

FULANO DE TAL

**INTERFACE GRÁFICA PARA ADMINISTRAÇÃO DE
REDE LOCAL COM SOFTWARE LIVRE**

Ilha Solteira - SP
2012

FULANO DE TAL

INTERFACE GRÁFICA PARA ADMINISTRAÇÃO DE REDE LOCAL COM SOFTWARE LIVRE

Tese apresentada à Faculdade de Engenharia do Câmpus de Ilha Solteira - UNESP como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.

Especialidade: Automação.

Prof. Dr. Ciclano

Orientador

Prof. Dr. Beltrano

Co-orientador

Ilha Solteira - SP

2012

FICHA CATALOGRÁFICA

Elaborada pela Seção Técnica de Aquisição e Tratamento da Informação
Serviço Técnico de Biblioteca e Documentação da UNESP - Ilha Solteira.

S235p Santim, Máira Peres Alves.
Projeto e implementação com chaveamento de reguladores fuzzy takagi-sugeno para um conjunto de pontos de operação / Máira Peres Alves Santim. - Ilha Solteira : [s.n.], 2012
84 f.:il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de Conhecimento: Automação, 2012

Orientador: Marcelo Carvalho Minhoto Teixeira
Co-orientador: Rodrigo Cardim
Inclui bibliografia

1. Modelos fuzzy Takagi-Sugeno. 2. Desigualdades matriciais lineares (LMIs).
3. Sistemas chaveados. 4. Controlador chaveado. 5. Rastreamento.



UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE ILHA SOLTEIRA
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

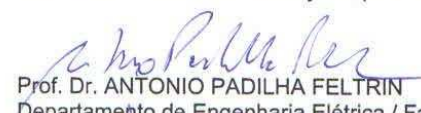
CERTIFICADO DE APROVAÇÃO

TÍTULO: Valoração de Serviços Ancilares de Geradores Distribuídos

AUTOR: AUGUSTO CÉSAR RUEDA MEDINA

ORIENTADOR: Prof. Dr. ANTONIO PADILHA FELTRIN

Aprovado como parte das exigências para obtenção do Título de DOUTOR EM ENGENHARIA ELÉTRICA, Área: AUTOMAÇÃO, pela Comissão Examinadora:


Prof. Dr. ANTONIO PADILHA FELTRIN

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. ANTONIO MARCOS COSSI

Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. CARLOS ROBERTO MINUSSI

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira


Prof. Dr. WALMIR DE FREITAS FILHO

Departamento de Sistemas de Energia Elétrica / Universidade Estadual de Campinas


Prof. Dr. JOSÉ ANTONIO DOMÍNGUEZ NAVARRO

Departamento de Engenharia Elétrica / Universidad de Zaragoza

Data da realização: 24 de fevereiro de 2012.

À minha família, em especial aos meus pais Francisco e Marilena, aos meus irmãos Rafael e Gisele e ao meu marido Ricardo, por todo amor, apoio, confiança e incentivo em todos os momentos.

AGRADECIMENTOS

Meus agradecimentos a todos os familiares, amigos, professores e funcionários da FEIS-UNESP, que direta ou indiretamente contribuíram para a realização deste trabalho. Em especial, dedico meus agradecimentos:

- A Deus, por ter me dado força e saúde para chegar até aqui;
- Aos meus pais Maria e João e aos meus irmãos Pedro e Paulo pelo carinho, apoio e incentivo;
- Ao meu marido Ricardo pelo amor, apoio, confiança e incentivo em todos os momentos;
- Ao Prof. Dr. Fulano de Tal, por todo ensinamento, incentivo, confiança e orientação;
- Ao Prof. Dr. Ciclano de Tal, pelo acompanhamento nas bancas examinadoras, sugestões e incentivo;
- Ao Dr. Beltrano pela co-orientação e todo o ensinamento.
- Aos meus amigos e colegas do laboratório que de forma direta ou indiretamente me ajudaram, em especial ao Chico, pela ajuda e o trabalho feito em conjunto ;
- Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela oportunidade e apoio financeiro.

*“O sol é para todos,”
mas a sombra é para quem.
chega primeiro.*

Geremias Ludu

RESUMO

Neste trabalho foi desenvolvido uma interface gráfica para uso via web, utilizando-se as linguagens shell-script e PHP, com o objetivo de facilitar a configuração e monitoração de diferentes serviços necessários em um servidor de rede, tais como: firewall, DHCP, squid/proxy, DNS, e-mail, dentre outros. Para isso, utilizou-se uma estratégia de desenvolvimento modular, para facilidade de uso e que permite a inclusão de novos módulos posteriormente. A ferramenta foi totalmente desenvolvida com software livre e o acesso ao seu código permite alterações de acordo com as necessidades do usuário.

Palavras-chave: Servidores. Redes. Firewall. Segurança.

ABSTRACT

In this work, was developed a graphical user interface for use by the web, using PHP and shell-script languages, in order to facilitate the configuration and monitoring of different services required on a network server, such as firewall, DHCP, squid/proxy, DNS, e-mail, among others, was developed in this paper. For this, was used a strategy for developing modular for easy of use and allows the addition of new modules later. The tool was developed entirely with free software and allows access to your code changes according to user needs.

Keywords: Servers. Networks. Firewall. Security.

LISTA DE FIGURAS

Figura 1	Esquema do grande colisor de Hádrons (LHC).	17
Figura 2	Esquerda - Máquina do LHC. Direita - Sub-detectores do CMS	18
Figura 3	Descrição do sistema, fluxo de dados e especificação de latência por meio dos dispositivos eletrônicos fora do detector que formam a decisão do L1TT. . .	19
Figura 4	Esquema do Sistema de comunicação Serial LpGBT.	20
Figura 5	Arquitetura do LpGBT ASIC.	22
Figura 6	Esquema de um registrador de deslocamento (<i>shift register</i>).	31
Figura 7	Tipos de sistemas <i>Linear Feedback Shift Registers</i>	32
Figura 8	Tabela de polinômios com máximo comprimento para circuitos LFSR (<i>shift register</i>).	33
Figura 9	Esquema de um registrador de deslocamento (<i>shift register</i>).	34
Figura 10	Esquema de um sistema (<i>Fibonacci Linear Feedback Shift Register</i>).	35
Figura 11	Esquema de um sistema (<i>Galois Linear Feedback Shift Register</i>).	36
Figura 12	Esquema simplificado de um sistema (<i>Galois Linear Feedback Shift Register</i>).	36
Figura 13	Esquema de um <i>Scramblers</i> Aditivo.	39
Figura 14	Esquema de um <i>Scramblers</i> Multiplicativo.	40
Figura 15	Diagrama de bloco Típico de links de alta velocidade.	42
Figura 16	Esquemático de um PLL básico.	44
Figura 17	Esquemático de um CDR/PLL básico.	46
Figura 18	A relação entre diferentes tipos de FEC.	49
Figura 19	Diagrama do código Trellis para o codificador	51
Figura 20	Bloco de dados deteriorado por 25 bits de <i>noise burst</i>	53
Figura 21	Diagrama em Blocos do Decodificador FEC Reed-Solomon	55
Figura 22	Exemplo de Entrelaçamento de Bloco.	62

Figura 23	Implementação de um Entrelaçador Convolutacional com <i>Shift Register</i> . . .	64
Figura 24	Computação de um Entrelaçador Convolutacional	65
Figura 25	Ilustração é um exemplo de figura	71
Figura 26	Novo sistema operacional.	72

LISTA DE TABELAS

Tabela 1	Resultado para o sistema	24
Tabela 2	Mapeamentos dos Elementos Finitos em elementos básicos do Campo Finito $GF(2^3)$ com $p(x) = 1 + X + X^3$	28
Tabela 3	Espaço de busca combinatório reduzido (<i>EBCR</i>) de 10, 5, 3 e 2 soluções com <i>gap</i> de 5% Para IEEE	72

LISTA DE ABREVIACES E SIGLAS

ALICE	A Large Ion Collider Experiment
ASIC	Application-Specific Integrated Circuit
ATLAS	A Toroidal LHC ApparatuS
BCH	Bose-Chaudhuri Hocquen-hem
BM	Berleykamp Massey
CMS	Solenóide de Muon Compacto
CDR/PLL	Recuperação de Dados, <i>Clock</i> e <i>Loop</i> de Fase Bloqueada
CDR	Clock Data Recovery
CF	Campos Finitos
CRC	Cyclic Redundancy Check
DA	Driver Amplifier
DAQ	Data Acquisition
DEC	Decoder
DeSER	Deserializer
DSCR	Descrambler
DTC	Data, Trigger and Control
ECAL	Eletromagnetic Calorimeter
eLinks	Links Elétricos de Saída/Entrada
ePorts	Electrical Ports
ePortsRx	ePorts do Receptor
ePortsTX	ePorts do transmissor
FPGA	Field Programmable Gate Array
FEC	Forward Error Corretion
HCAL	Muon Chamber, Hadronic Calorimeter
HL-LHC	High Luminosity - Large Hadron Collider
ISI	Interferência Inter-Simbólica
L1TT	Level 1 Track and Trigger
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LFSR	Linear Feedback Shift Register
TX	Transmissor
NRZ	non-return-to-zero
PA	Phase-Aligner
PLL	Phase-Locked Loop
PRBS	Sequência Binária Pseudoaleatória
pT	Momento Transversal
PU	Pileup
RX	Receptor
R-S	Reed Solomon
SC	Slow Control
SCR	Scrambler
SER	Serializer

LISTA DE SÍMBOLOS

T	Tesla
GeV	Giga Eletrovolt
μs	Micro Segundo
θ_i	Ângulo de fase na barra i
g_{ij}	Condutância da linha no ramo ij
Y	Conjunto das linhas que podem ou não serem adicionadas no ramo ij
Ω_b	Conjunto de barras
Ω_l^1	Conjunto de caminhos nos quais existem Linhas na configuração base
Ω_l^2	Conjunto de caminhos novos (onde serão adicionadas novas Linhas)
Ω_l^0	Conjunto de linhas existentes na configuração base
Ω_l	Conjunto de ramos
c_{ij}^n	Custo de construção das linhas no ramo ij
d_i	Demanda na barra i
ε_f	Error da condição de factibilidade
ε_o	Error da condição de otimalidade
ε_μ	Error do parâmetro de barreira
γ	Fator de segurança
\bar{f}_{ij}^0	Fluxo de potência ativa máximo nos ramos para o conjunto de linhas já existentes
\bar{f}_{ij}^1	Fluxo de potência ativa máximo nos ramos para o conjunto de linhas já existentes ou linhas adicionadas em paralelo
\bar{f}_{ij}^2	Fluxo de potência ativa máximo nos ramos para o conjunto de linhas correspondentes aos novos caminhos
\bar{f}_{ij}	Fluxo de potência ativa máximo permitida no ramo ij para linhas novas
f_{ij}^0	Fluxo de potência ativa nos ramos para o conjunto de linhas já existentes
f_{ij}^1	Fluxo de potência ativa nos ramos para o conjunto de linhas já existentes ou linhas adicionadas em paralelo
f_{ij}^2	Fluxo de potência ativa nos ramos do conjunto de linhas correspondentes aos novos caminhos
f_{ij}	Fluxo de potência ativa no ramo ij para linhas novas
$f_{ij,y}$	Fluxo na linha y do ramo ij
p_i	Geração na barra i
\bar{p}_i	Geração máxima na barra i
v	Investimento devido às adições de Linhas no sistema - Função Objetivo
ij	Linha entre as barras i e j
n_{ij}	Número de linhas adicionadas no ramo ij

