_en1`) agrega as cinco

saídas parciais y_i e produz a amostra $y[n]$.

A Figura 36 ilustra o *RTL Viewer* do arranjo: observam-se as cinco instâncias do processador (proc1 a proc5) e os blocos de soma. Essa visualização ajuda a compreender por que o custo temporal diminui — as operações que antes eram realizadas em série por um único núcleo, agora são distribuídas entre unidades que avançam em paralelo.

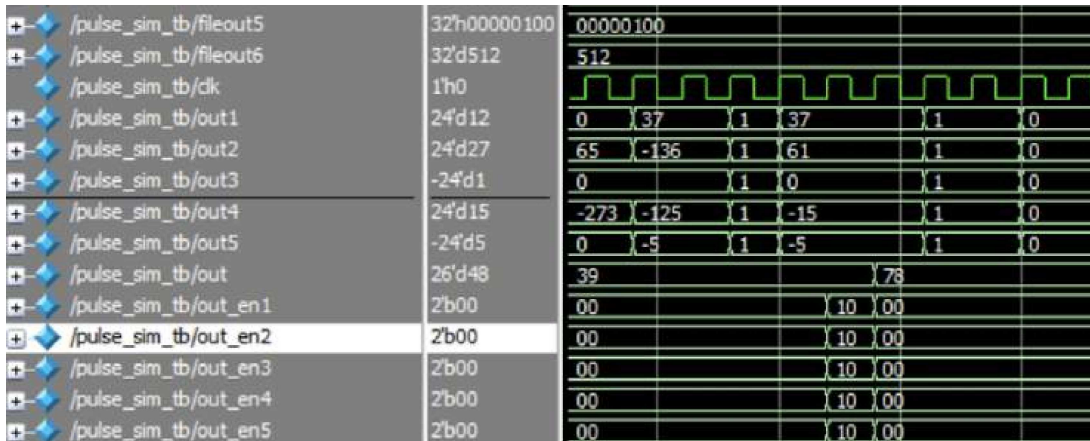
Figura 36 – Visualização RTL do Simulador 4: cinco núcleos SAPHO paralelos e somador no *top level*.



Fonte: Elaborado pelo autor (2025).

Na Figura 37, que apresenta um recorte da saída do simulador em um ambiente de simulação do ModelSim, é possível verificar o sincronismo entre os processadores, através das variáveis de habilitação. A saída é atualizada pela soma das contribuições individuais, no momento em que seu segundo bit assume nível alto.

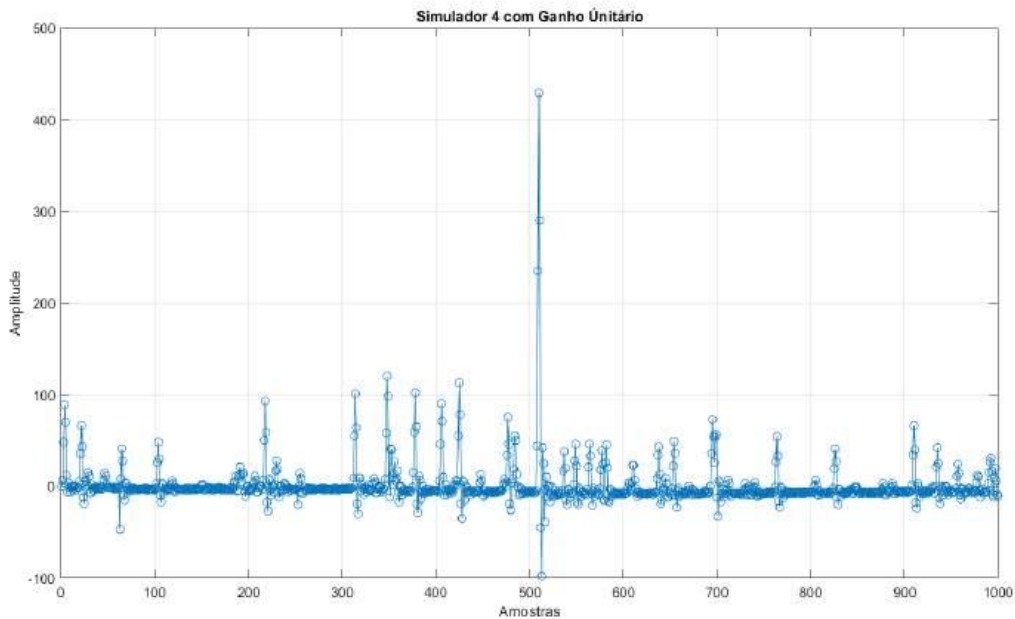
Figura 37 – Recorte da simulação no ModelSim das amostras de saída.



Fonte: Elaborado pelo autor (2025).

A Figura 38 apresenta uma janela de amostras de energia já após a soma das cinco vias ($y[n] = \sum_{i=1}^5 y_i[n]$), com ganho unitário. Esse sinal é o resultado do simulador no domínio discreto e concentra os efeitos de (i) geração pseudoaleatória com taxa de ocupação, (ii) mapeamento exponencial de amplitudes e (iii) convolução digital pelas cinco respostas implementadas em paralelo. Cada “pulso” visível é produzido por um evento aceito ($\text{rand_out} \neq 0$) e pela resposta agregada dos filtros; nos intervalos sem aceitação, o traçado tende ao *baseline*, mas exibe caudas persistentes devido à memória interna (polos próximos de $z = 1$).

Figura 38 – Amostras de saída do Simulador 4.



Fonte: Elaborado pelo autor (2025).

Do ponto de vista estatístico, as amplitudes aceitas seguem a distribuição da tabela exponencial (valores baixos mais frequentes, altos mais raros). Temporalmente, a subida é rápida (contribuições mais “rápidas” dos filtros) e o decaimento é lento (caudas dominadas pelos polos mais estáveis), reproduzindo a assinatura típica do TileCal. A sobreposição ocorre quando dois eventos aceitos chegam a distâncias menores que a duração efetiva da cauda: a linha de base se eleva localmente e picos sofrem leve deformação por soma de contribuições. Mesmo quando `rand_out` é zerado pela condição de ocupação, o traçado não colapsa imediatamente a zero — as caudas decaem até que a energia “armazenada” nos estados internos se dissipe. Em resumo, o traçado confirma três propriedades centrais da arquitetura: (a) sincronismo perfeito entre vias (mesma entrada no mesmo ciclo), (b) fidelidade de forma (subida/cauda e sobreposição condizentes com o modelo) e (c) prontidão para análise quantitativa.

Antes de entrar na validação numérica, convém quantificar o desempenho temporal. Aplicando ao Simulador 4 ($N_{\text{clocks}} = 31$), obtém-se:

$$F_{\text{sim4}} = 31 \times 40 \text{ MHz} = \mathbf{1,24 \text{ GHz}} \quad (6.9)$$

Esse resultado aproxima o sistema da viabilidade para operação em tempo real em dispositivos modernos e taxa de evento do LHC, sobretudo considerando-se possíveis otimizações físicas (mapeamento em DSPs, *retiming* e *pipelining*). De acordo com a frequência típica de operação de uma FPGA, essa versão é capaz de operar em:

$$F_{\text{sim4}} = \frac{500 \text{ MHz}}{31} \approx \mathbf{16,13 \text{ MHz}} \quad (6.10)$$

Esse valor aproxima o modelo do resultado desejado, e capacita a versão para inúmeras aplicações, principalmente no que refere a testes de metodologias. Em uma *RTL Simulation*, utilizando um *clock* de 20 ps (50 GHz), a Tabela 11 compara quantas amostras inteiras cada versão produz em $1 \mu\text{s}$. Essa leitura não substitui a exigência de 40 MHz, apenas ilustra a eficiência relativa quando se dispõe de um *clock* elevado de referência.

Tabela 11 – Amostras inteiras produzidas em $1 \mu\text{s}$ com $T_{\text{clk}} = 20 \text{ ps}$ (50 GHz).

Simulador	Ciclos/Amostra	T_{amostra} (ns)	Amostras em $1 \mu\text{s}$ (aprox.)
1	5104	102,08	9,80 (\rightarrow 9 inteiras)
2	3390	67,80	14,76 (\rightarrow 14 inteiras)
3	78	1,56	641,03 (\rightarrow 641 inteiras)
4	31	0,62	1612,90 (\rightarrow 1612 inteiras)

Para avaliar a aderência ao modelo de referência, primeiramente comparam-se **as primeiras 1000 amostras** da saída de hardware (HW) com as do MATLAB, em dois cenários:

- **Ganho 1:** $y^{\text{HW}} \times y^{\text{MAT}}$, ganho unitário. Mantissa = 13, expoente = 6.
- **Ganho 100:** y^{HW} contra $100 \times y^{\text{MAT}}$ (equivalente a dividir o HW por 100), ganho aplicado na tabela exponencial, com o objetivo de não influenciar no número de clocks dentro do processador. Mantissa = 19 e expoente = 6.