\bar{n}_{ij}^2	Número máximo de linhas em caminhos novos
\bar{n}_{ij}^1	Número máximo de linhas que podem ser adicionadas em paralelo às linhas dos caminhos já existentes
\bar{n}_{ij}	Número máximo de Linhas que podem ser adicionados no ramo ij
n_{ij}^1	Número de linhas adicionadas em paralelo às linhas já existentes
n_{ij}^0	Número de linhas existentes na configuração base no ramo ij
n_{ij}^2	Número de linhas novas adicionadas no ramo ij
γ_{ij}	Susceptância nas linhas do ramo ij
γ_{ij}^0	Susceptância nas linhas existente do ramo ij
$w_{ij,y}$	Variável binária correspondente à linha y candidata a ser adicionada ou não no ramo ij
x_{ij}	reatância do circuito ij
q_i	vetor de geração de potência reativa na barra i
\bar{q}_i	limite máximo de geração de potência reativa na barra i
\underline{q}_i	limite mínimo de geração de potência reativa na barra i
e_i	vetor de demanda de potência reativa na barra i
V_i	magnitude de tensão na barra i
\bar{V}_i	limite máximo da magnitude de tensão na barra i
\underline{V}_i	limite mínimo da magnitude de tensão na barra i
e_i	vetor de demanda de potência reativa na barra i
s_{ij}^{de}	fluxo de potência aparente (MVA) no ramo ij saindo do terminal
s_{ij}^{para}	fluxo de potência aparente (MVA) no ramo ij chegando no terminal
\bar{s}_{ij}	limite de fluxo de potência aparente (MVA) no ramo ij
θ_{ij}	diferença angular entre as barra i e j
Ω_{bi}	conjunto das barras vizinhas da barra i
g_{ij}	condutância da linha no ramo ij
g_{ij}^0	condutância existente da linha no ramo ij
b_{ij}	susceptância da linha no ramo ij
b_{ij}^{sh}	susceptância shunt da linha no ramo ij
b_i^{sh}	susceptância shunt na barra i
G_{ij}	matriz de condutância
B_{ij}	matriz de susceptância

SUMÁRIO

1	INTRODUÇÃO	16
1.1	Motivação	16
1.2	Descrição do Sistema LpGBT ASIC	21
1.3	TRABALHO DESENVOLVIDO	23
2	CAMPOS FINITOS (FINITE FIELDS).	25
2.1	Propriedades do Campo de Galois	25
2.2	Polinômios Primitivos	26
2.3	Construção dos Campos de Galois Estendidos	26
2.4	Operações nos Campos de Galois	27
2.4.1	Adição e Subtração	28
2.4.2	Multiplificação e Divisão	29
3	LINEAR FEEDBACK SHIFT REGISTERS	31
3.1	Fibonacci Linear Feedback Shift Registers	34
3.2	Galois Linear Feedback Shift Registers	35
4	SCRAMBLERS	38
4.1	<i>Scramblers</i> Aditivos (Synchronous)	38
4.2	<i>Scramblers</i> Multiplicativos (Self-Synchronizing)	39
5	Descrição do Link Serial de Alta Velocidade	42
5.1	Serializador	43
5.2	Driver Amplificador	43
5.3	Phase-Locked Loop (PLL)	43
5.4	Equalizador	45
5.5	Clock and Data Recovery (CDR)	45

5.6	Desserializador	46
5.7	Esquemas de Codificação	47
6	FORWARD ERROR CORRECTION (FEC)	48
6.1	Tipos de FECs	48
6.1.1	Block Codes	49
6.1.1.1	<i>Cyclic Codes</i>	49
6.1.1.2	<i>Hamming Codes</i>	50
6.1.1.3	<i>Bose-Chaudhuri Hocquen-hem (BCH) Codes</i>	50
6.1.2	Convolutional Codes	51
6.2	Propriedades dos códigos FEC	51
6.3	Código Reed-Solomon	52
6.3.1	Performance de códigos R-S contra Burst Noise	53
6.3.2	Codificador Reed-Solomon	53
6.3.3	Decodificador Reed-Solomon	54
6.3.3.1	<i>Computação da Síndrome</i>	55
6.3.3.2	<i>Algoritmo Berlekamp-Massey</i>	57
6.3.3.3	<i>Algoritmo Chien Search</i>	59
6.3.3.4	<i>Algoritmo de Forney</i>	60
7	Entrelaçador (<i>Interleaving</i>)	61
7.1	Entrelaçador de Bloco (<i>Block Interleaving</i>)	61
7.2	Entrelaçador Convolutacional (<i>Convolutional Encoder</i>)	63
8	RESULTADOS E DISCUSSÕES	66
9	CONCLUSÕES	67
	REFERÊNCIAS	68

APÊNDICE A - LINUX	71
APÊNDICE A.1 - HISTÓRICO DO LINUX	71
APÊNDICE B - AINDA FALANDO DO LINUX	72
APÊNDICE B.1 - MELHORIAS PARA O LINUX EM UM AMBIENTE CO- ORPORATIVOS DE DUAS GRNDES FRNTES INTERPRETATIVAS	72
ÍNDICE REMISSIVO	75

1 INTRODUÇÃO

Em qualquer sistema de comunicação deve-se garantir uma alta confiabilidade dos dados transmitidos, de forma que no lado do receptor sejam recebidos os mesmos dados enviado pelo transmissor. A forma de onda do sinal transmitido é afetado por dois mecanismos básicos: todas as linhas de transmissão e circuitos possuem função de transferência não lineares no domínio da frequência e ruídos elétricos não desejáveis ou algum tipo de interferência que distorce o sinal enviado (SKLAR, 1988a, p. 3). Em comunicações digitais e seriais de alta velocidade presentes em um ambiente com alto nível de radiação eletromagnética, o uso de transmissões por fibra óptica são mais adequadas para atingir o nível de performance exigido. Entretanto, transmissões em fibras ópticas estão sujeitas a perdas devido aos efeitos ópticos lineares e não-lineares. Os tipos de perda são: atenuação, dispersão, auto-modulação de fase, mistura de quatro ondas, espalhamento Brillouin Estimulado e espalhamento Raman estimulado (GHATAK; THYAGARAJAN, 1998).

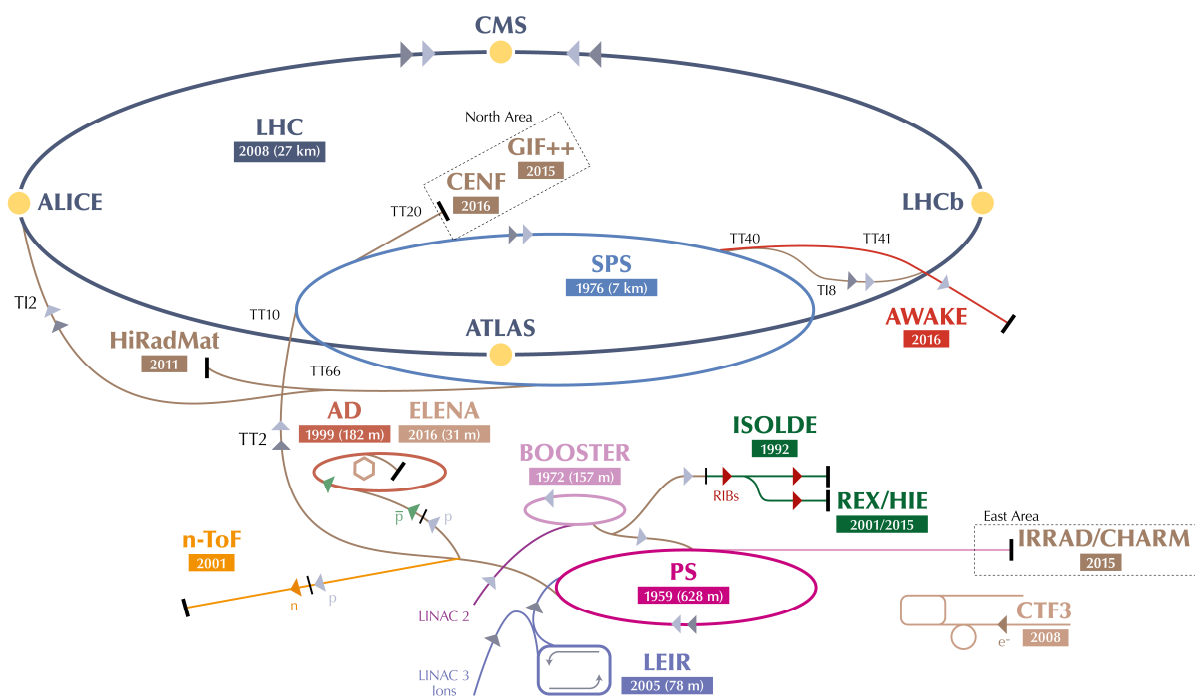
Desta forma, em transmissões seriais de alta velocidade é empregado protocolos de comunicação para garantir a maior eficiência e o menor número de erros possível. Este tipo de implementação é extremamente útil em sistemas para física de altas energias. Nestes sistemas, há a presença de uma alta taxa de transmissão combinado com uma alta radiação eletromagnética e um interesse de uma alta confiabilidade no canal, implicando na necessidade de implementação de protocolos de comunicação. Estes protocolos possibilitam a introdução de sistemas para identificar e corrigir possíveis erros, além de possibilitar que circuitos externos sincronizem os dispositivos comunicantes.

1.1 Motivação

Este trabalho é parte de uma colaboração com o laboratório *São Paulo Research and Analysis Center* (SPRACE) (SPRACE, 2019). O laboratório SPRACE possui vários ramos de pesquisa, sendo uma delas a instrumentação eletrônica para os sistemas do LHC. Este possui uma extensão de 27 km de circunferência, localizado na fronteira Franco-Suíça, tendo por objetivo descobrir a origem da massa das partículas elementares e outras dimensões do espaço (WIKIPÉDIA, 2019). O colisor é o maior equipamento já construído para pesquisa em física de altas energias do mundo, obtendo resultados expressivos como a descoberta do Bóson de Higgs. Este Bóson é uma partícula elementar prevista pelo modelo padrão de partículas, que ajuda a explicar a massa de outras partículas elementares (RANDALL, 2013).

No percurso do colisor há 4 detectores : A *Toroidal LHC ApparatuS* (ATLAS), *Compact Muon Solenoid* (CMS), A *Large Ion Collider* (Alice) e o *LHC beauty* LHCb. O acelerador de partículas fornece velocidade e os detectores captam os produtos do impacto das partículas. Dessa forma, pode-se observar a existência de traços de partículas elementares que explicam teorias importantes sobre a física de altas energias (FERREIRA, 2009). Na Figura 1 é ilustrado um esquema do grande colisor de Hádrons que está localizado a 175 metros abaixo do solo.

Figura 1 - Esquema do grande colisor de Hádrons (LHC).



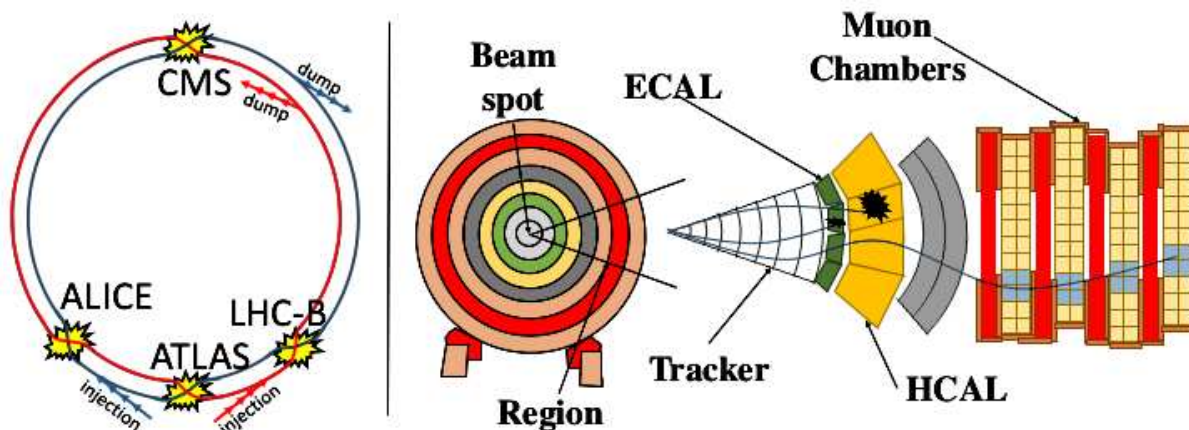
Fonte: Adaptado de Mobs (2016)

O grande colisor está em um processo de pesquisa para realizar uma grande atualização, a chamada Grande Luminosidade ou da sigla em inglês *High Luminosity - Large Hadron Collider* (HL-LHC). Com isto é esperado um aumento no número médio de colisões próton-próton de 100 para em torno de 140 a 200, denominadas *pileup* (PU). Estas colisões médias são obtidas pelo cruzamento de um grupo de prótons em uma frequência de 40MHz. Consequentemente, um grupo de prótons se cruza no ponto de colisão dentro do detector, para a cada 25ns (AG- GLETON et al., 2017, p. 2).

Em cada um dos 4 detectores há um ponto de colisão sendo possível por monitorar os eventos decorrentes do choque d. Especificamente, o projeto descrito é idealizado para atuar no detector de propósito geral CMS. Este é um grande detector com o objetivo de investigar uma vasta variedade de fenômenos físicos. O detector possui uma solenóide supercondutora central de 6 metros de diâmetro, provendo um campo magnético de 3,8T. Na figura 2 é ilustrado um

esquemático do detector CMS (AGGLETON et al., 2017, p. 2).

Figura 2 - Esquerda - Máquina do LHC. Direita - Sub-detectores do CMS



Fonte: Adaptado de ??, p. 19)

Pela figura 2, há vários componentes no colisor para a medição dos fenômenos ou elementos gerados pela colisão das partículas. Para o aumento na taxa de colisão, a instrumentação eletrônica presente no detector deverá ser atualizada para suportar o novo volume de dados gerados. Os sistemas desenvolvidos devem ser capazes de transmitir e processar um grande volume de dados em uma faixa muito curta de tempo (BRÜNING, 2019).

O trabalho desenvolvido pelo laboratório SPRACE está diretamente ligado ao detector Solenóide de Muon Compacto (CMS) do LHC. Os detectores do LHC tem estruturas diferentes e cada um obtém dados de partículas específicas. A junção de todos os dados de todos os detectores forma uma imagem completa do experimento, ajudando a realizar novas descobertas.

O detector CMS foi designado para registrar, direta ou indiretamente, o caminho e a energia de todos os modelos padrões de partículas resultantes da colisão (??). O detector é composto de várias camadas de sub-detectores especializados para medir as diferentes características das partículas produzidas, como ilustrado no lado direito da Figura 2. A instrumentação do CMS inclui diferentes componentes, dentre eles: *Muon Chamber*, *Hadronic Calorimeter* (HCAL), *Eletromagnetic Calorimeter* (ECAL) e o *Silicon Tracker*. Dentro do detector há um solenóide supercondutor aplicando um alto campo magnético que é capaz de curvar a trajetória da partícula após a colisão. Portanto, quanto maior a energia da partícula menor a curvatura na sua trajetória (AGGLETON et al., 2017, p. 3).

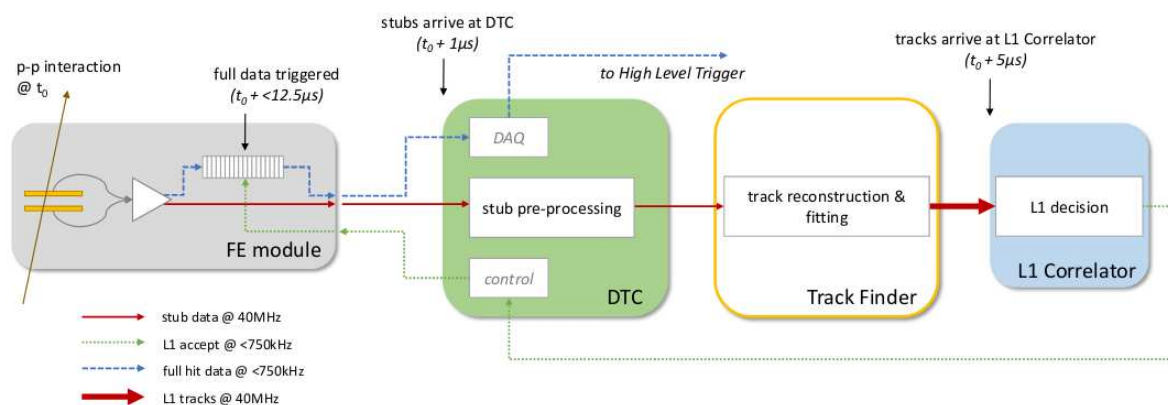
Pela análise do caminho e da energia da partícula, medida pelo *Tracker*, *muon chamber* e os calorímetros é possível reconstruir os eventos resultantes da colisão. Nos dispositivos de medição há uma pré-seleção para partículas com um limiar de momento transversal (pT) das partículas carregadas tipicamente maior do que 2 ou 3 Giga Eletrovolt (GeV). Entretanto, devido

ao grande número de colisões os dados devem ser filtrados por questões de armazenamento e capacidade de transmissão. Assim, os dados passam por um filtro denominado *Level 1 Track and Trigger* (L1TT). Este sistema faz a reconstrução dos caminhos das partículas e seleciona os eventos que são interessantes para estudo.(AGGLETON et al., 2017).

A colaboração entre o laboratório SPRACE com o CMS objetiva-se colaborar no desenvolvimento de sistemas na área da instrumentação eletrônica, o qual serão implementados dentro de um sistema completo de detecção. O novo sistema que está em desenvolvimento, principalmente objetiva-se eliminar as restrições de latência que o atual possui (COLLABORATION, 2019).

Para este propósito, foi desenvolvido um sistema chamado *Low Power GigaBit Transceiver* (LpGBT) para transmitir dados coletados nos dispositivos eletrônicos do colisor pelos *Front End module* (FE module), para um sistema responsável pela análise e seleção das amostras coletadas localizado fora do detector denominado *Data, Trigger and Control* (DTC). Na Figura 3 é descrito um esquema do sistema presente no colisor para a decisão dos eventos a serem armazenados, chamado L1TT.

Figura 3 - Descrição do sistema, fluxo de dados e especificação de latência por meio dos dispositivos eletrônicos fora do detector que formam a decisão do L1TT.



Fonte: Aggleton et al. (2017)

No *FE module* captura-se os dados da colisão das partículas. Esses dados são transmitidos para um sistema, fora do colisor, o *Data, Trigger and Control* (DTC). Este sistema, acompanhado do *Track Finder* e o *L1 Correlator* gera a decisão de armazenar o dado coletado no colisor. O tempo total para a decisão de quais eventos medidos nos sensores do detector é de $12.5\mu s$ (AGGLETON et al., 2017, p. 4).

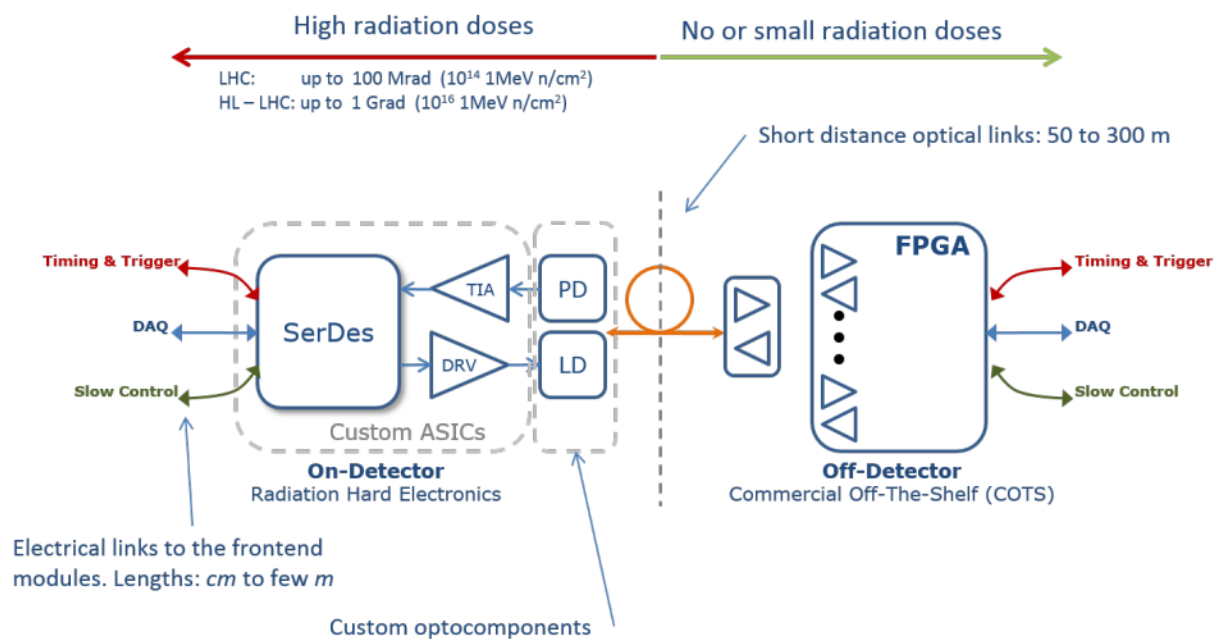
O *L1 correlator* necessita de $3.5\mu s$ para correlacionar os caminhos com dados primitivos do calorímetro e o sistema Muon e tomar a decisão se o evento é de interesse do estudo. A

propagação da decisão do *L1 correlator* para os buffers do *FE module* leva mais $1\mu s$, enquanto outros $3\mu s$ são necessários como margem de segurança. Isso significa que para as camadas sejam utilizadas pelo *L1 correlator* para tomar a decisão, os eventos devem ser extraídos dos componentes eletrônicos do rastreador, organizados e feita a reconstrução das trilhas no *Track Finder* em aproximadamente $5\mu s$ após a colisão. Em torno de $1\mu s$ é necessário para geração, empacotamento e transmissão dos eventos do *FE module* para o DTC. Portanto, a latência de processamento para reconstruir os rastros a partir dos dados chegando ao DTC é fixado em $4\mu s$ (AGGLETON et al., 2017, p. 5).