Os valores de mantissa e expoente são iguais aos mínimos possíveis em cada caso. As séries são alinhadas amostra a amostra. A seguir, definem-se as grandezas e métricas utilizadas, com suas finalidades:

1. Sinais de referência e janela de análise.

Definimos $y^{\text{HW}}[n]$ e $y^{\text{MAT}}[n]$ como as saídas de *hardware* e MATLAB, respectivamente, para $n = 1, \dots, N$ (aqui, $N = 1000$). A comparação é feita sobre essa janela fixa e alinhada.

2. Resíduo ponto a ponto (erro instantâneo).

O resíduo mede, em cada amostra, a diferença entre o hardware e a referência:

$$e[n] = y^{\text{HW}}[n] - y^{\text{MAT}}[n].$$

Ele é a base para todas as métricas subsequentes (dispersão, energia do erro, etc.).

3. Estatísticas básicas (média, variância e desvio-padrão).

Para qualquer sequência $z[n]$ na janela:

$$\mu_z = \frac{1}{N} \sum_{n=1}^N z[n], \quad \text{Var}(z) = \frac{1}{N-1} \sum_{n=1}^N (z[n] - \mu_z)^2, \quad \sigma_z = \sqrt{\text{Var}(z)}.$$

Usaremos μ , $\text{Var}(\cdot)$ e σ para caracterizar tanto o sinal de saída (y^{HW}) quanto o resíduo e .

4. Correlação de Pearson (ρ).

Quantifica a co-variabilidade linear entre as séries, em $[-1,1]$:

$$\rho = \frac{\sum_{n=1}^N (y^{\text{HW}}[n] - \mu_{y^{\text{HW}}}) (y^{\text{MAT}}[n] - \mu_{y^{\text{MAT}}})}{(N-1) \sigma_{y^{\text{HW}}} \sigma_{y^{\text{MAT}}}}.$$

Valores próximos de 1 indicam preservação de forma (morfologia do pulso) e alinhamento temporal.

5. Coeficiente de determinação (R^2).

Com mapeamento identidade (mesma escala nas séries), interpretamos:

$$R^2 = 1 - \frac{\text{Var}(e)}{\text{Var}(y^{\text{HW}})},$$

ou seja, a fração da variância do hardware explicada pela referência. Quanto mais perto de 1, melhor a aderência global.

6. RMSE e NRMSE (erro quadrático médio e sua versão normalizada).

O erro quadrático médio resume a energia do erro:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^N e[n]^2}.$$

Para torná-lo adimensional e comparável entre cenários, normalizamos pela dispersão do hardware:

$$\text{NRMSE} = \frac{\text{RMSE}}{\sigma_{y^{\text{HW}}}} \times 100\%.$$

A NRMSE indica, em porcentagem, o quão “grande” é o erro frente à variabilidade do sinal de saída.

7. Erro percentual por razão de variâncias (Erro (%)).

Métrica principal desta seção: fração de energia não explicada pela referência,

$$\text{Erro (\%)} = \frac{\text{Var}(e)}{\text{Var}(y^{\text{HW}})} \times 100\%.$$

Se ambas as séries forem escaladas por um ganho comum k , numerador e denominador escalam por k^2 , mantendo o quociente invariante. (Essa invariância também vale para a NRMSE, pois RMSE e $\sigma_{y^{\text{HW}}}$ escalam por k .)

8. Métricas auxiliares (para diagnóstico fino).

Além das métricas “de forma” acima, reportamos: *viés* μ_e (offset médio), *MAE* $= \frac{1}{N} \sum |e[n]|$, erro máximo absoluto $|e|_{\max}$, diferença de pico Δp (absoluta e relativa), diferença no tempo ao pico Δt_p (em amostras) e diferença relativa de área ΔA (soma discreta). Essas leituras ajudam a localizar se o desvio é um *offset* constante, um leve *gain mismatch* ou pequenas assimetrias temporais.

A Tabela 12 consolida as métricas primárias para ganho 1 e ganho 100; as leituras são interpretadas no texto logo a seguir.

Tabela 12 – Métricas primárias de aderência entre HW e MATLAB (1000 amostras).

Cenário	$\text{Var}(y^{\text{HW}})$	$\text{Var}(e)$	Erro (%)	ρ	R^2	NRMSE
Ganho 1	615,8905	3,7452	0,6081%	0,99696	0,993919	10,362%
Ganho 100	6.169.733,65	27,9796	0,0004535%	0,999998	0,999995	0,219%

Sob ganho 1, o erro percentual é da ordem de 0,61%, com $\rho \approx 0,997$ e $R^2 \approx 0,994$. Em ganho 100, os desvios tornam-se praticamente desprezíveis: Erro (%) abaixo de $5 \times 10^{-4}\%$, correlação praticamente unitária e NRMSE em torno de 0,22%. Esses resultados indicam que a morfologia do pulso é preservada; a diferença entre a razão de variâncias e

a NRMSE em ganho 1 relaciona-se a um *viés* médio no resíduo, que infla o RMSE sem afetar tanto a dispersão relativa.¹

Para identificar onde estão as diferenças residuais, a Tabela 13 lista métricas secundárias (viés, erros pontuais e integrativos), seguidas de definições para leitura.

Tabela 13 – Métricas secundárias: viés, erros pontuais e medidas integrativas (1000 amostras).

Cenário	Viés	MAE	RMSE	$ e _{\max}$	Δp	Δp (%)	Δt_p	ΔA (%)
Ganho 1	−1,695	1,697	2,572	51,877	−1,5335	−0,356%	0	−67,96%
Ganho 100	−1,298	2,452	5,444	97,709	−10,3499	−0,024%	0	−0,521%

Leituras e conclusões a partir da Tabela 13:

1. Sincronismo temporal: Em ambos os cenários, $\Delta t_p = 0$, indicando que o instante do pico é preservado amostra a amostra. Isso confirma alinhamento temporal entre HW e referência e corrobora que todos os núcleos consomem a mesma entrada no mesmo ciclo.
2. *Amplitude de pico*. A diferença relativa de pico é pequena e melhora com a escala: −0,356% (ganho 1) e −0,024% (ganho 100). Ou seja, a reprodução da amplitude máxima é muito próxima da referência, sobretudo quando as séries operam com maior faixa dinâmica efetiva.
3. Magnitude típica do erro: MAE e RMSE mantêm-se baixos nos dois casos, e o erro absoluto máximo ($|e|_{\max}$) aparece como evento isolado, típico de trechos de transiente rápido. Em termos de forma de onda, esses valores são coerentes com a alta correlação e com a baixa fração de variância não explicada reportadas na tabela de métricas primárias.
4. Viés e impacto integrativo: O resíduo apresenta viés negativo moderado (−1,695 em ganho 1; −1,298 em ganho 100). Esse offset tem pouco efeito sobre correlação e sobre a razão de variâncias, mas afeta medidas integrativas: em ganho 1, a soma do viés ao longo de $N = 1000$ amostras resulta em ≈ -1695 , valor compatível com a diferença de áreas observada ($\Delta A = -67,96\%$). Em ganho 100, o efeito relativo do mesmo viés dilui-se e ΔA cai para −0,521%.
5. Síntese: As métricas secundárias reforçam a leitura de fidelidade obtida nas métricas primárias: a forma é preservada (pico e tempo ao pico), as diferenças pontuais

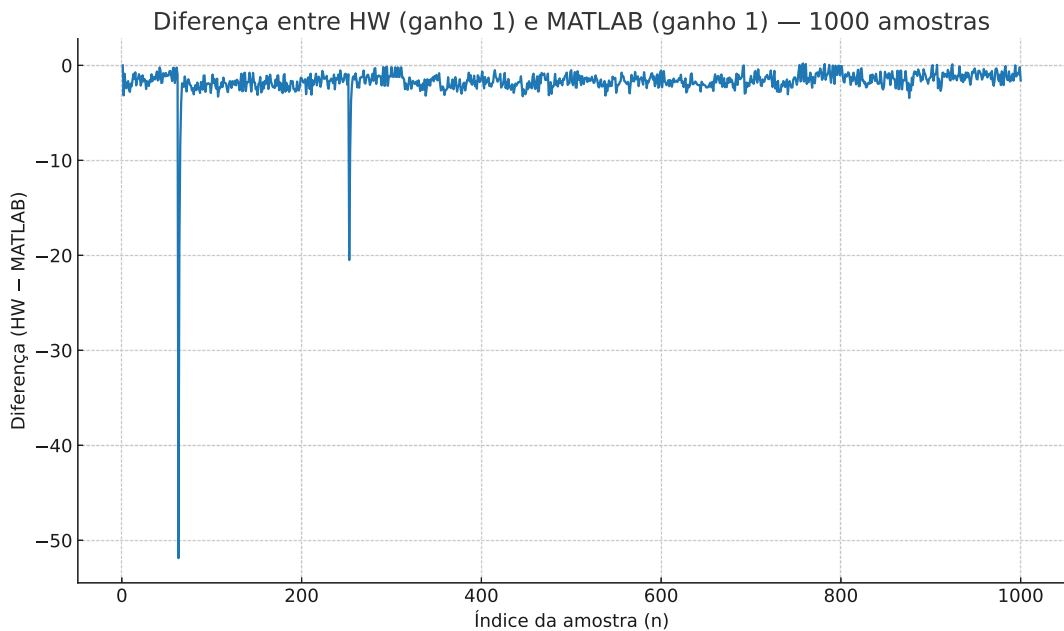
¹ Nas 1000 amostras avaliadas, o resíduo médio foi de −1,69 (ganho 1) e −1,30 (ganho 100). Ao remover o DC, a NRMSE em ganho 1 cai de 10,36% para 7,79%; em ganho 100, fica em $\sim 0,213\%$.

são pequenas e o principal desvio residual é um componente DC. Quando esse componente médio é removido, as métricas baseadas em erro quadrático convergem para patamares ainda mais baixos, sem indícios de deformação morfológica do pulso.

Em conjunto, esses resultados indicam que o Simulador 4 reproduz com alta fidelidade a dinâmica temporal e a morfologia das amostras do modelo de referência. As discrepâncias observadas concentram-se em um *offset* pequeno e estável, cuja mitigação é direta (remoção de DC ou harmonização de arredondamentos), não afetando o caráter determinístico, o sincronismo entre vias nem as conclusões sobre desempenho. Apesar de o erro percentual no cenário de ganho unitário ser baixo, é notório que o ganho proporciona uma diferença considerável na precisão dos resultados, pois com ganhos altos, a saída se comporta como um número decimal (caso dividisse a saída por 100 no cenário 2, por exemplo).

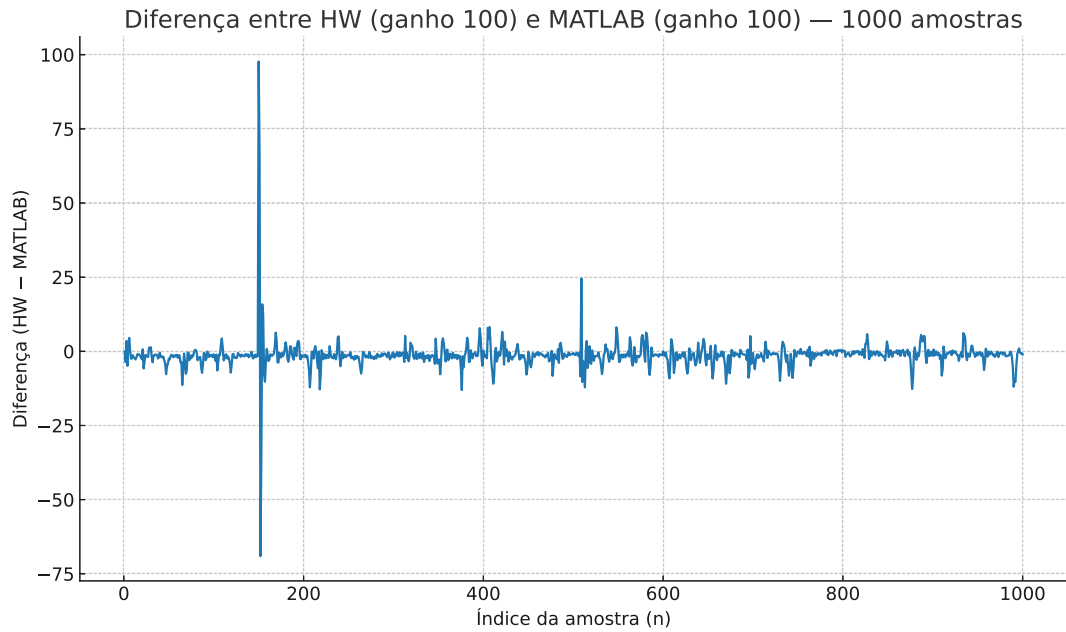
A inspeção visual do resíduo $e[n] = y^{\text{HW}}[n] - y^{\text{MAT}}[n]$ complementa os números. As Figuras 39 e 40 mostram flutuações de baixa amplitude centradas em zero, sem tendência sistemática — assinaturas compatíveis com discretizações, arredondamentos e pequenas assimetrias de coleta.

Figura 39 – Diferença entre HW (ganho 1) e MATLAB (ganho 1), 1000 amostras.



Fonte: Elaborado pelo autor (2025).

Figura 40 – Diferença entre HW (ganho 100) e MATLAB (ganho 100), 1000 amostras.

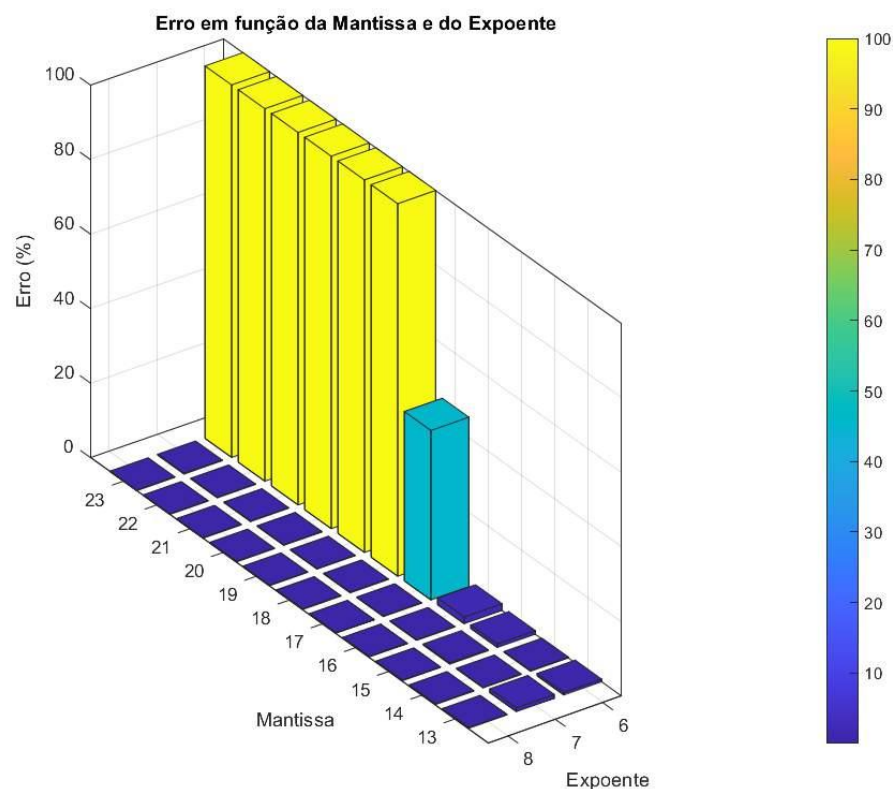


Fonte: Elaborado pelo autor (2025).

Diante das distorções observadas em função do ganho, foi realizada uma análise sobre a variação dos parâmetros de **mantissa** e **expoente** do processador, relacionando-os com o erro percentual. O objetivo foi identificar até que ponto a redução da precisão numérica (diminuição no número de bits) mantém o erro dentro de limites aceitáveis.

O ponto de partida para essa análise foi o padrão IEEE 754 (KAHAN; GOLDBERG, 1991). Avaliou-se a variação da mantissa de 23 \rightarrow 13 bits e do expoente de 8 \rightarrow 6 bits. Para valores inferiores, ocorre erro de compilação. A Figura 41 apresenta os erros percentuais obtidos nesse cenário.

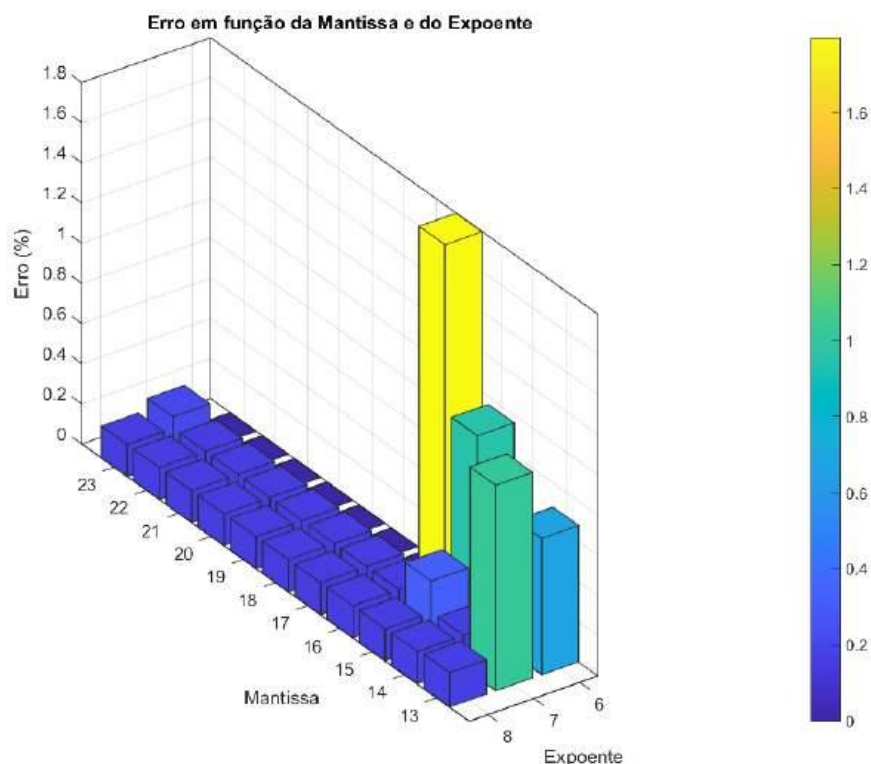
Figura 41 – Erro percentual em relação à variação de mantissa e expoente do processador.