Portanto, com o L1TT é possível selecionar os eventos físicos de interesse e reduzir a taxa de dados do detector. A redução é de 40MHz para um número menor do que 750kHz, sendo esta a máxima taxa que o sistema suporta. O transporte dos dados entre *FE module* e o DTC, feito pelo *LpGBT ASIC* e um FPGA com o protocolo LpGBT implementado, com uma latência tolerada de $1\mu s$ (AGGLETON et al., 2017, p. 4).

No sistema L1TT há 3 sinais lógicos: *Timing and Trigger Control* (TTC), *Data Acquisition* (DAQ) e *Slow Control* (SC). Estes sinais não necessitam possuir caminhos físicos diferentes, sendo possível transmiti-los em canal óptico através do LpGBT. Na figura 4 é ilustrado sistema usando o protocolo LpGBT em que pode-se observar a transmissão dos sinais lógicos. As cores dos sinais lógicos da Figura 4 representam as mesmas cores dos sinais lógicos da Figura 3.

Figura 4 - Esquema do Sistema de comunicação Serial LpGBT.



Fonte: Team (2019, p. 3)

O sistema FPGA com a implementação do LpGBT, o qual está fora do detector, já está

descrito e implementado. Entretanto, o sistema LpGBT ASIC que fica alocado dentro do Detector, em uma região de alta radiação eletromagnética, ainda está em fase de desenvolvimento. Todavia, o sistema LpGBT dentro do detector está descrito em códigos de descrição de hardware, porém não são sintetizáveis em um FPGA. A grande motivação do trabalho é descrever o sistema LpGBT ASIC de uma forma que seja sintetizável em um FPGA.

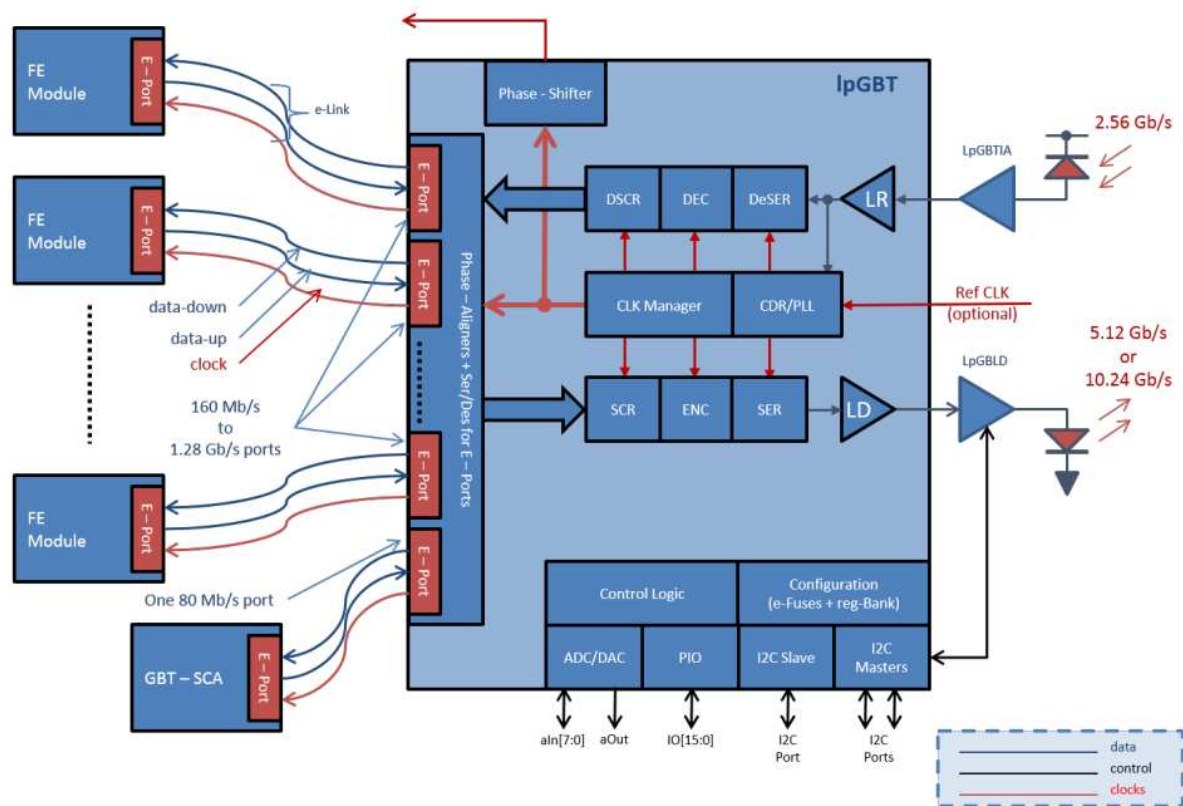
1.2 Descrição do Sistema LpGBT ASIC

O lpGBT ASIC faz parte do chipset lpGBT composto pelos seguintes chips: um amplificador de impedância trans para o receptor óptico (GBTIA/lpGBTIA-em desenvolvimento), um driver de diodo a laser quádruplo e o próprio lpGBT propriamente dito, um link ASIC que implementa todas as funções necessárias do transceptor de dados e tempo, além de um conjunto de funções necessárias para o controle do experimento.

Como descrito na Figura 4, cada *FE module* será atendido por um LpGBT ASIC, ao qual por um par de fibras ópticas, faz interface diretamente com o sistema DTC. Esses links ópticos poderão transferir dados do detector a 5.12 ou 10.24 Gb/s, fornecendo uma largura de banda eficaz entre 3.84 e 8.96 Gb/s, representando correção de erros e sobrecarga de protocolo. A comunicação entre os *FE modules* e cada chip LpGBT suporta até 7 links de entrada-saída a 1.28 Gb/s e 8 Links de controle a 160Mb/s (ORFANELLI, 2019).

Aproximadamente 75% dessa largura de banda será dedicada à leitura de dados dos eventos no detector a cada 25 ns. Os aproximadamente 25% restantes da largura de banda da leitura do módulo serão dedicados à transmissão dos dados completos do evento, incluindo todas as faixas/pixels de acerto, acionados por um sinal de aceitação do *L1 Correlator* (AGGLETON et al., 2017). A arquitetura geral do lpGBT ASIC e suas principais conexões externas são exibidas na Figura . Em seu genérico configuração, o lpGBT se conecta a um driver de laser ASIC e a um amplificador de trans-impedância ASIC. Na figura é descrito a arquitetura do sistema LpGBT.

Figura 5 - Arquitetura do LpGBT ASIC.



Fonte: Team (2019, p. 5)

Pela Figura 5, na parte do *downlink* há vários componentes essenciais para o sistema. Primeiramente, há o circuito de recuperação de dados, *clock* e *loop* de fase bloqueada (CDR/PLL) recebe dados seriais de alta velocidade (2.56 Gb/s). A partir dele, recupera e gera um relógio de alta velocidade apropriado para amostrar corretamente a entrada fluxo de dados. Os dados seriais são desserializados pelo *Deserializer* (DeSER), ou seja, convertidos para paralelo e, em seguida, decodificado pelo *decoder* (DEC), com as correções de erro apropriadas, e finalmente realiza-se o processo de desembaralhar o dado pelo *Descrambler* (DSCR).

Na parte do *uplink*, os dados a serem transmitidos são embaralhados pelo *scrambler* (SCR), e então codificado com um código de correção de erro denominado *forward Error Corretion* (FEC) antes de ser serializado pelo *serializer* (SER) e enviado ao driver do laser. A configuração do driver do laser pode ser realizada através de uma conexão I2C do LpGBT.

O *Clk Manager* se encarrega de gerar e gerenciar os diferentes relógios de alta e baixa frequência necessários nas diferentes partes do ASIC. Um deslocador de fase programável está disponível para gerar 4 relógios de usuário externo com frequência programável (40 MHz a 1,28 GHz) e fase (rotação total de 360 graus em incrementos de fase de 50 ps). No modo Transceptor, um relógio externo deve, ou, pode ser usado para operação durante a inicialização

como referência relógio para o SER ou como uma ajuda de travamento para o circuito CDR.

A lógica geral de controle e monitoramento cuida do controle das diferentes partes do chip de acordo com a operação modo selecionado e as informações de configuração do ASIC. As informações de configuração inicial são obtidas do chip on Fusíveis eletrônicos que podem ser modificados através do próprio link óptico ou de uma interface escrava I2C.

As conexões com os módulos *FE module* e o LpGBT ASIC são feitas através de conjuntos de links elétricos de saída/entrada (eLinks). Dependendo da taxa de dados e da mídia de transmissão usada, os eLinks permitem conexões que podem se estender até alguns metros. Os eLinks são direcionados por uma série de ePorts no LpGBT e estão associados às portas eLink nos módulos *FE module*. O número de eLinks ativos e sua taxa de dados são programáveis.

Os ePorts de recebimento (ePortRx) são associados ao *uplink* desserializam os dados recebidos dos módulos do *FE module* ou ASICs para que possam ser embaralhados, codificados e montados no *uplink* antes de serem transmitidos ao DTC. Cada ePortRx é associado a um *Phase-Aligner* (PA) usado para garantir que os dados seriais recebidos dos *FE module* são amostrados adequadamente no meio do diagrama de olho. Por outro lado, os ePorts do transmissor (ePortTx) são associado ao *downlink* e o SER, recebidos do DTC, e transmitem dados aos dispositivos *FE module* usando o eLinks. Por fim, os ePorts também têm eClocks associados. Esses relógios são programáveis em frequência, mas possuem fase fixa.

1.3 TRABALHO DESENVOLVIDO

No ambiente do detector há uma alta taxa de radiação eletromagnética, por conta da alta velocidade dos átomos presentes no tubo. Dessa forma, em transmissões de alta velocidade de uma placa para outra podem acarretar a presença de ruídos no canal de transmissão. Os ruídos presentes no canal de transmissão danificam os dados originais, acarretando no armazenamento de dados incoerentes para um posterior estudo. Portanto, a implementação de um protocolo de comunicação possibilita a detecção e correção de erros e a solução de problemas envolvidos na transmissão em altas velocidades.

Este trabalho desenvolve o estudo do protocolo de comunicação LpGBT e suas características por meio de uma descrição em VHDL e FPGA. Dessa forma, o sistema desenvolvido pode ser usado como codificação do canal de transmissão entre duas placas FPGA garantindo uma maior confiabilidade dos dados transmitidos.

O trabalho confeccionado foi dividido em

A tabela abaixo é uma tabela de exemplo de ...

Tabela 1 - Resultado para o sistema

	Estágio 1	Estágio 2	Estágio 3
Número de linhas n_{ij}	$n_{6-10} = 1$ $n_{7-8} = 2$ $n_{10-12} = 1$ $n_{11-13} = 1$	$n_{20-23} = 1$	$n_{1-5} = 1$ $n_{3-24} = 1$
Função Objetivo $v = 220.2860$			

Fonte: Dados da pesquisa do autor.

2 CAMPOS FINITOS (FINITE FIELDS).

Uma noção básica de um campo finito pode ser explicitada como um conjunto que possui um número finito de elementos, definindo-se adição e multiplicação como operações básicas existentes dentro deste conjunto. Desta forma, este conjunto possui as seguintes propriedades para qualquer conjunto F :

- Para qualquer elemento a, b dentro de um conjunto F , as operações de adição e multiplicação são operações binárias no conjunto F .
- Para qualquer elemento dentro do conjunto F as propriedades associativas e distributivas devem ser mantidas.
- Para qualquer a, b em F , as seguintes relações são mantidas: $a + b = b + a$ e $a \cdot b = b \cdot a$.
- No conjunto F o elemento identidade aditivo ($a + 0 = a$) e o multiplicativo ($a \cdot 1 = a$) deve existir.
- Para qualquer a, b e c em F a igualdade $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ deve ser mantida.

Por fim pode-se definir campo com sendo um conjunto de elementos os quais é possível realizar as operações de adição, multiplicação, subtração e divisão. O resultado das operações sempre é um elemento do próprio campo. A adição e a multiplicação devem satisfazer as propriedades comutativa, associativa e distributiva (SILVA, 2011).

Um exemplo de campo finito é o campo de Galois, descrito na próxima seção.

2.1 Propriedades do Campo de Galois

Os campos Galois $GF(n)$ referem-se à um conjunto de n elementos finitos, fechado com relação às operações de adição e multiplicação. Pode-se enunciar que nos campos finitos todo elemento possui o seu inverso aditivo e multiplicativo, com exceção para o elemento nulo (0).

Um campo de Galois possui $n = p^q$ elementos, porém não existe um campo de Galois para um número qualquer de elementos. Os campos de Galois são formados por um número primo de elementos ou por sua extensão, onde a extensão é uma potência do número primo, onde p representa um número primo e q é um número inteiro positivo. Os elementos são definidos por n , em que $GF(n) = 0, 1, \dots, n - 1$ sendo n a sua ordem.

Um campo descrito por $GF(2)$ possui o seguinte formato: $GF(2) = 0, 1$. Desta forma, um campo de ordem 2 possui somente dois elementos que são 0, 1, sendo denominado de campo binário. Portanto, um campo de Galois de ordem q é um conjunto de q elementos com duas operações binárias módulo- p .

Para um campo da forma $GF(2)$, pela fórmula $n = p^q$, os coeficientes são definidos como: $p = 2$ e $q = 1$. As operações binárias de adição e multiplicação executadas dentro deste campo são em módulo - 2 .

É definido como elemento primitivo, ou gerador, o elemento de ordem $(n - 1)$. As potências consecutivas do elemento primitivo gera todos os outros elementos do campo (KERL, 2004).

2.2 Polinômios Primitivos

Um polinômio $f(x)$ de ordem m é primitivo, e somente se, ele for irredutível. A irredutibilidade ocorre quando um polinômio $f(x)$ não for divisível por nenhum outro polinômio de grau inferior a ele e maior que 0. Um polinômio de grau 2 é irredutível se, somente se, ele não for divisível por polinômios de grau 1.

Por exemplo o polinômio $X^2 + X + 1$ é irredutível, porque ele não é divisível por $X + 1$, nem por X . Um polinômio irredutível (que não pode ser fatorado) $f(X)$ de ordem m é primitivo se, somente se, o menor número inteiro positivo n , para o qual $f(X)$ é um divisor de $X^n + 1$, seja igual a: $n = 2m - 1$ (FARRELL; MOREIRA, 2006).

2.3 Construção dos Campos de Galois Estendidos

Como definido, para qualquer número primo p existe um campo finito $GF(p)$ contendo p elementos. É possível estender o campo finito $GF(p)$ para um campo de p^m elementos, descrito como $GF(p^m)$ em que m é um inteiro não negativo e positivo. Portanto, $GF(p^m)$ contém como se fosse um subconjunto de elementos de $GF(p)$.

Pode-se exemplificar a expansão do campo finito $GF(p)$, tendo em mente o exemplo do campo finito $GF(2)$. Este campo é um subconjunto do campo estendido $GF(2^m)$, da mesma forma que um campo numérico real é um subconjunto do campo numérico complexo. Além dos números 0 e 1, há elementos adicionais no campo estendido os quais são representados pelo símbolo α . Cada elemento não negativo de $GF(2^m)$ pode ser representado por múltiplos de α (SKLAR, 1988b, p. 446). Portanto os elementos do campo finito estendido são definidos por:

$$GF(2^m) = 0, \alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^m-2} \quad (1)$$

As m raízes de um polinômio $p(x)$ de grau m , fornecem os seus elementos. As raízes do polinômio serão representadas por α , então faz-se $p(\alpha) = 0$ para encontrar as respectivas raízes (SKLAR, 1988b, p. 448). Sendo assim pode-se escrever para o polinômio $p(x) = X^3 + X + 1$:

$$p(\alpha) = 0 \quad (2)$$

$$\alpha^3 + \alpha + 1 = 0 \quad (3)$$

$$\alpha^3 = -\alpha - 1 \quad (4)$$

Em um campo finito $GF(q)$, as operações de soma e subtração são realizadas da mesma forma simplificando-se a equação 4 para: $\alpha^3 = \alpha + 1$. Os elementos do campo $GF(2^m)$ podem ser expressos em potências de α , polinômios de degrau $(m - 1)$ ou também como vetor binário. A raiz α^3 e os sucessivos elementos do campo $GF(2^3)$ são expressos na forma polinomial abaixo:

$$\alpha^3 = 1 + \alpha \quad (5)$$

$$\alpha^4 = \alpha \cdot \alpha^3 = \alpha(\alpha + 1) = \alpha + \alpha^2 \quad (6)$$

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(\alpha + \alpha^2) = \alpha^2 + \alpha^3 = 1 + \alpha + \alpha^2 \quad (7)$$

$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha(1 + \alpha + \alpha^2) = \alpha + \alpha^2 + \alpha^3 = 1 + \alpha^2 \quad (8)$$

$$\alpha^7 = \alpha \cdot \alpha^6 = \alpha(1 + \alpha^2) = \alpha + \alpha^3 = \alpha + 1 + \alpha = 1 = \alpha^0 \quad (9)$$

Como $\alpha^7 = \alpha^0$, portanto os oito elementos finitos do campo $GF(2^3)$ são:

$$0, \alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6 \quad (10)$$

Portanto, na Tabela 2 é ilustrado o mapeamento dos elementos finitos em termos dos elementos básicos para o campo finito estendido $GF(8)$.

2.4 Operações nos Campos de Galois

Em um campo finito as operações entre quaisquer elementos resultam em outro elemento dentro do mesmo campo. Em um campo pode ser realizadas as operações de adição, subtração, multiplicação e divisão. As operações de adição e multiplicação satisfazem as propriedades comutativa, associativa e distributiva. O elemento identidade para adição é o 0 e para multiplicação é o 1.

Essas operações podem ser realizadas por portas lógicas, tabelas, flip-flops e registradores

Tabela 2 - Mapeamentos dos Elementos Finitos em elementos básicos do Campo Finito $GF(2^3)$ com $p(x) = 1 + X + X^3$

		X^0	X^1	X^2
Elementos dos Campos	0	0	0	0
	α^0	1	0	0
	α^1	0	1	0
	α^2	0	0	1
	α^3	1	1	0
	α^4	0	1	1
	α^5	1	1	1
	α^6	1	0	1
	α^7	1	0	0

Fonte: Elaborado pelo Autor.

de deslocamento (*Shift Registers*) (KERL, 2004).

2.4.1 Adição e Subtração

Para um campo $GF(m)$ a soma de dois elementos i e j é dado pelo resto da divisão $\frac{(i+j)}{m}$. Essa operação é chamada de adição módulo m .

A adição no módulo 2 é definida pelo campo $GF(2) = 0, 1$ e a sua operação é demonstrada abaixo:

$$0 \oplus 0 = 0 \quad (11)$$

$$1 \oplus 1 = 0 \quad (12)$$

$$0 \oplus 1 = 1 \quad (13)$$

$$1 \oplus 0 = 1 \quad (14)$$

Assim para campos estendidos, com elementos 2^m , a soma de quaisquer elementos tem que resultar em um outro elemento pertencente ao mesmo campo. Este resultado acontece pelo fato do campo de Galois ser um conjunto fechado. Denota-se cada elemento não nulo de $GF(2^m)$ como um polinômio $a_i(X)$, onde pelo menos um dos coeficientes m de $a_i(X)$ é não nulo. Para $i = 0, 1, 2, \dots, 2^m - 2$, tem-se:

$$\alpha^i = a_i(X) = a_{i,0} + a_{i,1}X + a_{i,2}X^2 + \dots + a_{i,m-1}X^{m-1} \quad (15)$$

Para adição de um elemento representado na notação estendida, é feita como uma adição elemento por elemento em módulo-2. Observa-se que esta operação possui os mesmos resul-

tados que uma porta lógica XOR, executada bit a bit nos vetores a serem adicionados. Um benefício de usar a notação de elementos dos campos estendidos α^i no lugar dos elementos binários é a facilidade da representação matemática do processo de codificação e decodificação não binária (SKLAR, 1988b, p. 446). Portanto a expressão resultante da adição de dois elementos do campo finito é da forma:

$$\alpha_i \oplus \alpha_j = (a_{i0} \oplus a_{j0}) + (a_{i1} \oplus a_{j1})X + (a_{i2} \oplus a_{j2})X^2 + \dots + (a_{i,m-1} \oplus a_{j,m-1})X^{m-1} \quad (16)$$

A subtração de elementos no campo de Galois é definida como sendo a mesma porta XOR, uma vez que na subtração de módulo-2 $-\alpha = \alpha$ então $\alpha^n - \alpha^j = \alpha^n + \alpha^j$ (MORENO, 2010).

2.4.2 Multiplicação e Divisão

A multiplicação entre dois elementos, i e j , de um campo de ordem primária, m é dado por $\frac{(ij)}{m}$. Para o campo binário $GF(2)$ tem-se:

$$0 \oplus 0 = 0 \quad (17)$$

$$1 \oplus 1 = 1 \quad (18)$$

$$0 \oplus 1 = 0 \quad (19)$$

$$1 \oplus 0 = 0 \quad (20)$$

A operação de multiplicação entre dois elementos pertencentes a $GF(2^8)$ pode ser comparada às portas lógicas AND. A divisão de um elemento por outro é realizada multiplicando o elemento pelo inverso do outro que o divide. Portanto, pode-se representar a operação descrita com a equação abaixo:

$$X = \frac{\alpha^i}{\alpha^j} = \alpha^i \times \alpha^{-j} \quad (21)$$

O inverso é obtido utilizando uma tabela que contém todos os elementos pertencentes ao campo e o seu respectivo inverso (MORENO, 2010). Pela condição de fechamento implica que:

$$\alpha^{2^m-1} = \alpha^0 = 1 \quad (22)$$

Portanto, a multiplicação de dois elementos resulta num elemento cujo expoente é obtido dividindo-se a soma dos expoentes dos elementos por $2m - 1$ e tomando o resto, ou seja, na forma exponencial:

$$\alpha^i \times \alpha^j = \alpha^{(i+j) \bmod (2^m-1)} \quad (23)$$

Se os fatores forem expressos na forma polinomial, é necessário então convertê-los na forma exponencial e então efetuar o produto conforme a Equação 23.