Fonte: Elaborado pelo autor (2025).

Observa-se no gráfico que, para valores de mantissa iguais ou superiores a 18 com expoente 6, o erro alcança 100%. Além disso, o crescimento do erro torna-se evidente a partir de mantissa igual a 16, intensificando-se a partir de 17. Para facilitar a interpretação, foram omitidos os valores de erro acima de 20%, resultando em uma visualização mais clara, mostrada na Figura 42.

Figura 42 – Erro percentual em função dos parâmetros do processador (versão ampliada).



Fonte: Elaborado pelo autor (2025).

A análise detalhada desse gráfico evidencia que algumas configurações podem ser consideradas aceitáveis, a depender dos requisitos do sistema. Para expoente igual a 7, apenas a mantissa de 13 produziu erro superior a 0,5%. Assim, a configuração com **mantissa = 14 e expoente = 7** representa o menor custo em termos de precisão (soma mantissa + expoente +1) com desempenho adequado, mantendo o erro abaixo de 0,5% (0,1615%).

De forma semelhante, para expoente = 6 e mantissa = 14, o erro obtido foi de 0,1882%, valor suficientemente baixo para diversas aplicações. Essa configuração corresponde ao menor custo entre todas aquelas que mantêm o erro percentual inferior ou igual a 0,5%, podendo ser considerada uma solução eficiente para validação de metodologias e uso em cenários de restrição de recursos. O Apêndice B apresenta gráficos para ilustrar a diferença ponto a ponto entre o simulador 4 e o simulador do MATLAB das combinações destacadas, além da tabela com os valores dos erros percentuais para todas as combinações.

Para registrar em um só lugar os parâmetros de desempenho e o arranjo lógico desta versão, a Tabela 14 sintetiza os aspectos que caracterizam o Simulador 4.

Tabela 14 – Resumo de desempenho e arranjo lógico do Simulador 4.

Parâmetro	Valor
Número de núcleos SAPHO	5 (paralelos; uma equação por núcleo)
Entrada comum	<code>rand_out</code> (gerador externo; ocupação + tabela exp.)
Sinal de sincronização da soma	<code>out_en1</code> (amostras alinhadas)
Operações de balanceamento	$x = x + 0$ nos caminhos mais curtos
Representação numérica	Ponto flutuante
Ciclos por amostra	31
Frequência mínima para 40 MHz	1,24 GHz
Bloco de soma	Verilog no <i>top level</i> (<code>always @(posedge clk)</code>)

O arranjo resumido na Tabela 14 precisa ser lido à luz das evidências quantitativas apresentadas anteriormente. Do ponto de vista de **desempenho**, a produção de uma amostra a cada **31 ciclos** coloca esta versão em outro patamar: a frequência mínima para sustentar 40 MHz fica em **1,24 GHz**, o que representa reduções de aproximadamente $165\times$, $109\times$ e $2,5\times$ em relação aos Simuladores 1, 2 e 3, respectivamente. Esses ganhos derivam de três decisões arquiteturais que se reforçam mutuamente: (i) paralelismo real com cinco núcleos SAPHO; (ii) equalização de latências por inserção de operações neutras nos caminhos curtos; e (iii) desacoplamento do gerador, que fornece a mesma entrada para todos os núcleos no mesmo ciclo, eliminando dependências internas desnecessárias.

No quesito da **fidelidade**, as métricas primárias indicam que a fração de variância não explicada é residual e decresce com a ampliação da faixa dinâmica efetiva (ganho 100). A correlação quase unitária, R^2 próximo de 1 e NRMSE subpercentual corroboram que a morfologia do pulso foi preservada. As métricas secundárias complementam a leitura: o instante do pico coincide ($\Delta t_p = 0$), a diferença relativa do pico é pequena (e torna-se desprezível em alto ganho) e o principal desvio remanescente manifesta-se como *offset* médio no resíduo, facilmente mitigável por remoção de componente DC ou harmonização de arredondamentos. Em síntese, o que resta explicar não é deformação de forma, mas um deslocamento quase constante de nível.

As **implicações práticas** desse conjunto são diretas. Primeiro, o *top level* atual — gerador externo com verificação de ocupação e tabela exponencial embutidas, cinco processadores em paralelo e somador síncrono — provê um caminho claro para síntese em FPGA de última geração, com espaço para: mapeamento explícito em blocos DSP para multiplicações, retiming e *pipelining* para reduzir caminho crítico, e pequenas reordenações locais que preservem o determinismo do protocolo de entrada (`in(0)`) e do disparo de soma (`out_en1`). Segundo, a sincronização por consumo simultâneo de `rand_out` confere reprodutibilidade ao experimento: o mesmo *seed* e o mesmo agendamento de leitura produzem saídas bit a bit congruentes no domínio simulado.

Ainda que os **números finais de implementação física** (ocupação de LUT, FF, DSP, BRAM e F_{\max}) dependam da família-alvo e das escolhas de *place & route*, o projeto

já começa com propriedades favoráveis à temporização: caminhos aritméticos curtos em cada núcleo, soma final em um único estágio e ausência de dependências inter-núcleo além da entrada e do pulso de habilitação. Caso a frequência de síntese atingida na prática fique abaixo de 1,24 GHz, há alternativas compatíveis com a arquitetura apresentada: distribuição de canais entre múltiplos domínios de *clock* (multiclock), divisão de carga por interleaving temporal e/ou duplicação do arranjo para manter a taxa efetiva de amostragem no barramento de saída — sem alterar o modelo matemático nem o protocolo de sincronismo.

Por fim, as **conclusões** que emergem desta seção alinham o objetivo central da dissertação: (i) a reorganização em cinco núcleos paralelos reduz de forma decisiva o custo temporal por amostra, aproximando a execução da janela operativa desejada; (ii) a estratégia de sincronismo garante determinismo na amostragem e soma, requisito essencial para validação e comparação com modelos de referência; e (iii) a aderência quantitativa demonstra que a arquitetura preserva a forma do pulso, com discrepâncias residuais atribuíveis majoritariamente a nível DC. Esses resultados consolidam o Simulador 4 como a versão final apta a sustentar estudos de algoritmos em regime de alta taxa de eventos, ao mesmo tempo em que oferecem um ponto de partida robusto para exploração física em FPGA e extensões futuras (multi-canais, variações de ocupação e varredura sistemática de coeficientes).

7 CONSIDERAÇÕES FINAIS

Esta dissertação propôs, desenvolveu e avaliou uma família de simuladores de pulsos do TileCal com implementação em *hardware* reconfigurável, tendo como norte (i) a reprodução fiel da morfologia temporal do pulso sob condições de alta taxa de eventos (40 MHz) e (ii) a redução sistemática do custo temporal por amostra, de forma a aproximar a execução em tempo (quase) real. O contexto de Física de Altas Energias exige processamento determinístico, latência baixa e previsível, além de modelos controláveis para estudo de *pile-up*, calibração e validação de algoritmos embarcados. O arcabouço construído ao longo do texto atende a esses requisitos ao oferecer uma cadeia completa: geração pseudoaleatória com taxa de ocupação e mapeamento exponencial, filtragem digital com diferentes realizações estruturais e integração em *top level* para observação e mensuração de desempenho.

Metodologia e escolhas de projeto. Partiu-se da plataforma SAPHO para compilar C em Verilog, o que permitiu controlar explicitamente o *micro-pipeline* e medir, com granularidade de instrução, a quantidade de *clocks* por amostra. Separou-se a geração de dados do processamento: o gerador congruencial, com verificação de ocupação e índice para tabela exponencial, foi alocado fora do processador quando necessário, reduzindo dependências e abrindo espaço para paralelismo. No bloco de filtragem, empregaram-se estruturas clássicas (incluindo Forma Direta II quando vantajosa), com atualização explícita de estados internos. A validação quantitativa foi conduzida por comparação direta amostra a amostra com um modelo de referência em MATLAB, utilizando métricas de variância, correlação, R^2 , NRMSE e leituras complementares (viés, erro máximo, pico e área).

Trajetória evolutiva dos simuladores. A construção em quatro versões mostrou, com números, como decisões arquiteturais se traduzem em latência:

- **Simulador 1** (ponto fixo; geração interna; equações separadas): 5104 *clocks*/amostra \Rightarrow 204,16 GHz. Contribuíram para o custo temporal o desvio condicional (ocupação) e conversões fixo \leftrightarrow flutuante por conta de coeficientes fracionários.
- **Simulador 2** (ponto fixo; geração interna; equações combinadas): 3390 *clocks*/amostra \Rightarrow 135,6 GHz. A unificação reduziu instruções, mantendo fidelidade de forma.
- **Simulador 3** (ponto flutuante; gerador externo; forma direta II para a equação unificada): 78 *clocks*/amostra \Rightarrow 3,12 GHz. A mudança de domínio numérico eliminou normalizações custosas e melhorou a estabilidade numérica dos coeficientes.
- **Simulador 4** (ponto flutuante; 5 processadores SAPHO em paralelo; soma no *top level*): 31 *clocks*/amostra \Rightarrow 1,24 GHz. O paralelismo por vias, somado ao

balanceamento de latências com operações neutras ($\mathbf{x} = \mathbf{x} + 0$), alinhou as saídas no mesmo ciclo de *clock* e trouxe o arranjo para uma faixa de viabilidade muito mais próxima do desejável.

Resultados finais (Simulador 4) e sua leitura. Além da redução drástica de latência, a arquitetura paralela preservou a morfologia do pulso. Nas primeiras 1000 amostras, a comparação $HW \times MATLAB$ indicou, em *ganho 1*, fração de variância não explicada de $\sim 0,61\%$, $\rho \approx 0,997$ e $R^2 \approx 0,994$; em *ganho 100*, $\sim 4,5 \times 10^{-4}\%$, correlação praticamente unitária e NRMSE subpercentual. O instante de pico foi preservado ($\Delta t_p = 0$), e as diferenças de área, quando presentes em baixo ganho, foram atribuíveis majoritariamente a um pequeno *offset* médio do resíduo, sem implicar deformação da forma. Em termos de rendimento, com $T_{clk} = 20$ ps, o Simulador 4 entrega aproximadamente 1612 amostras inteiras por $1 \mu s$, valor que ilustra a eficiência relativa frente às versões anteriores.

Relevância para Física de Altas Energias. O artefato produzido é mais que um conjunto de códigos: trata-se de uma *plataforma de ensaio* para algoritmos sob condições realistas de ocupação e *pile-up*, com sincronismo estrito entre vias, determinismo de amostragem e interface clara para integração com fluxos de leitura. Isso encurta o ciclo entre *prototipagem em software* e *implante em hardware*, oferecendo terreno seguro para estudos de calibração, rejeição de ruído, avaliação de *trigger* e desenvolvimento incremental de algoritmos, inclusive quando múltiplos canais ou regiões do detector precisarem ser simulados em paralelo.

Limitações e leitura realista. A frequência mínima teórica de 1,24 GHz é de fato desafiadora para muitas famílias de FPGA, já que depende de fatores como F_{max} , ocupação de DSP/BRAM/LUT, estratégia de *place&route*, *retiming* e profundidade de *pipeline*. Ainda assim, a própria arquitetura proposta mostra-se *amigável à temporização*, com caminhos aritméticos curtos por núcleo, somador final simples e dependências inter-núcleo restritas à entrada comum (`rand_out`) e ao pulso de habilitação.

Vale ressaltar que, mesmo quando não se atinge o patamar de 40 MHz exigido para operação em tempo real no LHC, o simulador permanece útil em frequências mais baixas. Nessas condições, pode ser empregado para testes funcionais, validação de metodologias e estudos comparativos de reconstrução, preservando o determinismo e a fidelidade do modelo. Caso se deseje aproximar-se da frequência alvo, ainda existem estratégias clássicas de otimização, como *time-interleaving*, duplicação/particionamento de arranjos, domínios de *clock* escalonados e aprofundamento seletivo de *pipeline* em operações críticas.

Contribuições consolidadas.

- *Método de projeto* reproduzível, do C ao Verilog, com mensuração objetiva de *clocks*/amostra e seleção informada de estruturas de filtro (incluindo Forma Direta II).

- *Arquitetura paralela* com cinco processadores SAPHO sincronizados, entrada comum e soma síncrona no *top level*, reduzindo a latência para 31 *clocks*/amostra.
- *Validação quantitativa* rigorosa frente ao MATLAB em 1000 amostras, com métricas de variância, correlação, R^2 , NRMSE e análises pontuais (pico, área, erro máximo).

Perspectivas e desdobramentos.

- *Síntese física e perfil de recursos*: caracterizar LUT/FF/DSP/BRAM, F_{\max} e consumo, com mapeamento explícito em blocos DSP e *retiming/pipelining* para aproximar ou superar 1,24 GHz.
- *Escalonamento multi-canais*: instanciar bancos de núcleos em paralelo, preservando sincronismo e explorando particionamento por canal/região.
- *Robustez e não idealidades*: inserir ruído, jitter, saturação e variação de coeficientes; varrer cenários de ocupação e *pile-up*; estudar tolerância a *bit-flips*.
- *Mitigação de offset e harmonização numérica*: padronizar arredondamentos, remover componentes DC quando apropriado e avaliar o impacto na NRMSE e nas métricas integrativas.
- *Integração com cadeias reais de leitura*: anexar protocolos, buffers e janelas deslizes, elevando o realismo e preparando o caminho para uso em bancadas de teste.

Síntese final. O objetivo central foi alcançado: reduzir ordens de grandeza no custo temporal por amostra *sem* abrir mão da fidelidade de forma, culminando em uma versão paralela, sincronizada e validada (31 *clocks*/amostra, 1,24 GHz teóricos). Mais do que um simulador pontual, o trabalho entrega um *método* de concepção e validação que pode ser transposto para outros blocos e cenários de instrumentação em Física de Altas Energias, encurtando a distância entre a modelagem em alto nível e a implementação prática em FPGA.

REFERÊNCIAS

- AAD, G. *et al.* The atlas tile calorimeter: Design, construction and performance. **The European Physical Journal C**, v. 70, p. 1193–1236, 2010. Acessado em: 18 fev. 2025. Disponível em: <https://link.springer.com/article/10.1140/epjc/s10052-010-1508-y>.
- AAD, G. *et al.* Studies of the performance of the atlas detector using cosmic-ray muons. **The European Physical Journal C**, v. 71, n. 3, 2011. ISSN 1434-6052. Disponível em: <https://doi.org/10.1140/epjc/s10052-011-1593-6>.
- AAMODT, K. *et al.* The alice experiment at the cern lh. **Journal of Instrumentation**, v. 3, p. S08002, 2008.
- AGARAS, M. N. e. a. The atlas tile calorimeter performance and its upgrade towards the high-luminosity lh. **arXiv preprint arXiv:2105.09099**, 2021. Acesso em: 12 dez. 2024. Disponível em: <https://arxiv.org/pdf/2105.09099>.
- AGUIAR, M. S. **Implementação Embarcada em FPGA de Métodos Visando a Reconstrução Online de Energia no Calorímetro Hadrônico do Experimento ATLAS**. Dissertação (Dissertação de Mestrado em Engenharia Elétrica) — Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, 2023.
- APOLLINARI, G. *et al.* **High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report**. Geneva: CERN Yellow Reports: Monographs, 2015.
- ARISTOTLE. **Physics**. Cambridge: Harvard University Press, 1984. (Loeb Classical Library). ISBN 9780674993302. Disponível em: https://www.loebclassics.com/view/aristotle-physics/1934/pb_LCL228.3.xml.
- ATLAS COLLABORATION. **ATLAS: Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN**. Geneva, 1994. Acessado em: 18/06/2025. Disponível em: <https://cds.cern.ch/record/290968>.
- ATLAS COLLABORATION. **ATLAS Liquid Argon Calorimeter Technical Design Report**. Geneva, 1996. Acessado em: 13/02/2025. Disponível em: <https://cds.cern.ch/record/331061>.
- ATLAS COLLABORATION. **ATLAS Tile Calorimeter Technical Design Report**. Geneva, 1996. Acessado em: 23/02/2025. Disponível em: <https://cds.cern.ch/record/331062>.
- ATLAS COLLABORATION. The atlas experiment at the cern large hadron collider. **Journal of Instrumentation**, v. 3, p. S08003, 2008. Disponível em: <https://doi.org/10.1088/1748-0221/3/08/S08003>.
- ATLAS COLLABORATION. The atlas tile calorimeter calibration system. **Journal of Instrumentation**, v. 5, p. P09003, 2010. Acessado em: 15 jan. 2025. Disponível em: <https://doi.org/10.1088/1748-0221/5/09/P09003>.
- ATLAS COLLABORATION. **Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC**. 2012. 1–29 p.

ATLAS COLLABORATION. Muon response and calibration of the atlas tile calorimeter. **European Physical Journal C**, v. 73, p. 2306, 2013. Acessado em: 15 jan. 2025. Disponível em: <https://doi.org/10.1140/epjc/s10052-013-2306-0>.

ATLAS COLLABORATION. Performance of the atlas tile calorimeter charge injection system. **Journal of Instrumentation**, v. 9, p. P04024, 2014. Acessado em: 15 jan. 2025. Disponível em: <https://doi.org/10.1088/1748-0221/9/04/P04024>.

ATLAS COLLABORATION. Atlas tile calorimeter performance: laser calibration system. **Journal of Instrumentation**, v. 12, p. P05019, 2017. Acessado em: 15 jan. 2025. Disponível em: <https://doi.org/10.1088/1748-0221/12/05/P05019>.

ATLAS COLLABORATION. Operation of the atlas trigger system in run 2. **Journal of Instrumentation**, v. 15, p. P10004, 2020. Acessado em: 15 fev. 2025. Disponível em: <https://doi.org/10.1088/1748-0221/15/10/P10004>.