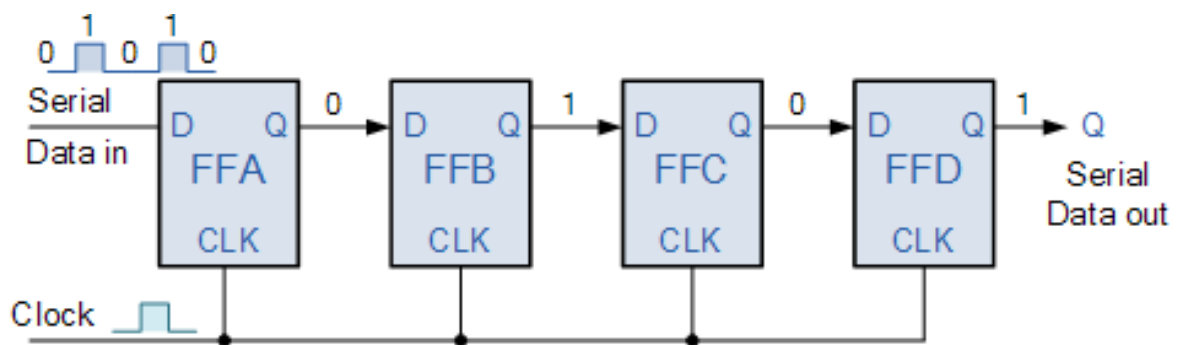
3 LINEAR FEEDBACK SHIFT REGISTERS

Neste capítulo serão tratados alguns tipos de servidores que pode ser configurado no linux através de módulos pré-instalados e configurados.

Um *Linear Feedback Shift Register* (LFSR) são descritos com base na álgebra de campos finitos, possuindo 2 formas: por meio de polinômios e a outra por meio de matrizes. Para analisar o comportamento básico do sistema pode-se utilizar primeiramente uma abordagem bit a bit, o qual refere-se ao comportamento sistema explicitando os seus componentes e funcionalidade.

Um registrador de deslocamento (*shift register*) é um circuito digital que pode tanto armazenar dados, bem como movê-los. O armazenamento dos dados é feito por meio de *flip-flops*, como por exemplo o do tipo D. Quando um sinal de *clock* é enviado ao circuito, os *flip-flops* armazenam o valor de entrada a cada estágio. Determinada saída de uma célula está conectada na entrada da próxima, desta forma os bits são propagados de um lado para o outro até encontrar a saída do sistema na última célula (FLOYD, 2009). Um shift register é ilustrado na Figura 6.

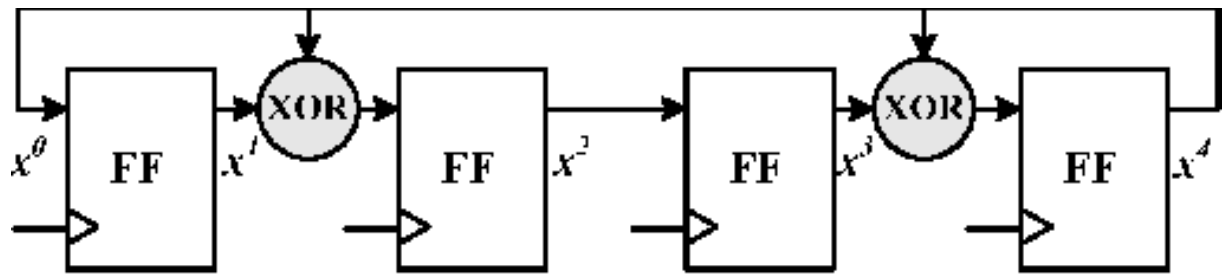
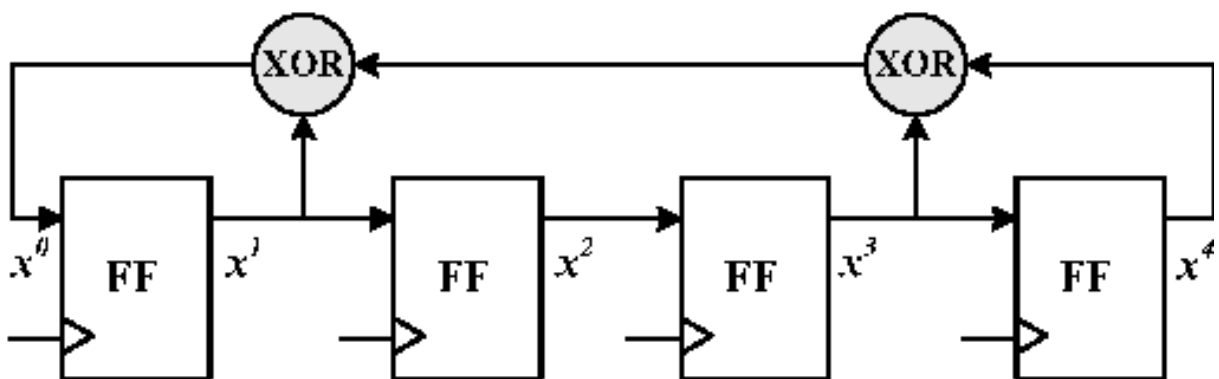
Figura 6 - Esquema de um registrador de deslocamento (*shift register*).



Fonte: Elaborado pelo Autor

Ao inserir uma realimentação no sistema *shift register*, altera-se a sua entrada e consequentemente os seus estados. A introdução da realimentação gera um sistema LFSR, porém este procedimento deve ser feito de acordo com algumas regras quando deseja-se obter uma determinada propriedade na saída.

Esta realimentação é implementada adicionando portas XOR, juntamente com *flip-flops* como ilustrado na Figura 7. Cada porta XOR inserida no sistema é denominado de *tap*, definindo um padrão nos dados de saída além de gerar um polinômio característico do LFSR.

Figura 7 - Tipos de sistemas *Linear Feedback Shift Registers*.**Fibonacci Linear Feedback Shift Register****Galois Linear Feedback Shift Register**

Fonte: Elaborado pelo Autor

A cada porta XOR inserida no sistema, indica uma expressão matemática ou um polinômio. Cada LFSR possui um padrão na geração dos bits de saída, ou seja, os dados na saída não são puramente aleatórios possuindo um ciclo de repetição. Este padrão é determinado de acordo com o número de estágios e as ligações na realimentação, ou seja, cada esquema (polinômio) implementado possui um padrão no dado da saída. Portanto, a cada ciclo o dado começa a ser repetido e o comprimento deste ciclo é dado por $2^n - 1$, em que n é o número de *shift registers* usados no sistema. Porém, para se obter o máximo comprimento no circuito deve-se usar os polinômios primitivos, ou seja, inserir determinados *taps* em estágios específicos para produzirem o máximo comprimento.

A teoria de campos finitos é utilizada para definir quando um polinômio é ou não primitivo. No capítulo 2 é descrito como obter um polinômio primitivo, ou seja, um polinômio irredutível. Normalmente, estes polinômios são utilizados para construir circuitos geradores de números aleatórios, pelo fato de ocuparem um espaço menor quando comparado com os circuitos desenvolvidos com contadores.

Na tabela da Figura 8 é esboçado alguns polinômios primitivos para 4, 8, 16, e 32 estágios. Observa-se pela tabela que há vários polinômios, ou *taps*, para o mesmo número de *flip flops* porém só há um polinômio primitivo que produz o máximo ciclo no sistema. Estes polinômios são os mesmos para as configurações Fibonacci e Galois.

Figura 8 - Tabela de polinômios com máximo comprimento para circuitos LFSR (*shift register*).

Size of LFSR	Possible feedback Polynomial	Maximum length feedback polynomial
4bit	$X^4 + X^2 + 1, X^4 + X^3 + 1$ etc.,	$X^4 + X^2 + 1$
8bit	$X^8 + X^7 + 1, X^8 + X^5 + 1,$ $X^8 + X^7 + X^6 + X^5 + 1,$ $X^8 + X^6 + X^4 + X^3 + X^2 + X^1 + 1,$ Etc.,	$X^8 + X^7 + X^6 + X^5 + 1$
16bit	$X^{16} + X^{15} + 1,$ $X^{16} + X^{13} + X^{12} + X^9 + 1,$ $X^{16} + X^{11} + X^{10} + X^7 + X^3 + X^1 + 1,$ $X^{16} + X^{15} + X^{14} + X^{12} + X^7 + X^6 +$ $+ X^3 + X^2 + 1,$ etc.,	$X^{16} + X^{14} + X^{13} + X^{11} + 1$
32bit	$X^{32} + X^{31} + 1,$ $X^{32} + X^{28} + X^{27} + X^9 + 1,$ $X^{32} + X^{21} + X^{15} + X^{13} + X^{12} + X^{10} +$ $X^8 + X^4 + 1,$ $X^{32} + X^{31} + X^{27} + X^{24} + X^{19} + X^{18} +$ $X^{17} + X^{14} + X^{13} + X^{11} + X^5 + X^4 +$ $X^1,$ etc.,	$X^{32} + X^{22} + X^2 + X^1 + 1$

Fonte: Adaptado de Abinaya e Prakasam (2014)

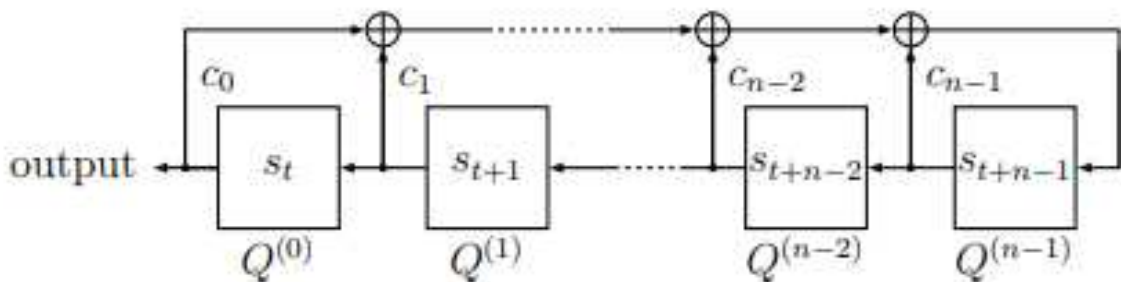
Deve-se notar que há algumas regras para a escolha do polinômio primitivo que será implementado pelo circuito LFSR. Estas regras são descritas em (ABINAYA; PRAKASAM, 2014) e possuem as seguintes características:

- O número 1 descrito nos polinômios da tabela da 8 não correspondem à um *tap* e sim à entrada para o primeiro estágio do sistema.
- Os expoentes dos termos do polinômio correspondem aos estágios, e a sua contagem normalmente é feita da esquerda para a direita. Porém, nem todo sistema pode ser montado desta forma.
- Um LFSR só terá comprimento máximo se o número de *taps* for par. Somente 2 ou 4 *taps* pode ser suficiente para gerar longas sequências.
- O número do conjunto de *taps*, tomados ao todo, deve ser relativamente primo. Em outras palavras, o polinômio tomado deve ser primitivo.
- Depois que um polinômio primitivo for encontrado, outro segue automaticamente. Se os expoentes em um sistema LFSR com n estágios são da forma $[n, A, B, C, 0]$, onde o 0 corresponde ao termo 1, então a sequência 'espelho' correspondente é $[n, n-C, n-B, n-A, 0]$. Assim, com uma sequência igual a $[32, 7, 3, 2 \text{ e } 0]$ pode-se produzir a sua contraparte igual a $[32, 30, 29, 25, 0]$. Ambos dão uma sequência de comprimento máximo.