ATLAS COLLABORATION. The environmental impact, carbon emissions and sustainability of computing in the atlas experiment. **arXiv preprint**, 2025. Acessado em: 16 ago. 2025.

BARRUÉ, R. The atlas trigger system for lhc run 3 and trigger performance in 2022. **Journal of Instrumentation**, v. 19, p. P06029, 2024. Acessado em: 15 mar. 2025. Disponível em: <https://doi.org/10.1088/1748-0221/19/06/P06029>.

BERNERS-LEE, T.; CAILLIAU, R.; LUOTONEN, A.; NIELSEN, H. F.; SECRET, A. The world wide web. **Communications of the ACM**, v. 37, n. 8, p. 76–82, 1994. Disponível em: <https://doi.org/10.1145/179606.179671>.

BLUMENHAGEN, R.; MOSTER, S.; WEIGAND, T. Heterotic gut and standard model vacua from simply connected calabi–yau manifolds. **Nuclear Physics B**, v. 751, n. 1, p. 186–221, 2006. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0550321306004780>.

BONNEFOY, R.; TEAM, L. **FENICS Shaper Circuit Documentation**. CERN, Geneva, Switzerland, 2021. Internal Note.

BRAIBANT, S.; GIACOMELLI, G.; SPURIO, M. **Particles and Fundamental Interactions**. New York: Springer-Verlag GmbH, 2011. ISBN 978-3-642-03559-6.

BRITANNICA. **Biografia de Demócrito de Abdera**. 2020. Acessado em Fevereiro de 2025. Disponível em: <http://www.britannica.com/biography/Democritus>.

BRITANNICA. **SLAC**. 2025. Acessado em Outubro de 2025. Disponível em: <https://www.britannica.com/topic/SLAC>.

BROWN, S.; VRANESIC, Z. **Fundamentals of Digital Logic with VHDL Design**. 2. ed. New York: McGraw-Hill, 2007. ISBN 9780071274478.

CDF COLLABORATION. Observation of top quark production in $\bar{p}p$ collisions. **Physical Review Letters**, v. 74, n. 14, p. 2626–2631, 1995. Disponível em: <https://doi.org/10.1103/PhysRevLett.74.2626>.

CERN. **Future Circular Collider**. 2025. Acessado em: 24/03/2025. Disponível em: <https://fcc.web.cern.ch/>.

CERN. **The Large Hadron Collider**. 2025. Página institucional. Acessado em Janeiro de 2025. Disponível em: <https://home.cern/science/accelerators/large-hadron-collider>.

CERN. **Portal oficial do CERN**. 2025. Página institucional. Acessado em Janeiro de 2025. Disponível em: <https://home.cern/>.

CERVELLÓ, M. J. The atlas tile calorimeter demonstrator data transfer and processing for phase-ii upgrade. In: **2022 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)**. [s.n.], 2022. p. 1–5. Acessado em: 15 mar. 2025. Disponível em: <https://doi.org/10.1109/NSS/MIC44805.2022.9975900>.

CHATRCHYAN, S. *et al.* The cms experiment at the cern lh. **Journal of Instrumentation**, v. 3, p. S08004, 2008.

CMS COLLABORATION. Observation of a new boson at a mass of 125 gev with the cms experiment at the lh. **Physics Letters B**, v. 716, n. 1, p. 30–61, 2012.

DYER, S. **Wiley Survey of Instrumentation and Measurement**. Wiley, 2004. (Wiley - IEEE). Acessado em: 12/07/2025. ISBN 9780471221654. Disponível em: <https://www.wiley.com/en-us/Wiley+Survey+of+Instrumentation+and+Measurement-p-9780471221654>.

DØ COLLABORATION. Observation of the top quark. **Physical Review Letters**, v. 74, n. 14, p. 2632–2637, 1995. Disponível em: <https://doi.org/10.1103/PhysRevLett.74.2632>.

EINSWEILER, K. The atlas tdaq phase-ii upgrade for the high-luminosity lh. In: **Proceedings of the Topical Workshop on Electronics for Particle Physics (TWEPP 2017)**. [s.n.], 2017. p. 1–8. Acessado em: 15 mar. 2025. Disponível em: <https://cds.cern.ch/record/2285585>.

EVANS, L. R.; BRYANT, P. Lhc machine. **Journal of Instrumentation**, v. 3, p. S08001.164, 2008.

FABJAN, C. W.; GIANOTTI, F. Calorimetry in high energy physics: Design and performance. **Reviews of Modern Physics**, v. 75, p. 1243–1286, 2003.

FILIMONOV, V.; BAUSS, B.; BÜSCHER, V.; SCHÄFER, U.; TA, D. B. Global trigger technological demonstrator for atlas phase-ii upgrade. **arXiv preprint**, 2020. Acessado em: 16 mar. 2025.

GIL-BOTELLA, I. The deep underground neutrino experiment (dune) program. **arXiv preprint arXiv:2412.14941**, 2024. Disponível em: <https://arxiv.org/abs/2412.14941>.

GOLOMB, S. W. **Shift Register Sequences**. San Francisco: Holden-Day, 1967. ISBN 9780894120485.

GONÇALO, R. A. T. C. S. Upgrade of the atlas tile calorimeter for the high luminosity lh. In: **Proceedings of Science (ICHEP2018)**. [s.n.], 2019. p. 535. Acesso em: 18 fev. 2025. Disponível em: <https://pos.sissa.it/340/535/>.

GONÇALVES, G. I. e. a. Energy reconstruction techniques in tilecal under high pile-up conditions. In: **ATL-TILECAL-Proceedings**. [s.n.], 2021. Acessado em: 15 mar. 2025. Disponível em: <https://cds.cern.ch/record/2767754/files/ATL-TILECAL-PROC-2021-002.pdf>.

GRIFFITHS, D. **Introduction to Elementary Particles**. 2. ed. Weinheim: Wiley-VCH, 2008.

GRUPEN, C.; SHWARTZ, M. **Particle Detectors**. 2. ed. Cambridge: Cambridge University Press, 2008.

HACKING, I. **The Taming of Chance**. 2nd. ed. Cambridge: Cambridge University Press, 2001. ISBN 9780521775014. Disponível em: <https://doi.org/10.1017/CBO9780511819766>.

HACKING, I. **The Emergence of Probability: A Philosophical Study of Early Ideas about Probability, Induction and Statistical Inference**. 2nd. ed. Cambridge: Cambridge University Press, 2006. ISBN 9780521685573. Disponível em: <https://doi.org/10.1017/CBO9780511817557>.

HAGIWARA, K. Higgs, susy and the standard model at colliders. **Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment**, v. 472, n. 1, p. 12–21, 2001. ISSN 0168-9002. Proceedings of the Workshop on High Energy Photon Colliders. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168900201011573>.

HALLER, J. The first-level trigger of atlas. **arXiv preprint physics/0512195**, 2005. Acessado em: 01 fev. 2025. Disponível em: <https://arxiv.org/abs/physics/0512195>.

HALZEN, F.; MARTIN, A. **Quarks and Leptons: An Introductory Course in Modern Particle Physics**. New York: John Wiley & Sons, 1984.

HARTLEY, H. John dalton, f.r.s. (1766-1844) and the atomic theory - a lecture to commemorate his bicentenary. **Proceedings of the Royal Society of London. Series B, Biological Sciences**, The Royal Society, v. 168, n. 1013, p. 335–359, 1967. ISSN 0080-4649.

HEINRICH, J. J. **The Phase-II TDAQ architecture for ATLAS: L0 Trigger latency and rate**. 2021. Slide presentation based on ATLAS-TDR-029, University of Oregon. Latency 10 s, rate ~ 1 MHz, full-granularity calorimeter input. Disponível em: https://indico.fnal.gov/event/46746/contributions/210803/attachments/141291/177861/210319_CPAD2021.pdf.

HERRERO-COLLANTES, M.; GARCIA-ESCARTIN, J. C. Quantum random number generators. **Reviews of Modern Physics**, v. 89, n. 1, p. 015004, 2017.

HYPER-KAMIOKANDE COLLABORATION. Hyper-kamiokande design report. **arXiv preprint arXiv:1805.04163**, 2018. Disponível em: <https://arxiv.org/abs/1805.04163>.

ICECUBE COLLABORATION. The icecube neutrino observatory: Instrumentation and online systems. **Journal of Instrumentation**, v. 12, p. P03012, 2017. Disponível em: <https://doi.org/10.1088/1748-0221/12/03/P03012>.

INTERNATIONAL LINEAR COLLIDER. **International Linear Collider: Technical Design Report**. 2013. Acessado em: 07/05/2025. Disponível em: <https://www.linearcollider.org/>.

JACKSON, J. D. **Classical Electrodynamics**. 3. ed. Hoboken, NJ: Wiley, 1998. ISBN 0-471-30932-X.

JAMES, F. Monte carlo theory and practice. **Reports on Progress in Physics**, v. 43, n. 9, p. 1145–1189, 1980.

JR., A. A. A. *et al.* The lhcb detector at the lh. **Journal of Instrumentation**, v. 3, p. S08005, 2008.