3.1 Fibonacci Linear Feedback Shift Registers

Um sistema LFSR Fibonacci é uma das formas de criar um LFSR a partir de um *shift register*. A realimentação de sistemas Fibonacci LFSR é caracterizada como externa, uma vez que no caminho da mesma encontra-se portas XOR's. Um esquema da realimentação em circuitos fibonacci é ilustrado na Figura 9

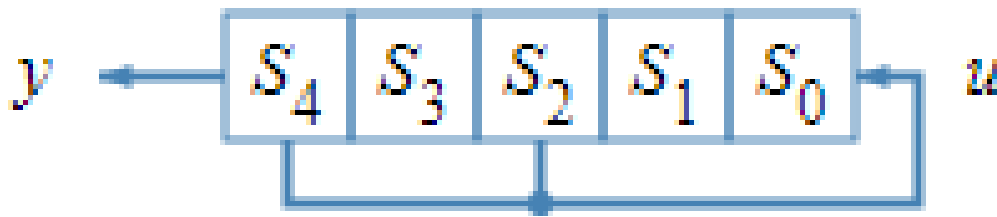
Figura 9 - Esquema de um registrador de deslocamento (*shift register*).



Fonte: Elaborado pelo Autor

Desta forma, pode-se simplificar a representação de um fibonacci LFSR uma vez que é conhecido a existência de seus estágios e ligações XOR. Uma representação simplificada é ilustrada na Figura 10. Cada estágio é representado pelo símbolo $S_j[k]$ e a ligação entre o estágio e a realimentação, feito por meio de portas XOR's, é representado por b_j o qual pode ser 0 ou 1. O j especifica o estágio e o k o período que está sendo referido.

Figura 10 - Esquema de um sistema (*Fibonacci Linear Feedback Shift Register*).



Fonte: Elaborado pelo Autor

A entrada do sistema representado na Figura 10 pode ser definido pela expressão matemática:

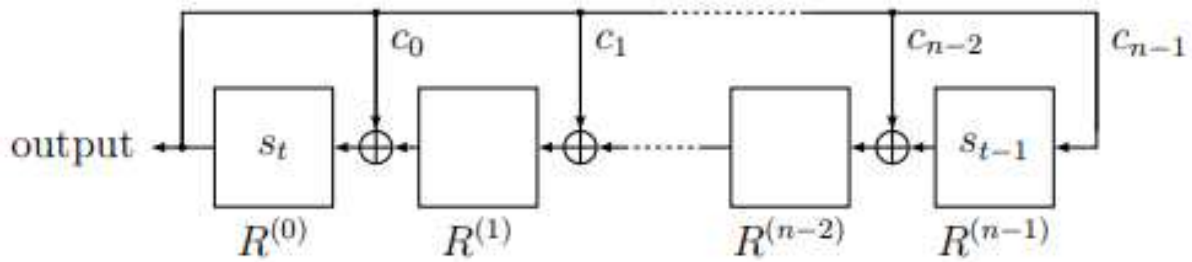
$$u[k] = \bigoplus_{j=0}^{N-1} b_j S_j[k]$$

O símbolo \bigoplus significa uma operação XOR com todas as entradas no mesmo tempo. Desta forma, na 10 as variáveis b_j são definidas como: $b_2 = b_4 = 1$ e $b_0 = b_3 = b_0 = b_1 = 0$. Portanto, a equação da entrada é definida como $u[k] = S_4[k] \oplus S_2[k] = u[k-5] \oplus u[k-3]$ e a saída é simplesmente a entrada atrasada o número de estágios no LFSR, ou seja, neste caso a saída é atrasada 5 períodos obtendo a equação $y[k] = u[k-5] = y[k-5] \oplus y[k-3]$.

3.2 Galois Linear Feedback Shift Registers

O outro tipo de sistema que pode ser construído realimentando um *shift register* é o Galois LFSR. A realimentação destes sistemas é caracterizada como interna, uma vez que as portas XOR's estão no caminho entre os *flips flops* ao invés de estarem na realimentação. Um esquema de circuitos Galois LFSR é ilustrado na Figura 11

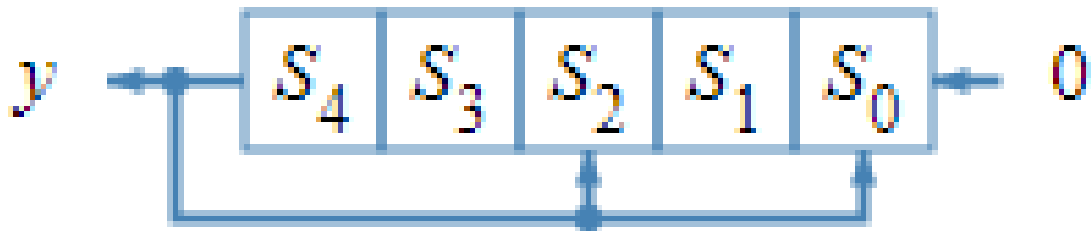
Figura 11 - Esquema de um sistema (Galois Linear Feedback Shift Register).



Fonte: Elaborado pelo Autor

Desta forma, pode-se simplificar a representação de um Galois LFSR uma vez que é conhecido a existência de seus estágios e ligações XOR. Uma representação simplificada é ilustrada na Figura 12. Cada estágio é representado pelo símbolo $S_j[k]$ e a ligação entre o estágio e a realimentação, feito por meio de portas XOR's, é representado por a_j o qual pode ser 0 ou 1. O j especifica o estágio e o k em qual período está sendo referido.

Figura 12 - Esquema simplificado de um sistema (Galois Linear Feedback Shift Register).



Fonte: Elaborado pelo Autor

Na representação de Galois de um LFSR, a entrada u é setada para 0 mas somente os estágios que possuem uma porta XOR ligada na saída podem alterar os bits transmitidos. Portanto, a saída $y[k]$ do LFSR representado na 12 pode ser definida como:

$$S_j[k] = S_{j-1}[k-1] \oplus a_j y[k-1] \quad \text{para } j > 0$$

$$S_0[k] = a_0 y[k-1]$$

$$y[k] = S_{N-1}[k] \quad N: \text{ nmero de estgios}$$

Desta forma, na 12 as variáveis a_j são definidas como: $a_0 = a_2 = 1$ e $a_1 = b_3 = b_4 = 0$. Portanto, a equação dos estágios e da saída são definidas da forma:

$$S_0[k] = y[k-1]$$

$$S_1[k] = S_0[k-1] = y[k-2]$$

$$S_2[k] = S_1[k-1] \oplus y[k-1] = y[k-3] \oplus y[k-1]$$

$$S_3[k] = S_2[k-1] = y[k-4] \oplus y[k-2]$$

$$S_4[k] = S_3[k-1] = y[k-5] \oplus y[k-3]$$

$$y[k] = S_4[k] = y[k-5] \oplus y[k-3]$$

Na configuração de Galois os estágios que não possuem *taps* conectados, são deslocados uma posição para o próximo estágio. Os *taps*, por outro lado, realizam uma operação XOR com o bit de saída do estágio antes de serem armazenados na próximo *flip-flop*. O efeito disto é que quando o bit de saída é zero, todos os bits no registrador mudam para a direita inalterados, e o bit de entrada se torna zero. Quando o bit de saída é um, os bits nas posições da derivação são invertidos (se forem 0, eles se tornarão 1, e se forem 1, eles se tornarão 0). Desta forma, o registrador inteiro será deslocado para a direita e o bit de entrada torna-se 1.

Um Fibonacci LFSR, na presença de muitos taps, opera em velocidades mais baixas quando comparado com os sistemas Galois LFSR. Isto ocorre pelo fato das diversas portas XOR no caminho da realimentação ocasionarem um *delay* maior do que somente uma porta entre cada estágio descrito nos sistemas Galois LFSR. Porém, sistemas Fibonacci LFSR podem operar em velocidades altas como os Galois, se existir no máximo uma porta XOR no caminho da realimentação (DHINGRA, 2019).

4 SCRAMBLERS

Em sistemas de comunicação, um *scrambler* é um dispositivo que consegue embaralhar ou modificar uma mensagem no lado do emissor para tornar a mensagem ininteligível ou com determinadas propriedades para o receptor que não possui a capacidade de interpretar aquela mensagem. O embaralhamento é realizado pela adição ou reordenamento de componentes ao sinal original a fim de dificultar a extração do mesmo. Alguns *scramblers* modernos são, na verdade, dispositivos de criptografia, permanecendo o nome devido às semelhanças no uso, em oposição à operação interna (HASSAN, 2019).

Nas comunicações digitais, em muitos casos, um *scrambler* é usado para manipular um fluxo de dados antes de transmitir. As manipulações são invertidas por um *descrambler* no lado de recepção. Estas manipulações tem o objetivo de embaralhar o dado a ser transmitido, porém pode não haver criptografia neste processo. A intenção nesse caso não é tornar a mensagem ininteligível, mas dar aos dados transmitidos propriedades de engenharia úteis (HASSAN, 2019). Estas propriedades são úteis para a codificação 64b/66b.

Estas propriedades podem serem resumidas em duas principais (HASSAN, 2019):

- O embaralhamento em sistemas de comunicação digital facilita a atuação dos circuitos de recuperação de *clock*, controle de ganho e outros circuitos adaptativos do receptor pela eliminação de sequências consistindo apenas de 0's ou 1's.
- Um circuito *scrambler* torna o espectro de potência do sinal mais disperso para atender ao requisitos de densidade espectral de potência. Este requisito trata da potência concentrada em uma faixa de frequência estreita, uma vez que esta característica pode interferir canais devido à modulação cruzada e à intermodulação causada pela não linearidade do receptor.