KAHAN, W.; GOLDBERG, D. What every computer scientist should know about floating-point arithmetic. **ACM Computing Surveys**, v. 23, n. 1, p. 5–48, 1991. Adendo: “Differences Among IEEE 754 Implementations”.

KENDALL, M. G.; SMITH, B. B. **Tables of Random Sampling Numbers**. Cambridge: Cambridge University Press, 1939.

KNUTH, D. E. **The Art of Computer Programming, Volume 2: Seminumerical Algorithms**. 3rd. ed. Reading: Addison-Wesley, 1997. ISBN 9780201896848.

KUON, I.; ROSE, J. Measuring the gap between fpgas and asics. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 26, n. 2, p. 203–215, 2008. ISSN 0278-0070.

LAWRENCE, E. O.; LIVINGSTON, M. S. The cyclotron: A new type of particle accelerator. **Physical Review**, v. 40, n. 3, p. 19–35, abr. 1 1932. Disponível em: <https://doi.org/10.1103/PhysRev.40.19>.

L’ECUYER, P. Random numbers for simulation. **Communications of the ACM**, v. 33, n. 10, p. 85–97, 1990. Disponível em: <https://dl.acm.org/doi/10.1145/84537.84555>.

LEHMER, D. H. Mathematical methods in large-scale computing units. In: **Proceedings of the 2nd Symposium on Large-Scale Digital Calculating Machinery**. Cambridge: Harvard University Press, 1951. p. 141–146. Disponível em: <https://www.ams.org/journals/mcom/1999-68-225/S0025-5718-99-00996-5/>.

LUNA, F. C.; DIAS, U. F.; LISBOA, P. H. B.; PASCHOALIN, T. C. A.; QUIRINO, T. M.; FILHO, L. d. M. A. Real-time fpga-based simulator for the tile calorimeter readout system in the atlas experiment. In: **Anais do XXVII Encontro Nacional de Modelagem Computacional (ENMC) e XV Encontro de Ciência e Tecnologia de Materiais (ECTM)**. Ilhéus, BA, Brasil: [s.n.], 2024.

MARCASTEL, F. **CERN’s Accelerator Complex: La chaîne des accélérateurs du CERN**. 2013. CERN Graphic, CDS 1621583.

MARJANOVIĆ, M. Atlas tile calorimeter calibration and monitoring systems. **arXiv preprint**, 2018. Acessado em: 15 mai. 2025.

MARSAGLIA, G. Xorshift rngs. **Journal of Statistical Software**, v. 8, n. 14, p. 1–6, 2003. Disponível em: <https://www.jstatsoft.org/v008/i14>.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, v. 8, n. 1, p. 3–30, 1998. Disponível em: <https://dl.acm.org/doi/10.1145/272991.272995>.

- MLYNARÍKOVÁ, M. *et al.* Performance of the atlas hadronic tile calorimeter. **arXiv preprint arXiv:1709.00100**, 2017. Acessado em: 02 mar. 2025. Disponível em: <https://arxiv.org/abs/1709.00100>.
- MOUCHE, P. **Overall View of the LHC**. 2014. CERN Document Server, OPEN-PHO-ACCEL-2014-001.
- NIPS. **SAPHO Project**. 2025. <https://www.nipscern.com/projects/sapho>. Acesso em: 18 mai. 2025.
- OLIVE, K. A. *et al.* Review of particle physics. **Chinese Physics C**, v. 40, n. 10, p. 100001, 2016.
- PAIS, A. **Inward Bound: Of Matter and Forces in the Physical World**. Oxford: Clarendon Press, 1986.
- PARK, S. K.; MILLER, K. W. Random number generators: good ones are hard to find. **Communications of the ACM**, v. 31, n. 10, p. 1192–1201, 1988. Disponível em: <https://dl.acm.org/doi/10.1145/63039.63042>.
- PARTICLE DATA GROUP. Review of particle physics. **Progress of Theoretical and Experimental Physics**, v. 2024, n. 8, p. 083C01, 2024. Disponível em: <https://doi.org/10.1093/ptep/ptae052>.
- PERALVA, B. S.-M. **Reconstrução de Energia para Calorímetros Finamente Segmentados**. Tese (Tese de Doutorado) — Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, MG, 2015. Acesso em: 13/05/2025. Disponível em: <https://www2.ufjf.br/ppee/producao-bibliografica/teses>.
- PEREIRA, R. A. **Estimação de energia e qualidade de dados em condições de fina segmentação e alto ruído de empilhamento**. Tese (Doutorado) — COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2019. Acessado em: 23/04/2025. Disponível em: <https://pantheon.ufrj.br/handle/11422/12290>.
- PERKINS, D. H. **Introduction to High Energy Physics**. [S.l.]: Cambridge University Press, 2000.
- POLCHINSKI, J. **String Theory. Volume 1: An Introduction to the Bosonic String**. Cambridge: Cambridge University Press, 1998. ISBN 9780521672276.
- POLCHINSKI, J. **String Theory. Volume 2: Superstring Theory and Beyond**. Cambridge: Cambridge University Press, 1998. ISBN 9780521672283.
- ROS, E. Atlas inner detector. **Nuclear Physics B Proceedings Supplements**, v. 120, p. 235–345, 2003.
- ROSS, S. M. **Introduction to Probability Models**. 11. ed. Boston: Academic Press, 2014. ISBN 978-0124079489.
- SCHINDLER, W. A stochastic model for biased physical random number generators and its applications to the evaluation of their randomness. In: **CHES 2001: Cryptographic Hardware and Embedded Systems**. [S.l.]: Springer, 2001. p. 5–23.

SESSLER, A. M.; WILSON, P. B. **Accelerator Physics**. 1. ed. Singapore: World Scientific, 2007.

STAROVOITOV, P. *et al.* Upgrade of atlas hadronic tile calorimeter for the high-luminosity lh. **Instruments**, v. 6, n. 4, p. 54, 2022. Acessado em: 10 abr. 2025. Disponível em: <https://www.mdpi.com/2410-390X/6/4/54>.

SUNAR, B.; MARTIN, W. J.; STINSON, D. R. A provably secure true random number generator with built-in tolerance to active attacks. **IEEE Transactions on Computers**, v. 56, n. 1, p. 109–119, 2007.

SUPER-KAMIOKANDE COLLABORATION. Evidence for oscillation of atmospheric neutrinos. **Physical Review Letters**, v. 81, n. 8, p. 1562–1567, 1998. Disponível em: <https://doi.org/10.1103/PhysRevLett.81.1562>.

TAJIMA, T.; DAWSON, J. M. Laser electron accelerator. **Physical Review Letters**, v. 43, n. 4, p. 267–270, 1979. Disponível em: <https://doi.org/10.1103/PhysRevLett.43.267>.

TAUSWORTHE, R. C. Random numbers generated by linear recurrence modulo two. **Mathematics of Computation**, v. 19, n. 90, p. 201–209, 1965. Disponível em: <https://www.ams.org/mcom/1965-19-090/S0025-5718-1965-0184406-1/>.

THOMSON, M. **Modern Particle Physics**. Cambridge: Cambridge University Press, 2013.

TRIMBERGER, S. M. **Field-Programmable Gate Array Technology**. New York: Springer, 2015. ISBN 978-1-4612-1201-4.

WAMBERSIE, A.; GAHBAUER, R. A. **Medical Applications of Electron Linear Accelerators**. 1996.

WIEDEMANN, H. **Particle Accelerator Physics**. 3. ed. Berlin: Springer, 2007. ISBN 3540490434.

WIKIMEDIA. **Wikimedia Commons**. 2025. Acessado em Dezembro de 2024. Disponível em: <http://commons.wikimedia>.

APÊNDICE A – PRODUÇÃO CIENTÍFICA RESULTANTE DA PESQUISA

RIBEIRO, L. G.; ANDRADE FILHO, L. M. "Simulador de Pulsos do TileCal em Tempo Real Utilizando o Processador SAPHO e FPGA". **XVII Simpósio Brasileiro de Automação Inteligente**, São João del-Rei, MG, 2025.

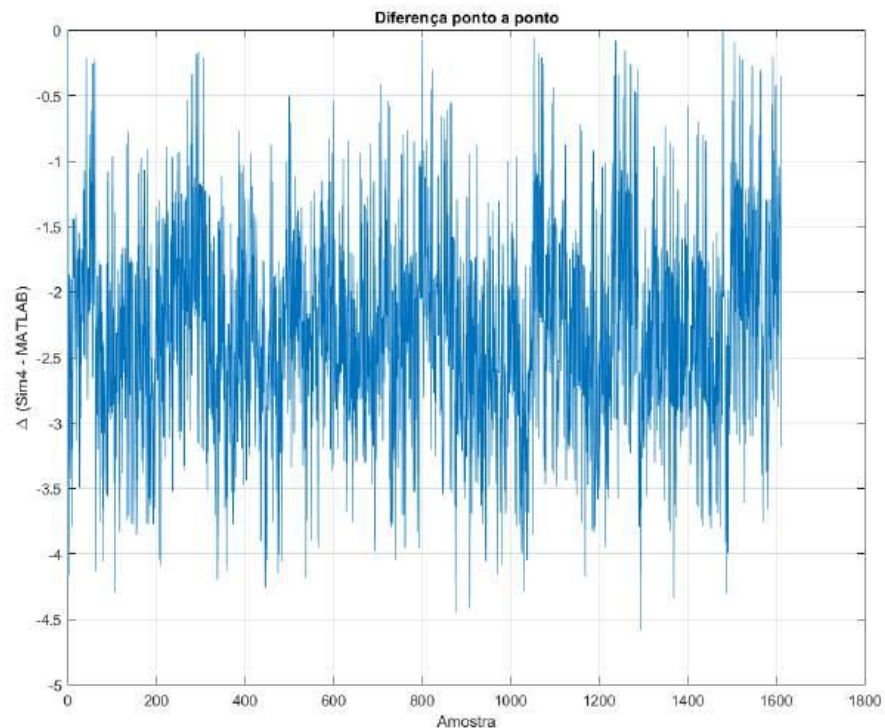
Resumo: A necessidade de simulação precisa e em tempo real de eventos físicos em experimentos de alta energia impulsiona o desenvolvimento de novas arquiteturas de processamento. Este artigo apresenta uma abordagem baseada no processador SAPHO e em Field Programmable Gate Arrays (FPGAs) para a simulação free-running de pulsos do calorímetro hadrônico de telhas do experimento ATLAS no CERN. O SAPHO compila código C e gera código Verilog (.v), que é implementado em FPGA para simular a resposta do calorímetro em tempo real. A utilização de FPGAs permite alta paralelização e baixa latência, características fundamentais para aplicações que exigem processamento rápido e determinístico. O artigo discute a arquitetura do sistema, os desafios na implementação e os benefícios do uso de hardware reconfigurável. Resultados demonstram a viabilidade da solução e sua aplicabilidade para validação de novos algoritmos de reconstrução de energia, contribuindo para a melhoria da qualidade dos dados adquiridos pelo detector.

APÊNDICE B – Tabelas e Gráficos dos Simuladores

Tabela 15 – Erro percentual para diferentes configurações de mantissa e expoente no Simulador 4 (1610 amostras, 1 μ s de simulação).

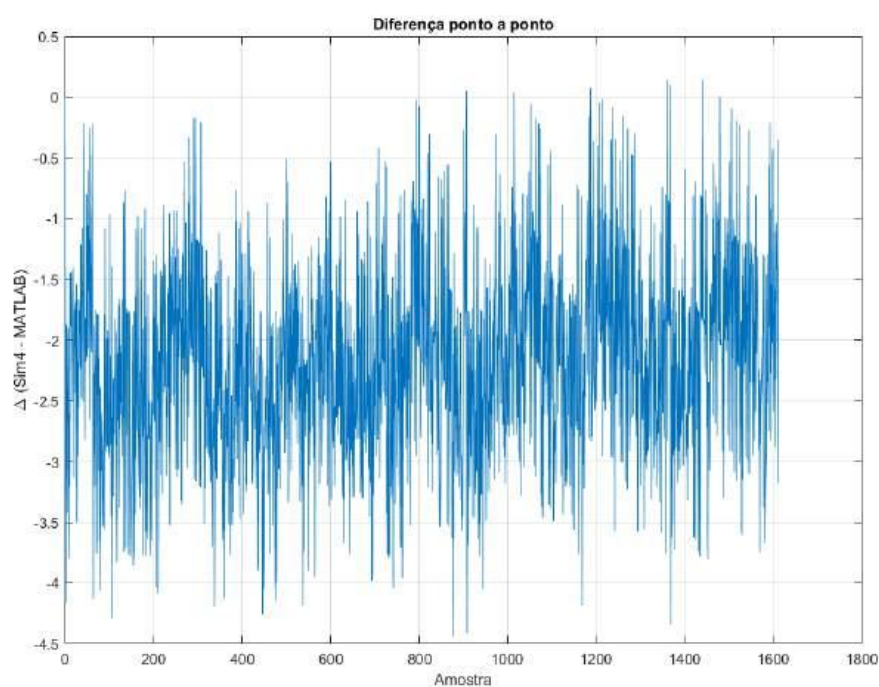
Mantissa	Expoente = 8	Expoente = 7	Expoente = 6
23	0.1589	0.2163	100.0000
22	0.1589	0.1597	100.0000
21	0.1589	0.1595	100.0000
20	0.1589	0.1595	100.0000
19	0.1589	0.1595	100.0000
18	0.1595	0.1595	100.0000
17	0.1595	0.1595	45.4891
16	0.1597	0.1597	1.7966
15	0.1587	0.3171	0.9600
14	0.1615	0.1615	0.1882
13	0.1706	1.0231	0.6783

Figura 43 – Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 23 e expoente = 8.



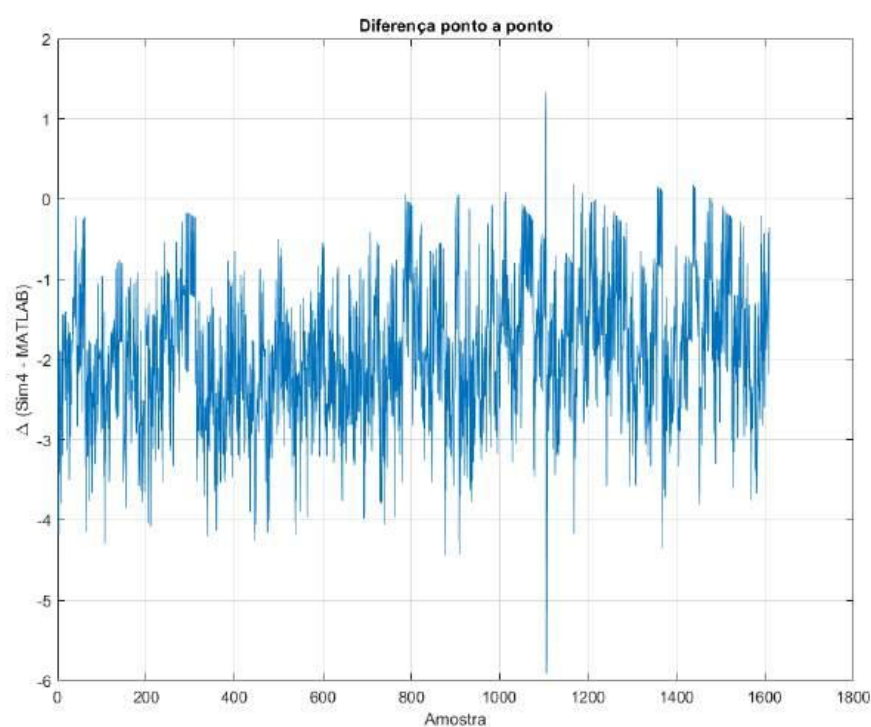
Fonte: Elaborado pelo autor (2025).

Figura 44 – Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 14 e expoente = 7.



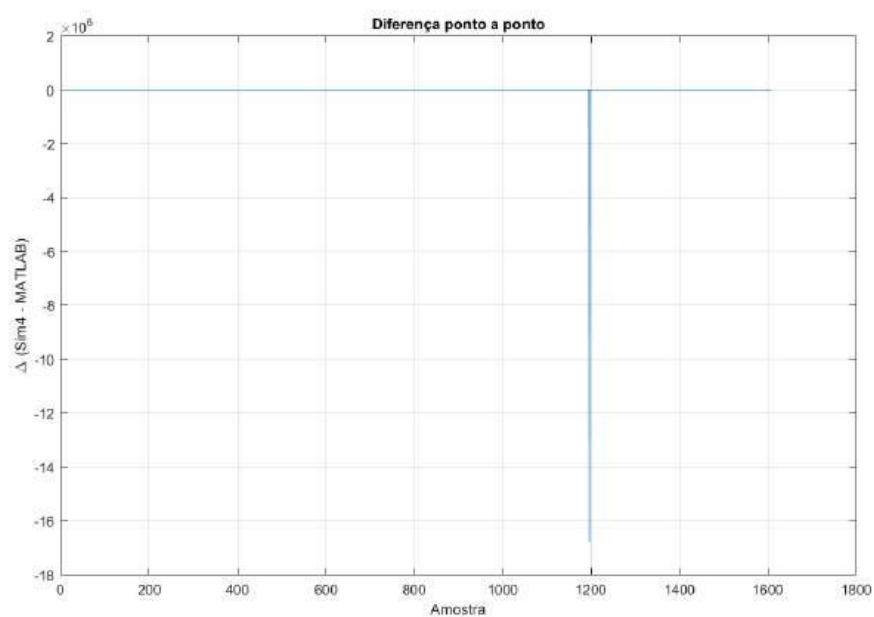
Fonte: Elaborado pelo autor (2025).

Figura 45 – Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 14 e expoente = 6.



Fonte: Elaborado pelo autor (2025).

Figura 46 – Diferença ponto a ponto entre o Simulador 4 e a Simulação em MATLAB, com mantissa = 18 e expoente = 6.



Fonte: Elaborado pelo autor (2025).