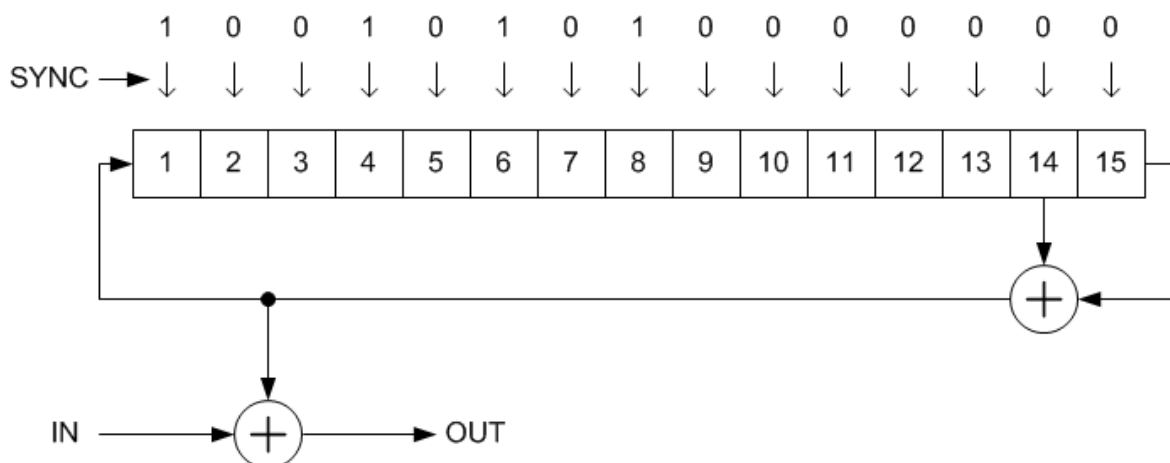
Os *scramblers* usualmente são definidos com base em LFSR por conta de suas boas características estatísticas, bem como pela facilidade de implementação em hardware. Os *scramblers* podem ser separados em dois tipos: *Scramblers* Aditivos (Synchronous) e Multiplicativos (Self-Synchronizing).

4.1 *Scramblers* Aditivos (Synchronous)

Scramblers Aditivos transformam o fluxo de dados de entrada, aplicando uma sequência binária pseudoaleatória (PRBS) por adição módulo-2. Em alguns casos, um PRBS pré-calculado

é armazenado em uma memória ROM, sendo usado quando necessário. Entretanto, frequentemente é gerado por um LFSR devidamente implementado no sistema. Um esquema de um *scrambler* aditivo é ilustrado na Figura 13.

Figura 13 - Esquema de um *Scramblers* Aditivo.



Fonte: Elaborado pelo Autor.

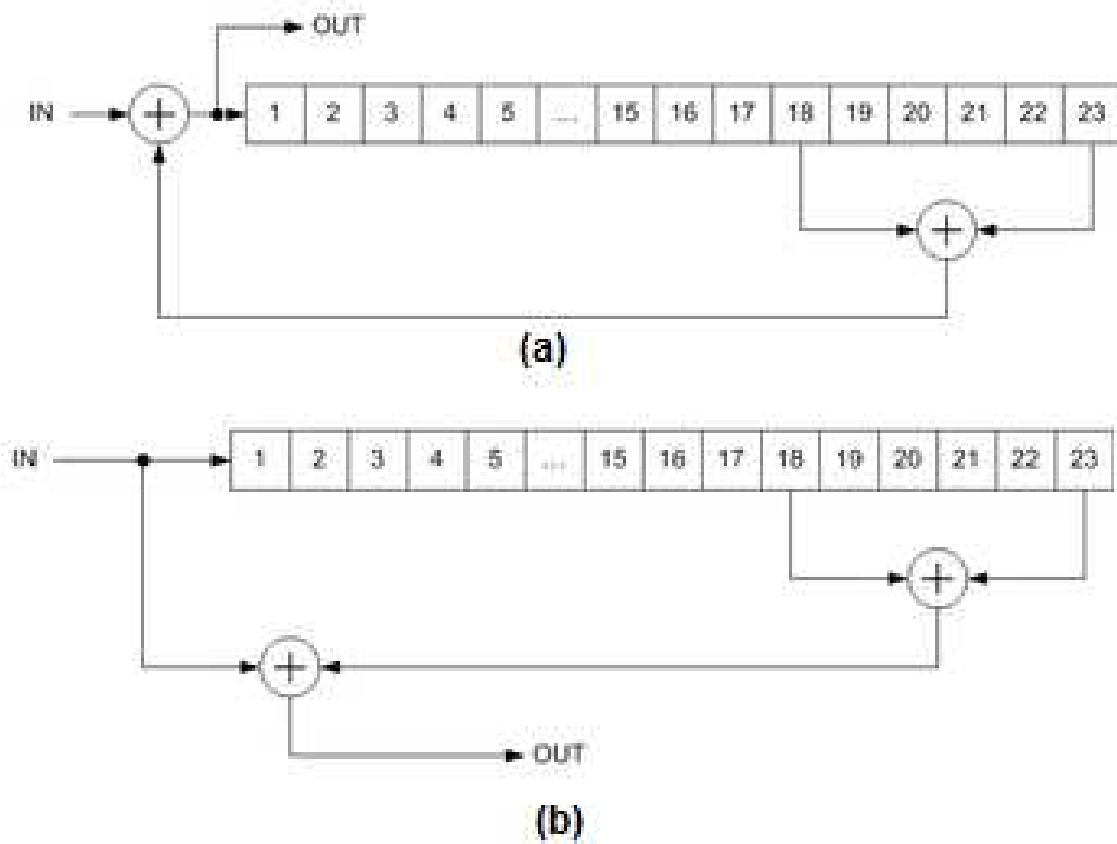
Uma palavra de sincronização é usada para assegurar uma operação síncrona do LFSR. Esta palavra é um padrão inserido no fluxo de dados em intervalos de tempos iguais, como por exemplo logo após o tempo para processar o dado introduzido no circuito. Um receptor procura algumas palavras de sincronização em dados adjacentes e, portanto, determina o local em que seu LFSR deve ser recarregado com um estado inicial predefinido (HASSAN, 2019).

O *descrambler* aditivo é apenas o mesmo dispositivo que o *scrambler* aditivo. O *scrambler* / *descrambler* aditivo são definidos pelo polinômio de seu LFSR e seu estado inicial.

4.2 Scramblers Multiplicativos (Self-Synchronizing)

Os *scramblers* multiplicativos são chamados assim por executarem uma multiplicação do sinal de entrada pela função de transferência do *scrambler* no espaço Z. Eles são sistemas lineares invariantes no tempo. Ao contrário dos *scramblers* aditivos, os *scramblers* multiplicativos não precisam da palavra sincronização, por isso eles também são chamados de auto-sincronizadores. Um exemplo da topologia dos *scramblers* multiplicativos está representado na 14. Na letra (a) é representado um *scrambler* e na letra (b) um *descrambler* (HASSAN, 2019).

Figura 14 - Esquema de um Scramblers Multiplicativo.



Fonte: Elaborado pelo Autor.

Scrambler multiplicativo é definido similarmente por um polinômio, que também é uma função de transferência do *descrambler*. A saída codificada, $S(x)$, é gerada no transmissor dividindo os dados $M(x)$ por um polinômio gerador $G(x)$:

$$S(x) = \frac{M(x)}{G(x)}$$

A operação de divisão é realizada bit a bit e cada etapa da divisão resulta em um novo bit embaralhado. O receptor reordena o sinal embaralhado multiplicando pelo mesmo polinômio gerador:

$$M(x) = S(x) * G(x)$$

A implementação da divisão e multiplicação polinomial é feita usando *Linear feedback shift registers*, além de toda a teoria de campos finitos e suas operações em módulo 2.

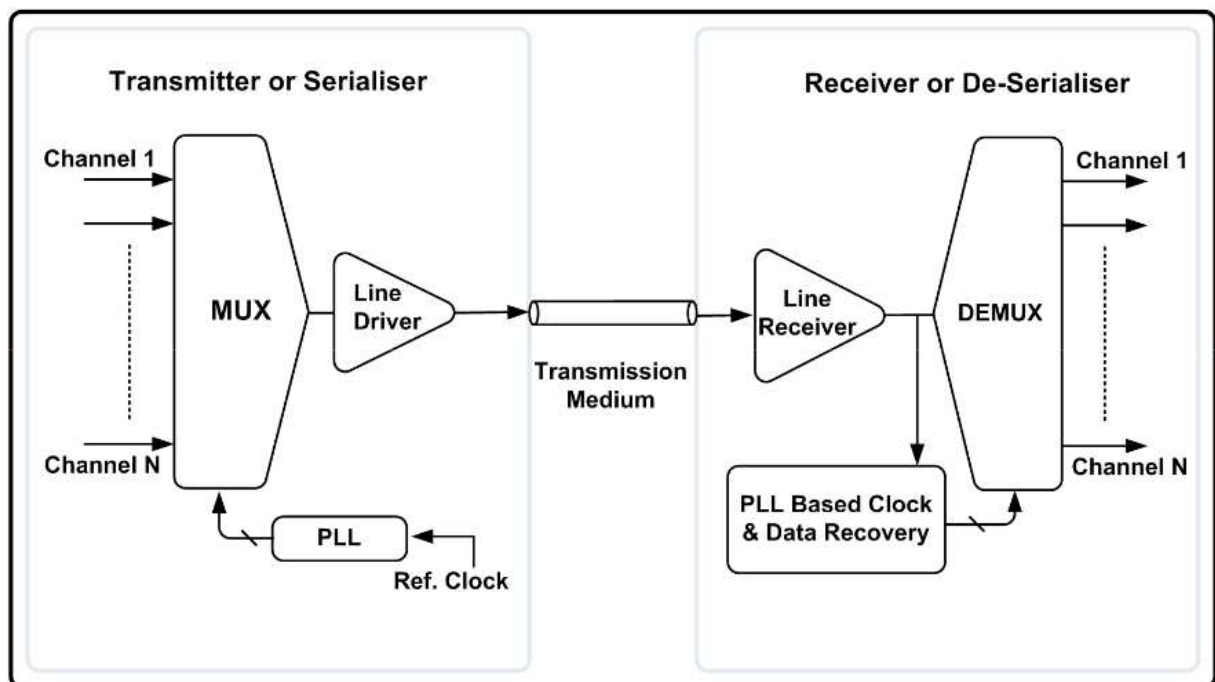
Os embaralhadores multiplicativos levam à multiplicação de erros durante a descodificação. Um erro de bit único na entrada do decodificador resultará em w erros na sua saída. Este aumento no número de erros depende do número de *taps* existente no sistema. Caso existir duas *taps*, um único bit de erro inserido no sistema resultará em 3 bits errôneos na saída do *descrambler* (HASSAN, 2019).

5 DESCRIÇÃO DO LINK SERIAL DE ALTA VELOCIDADE

O modelo generalizado de um Link Serial de Alta Velocidade é ilustrado na figura 15, consiste em um serializador e transmissor (TX) acionado por um *clock* proveniente de um PLL, um canal, um receptor (RX) e um desserializador acompanhado de uma unidade CDR (Clock-Data Recovery). O serializador aceita o fluxo de dados paralelo de entrada e o converte em um fluxo de dados serial, enviando-os ao transmissor (RATAN, 2014, p. 5).

Pela Figura 15, o receptor compreende basicamente um amostrador cujo objetivo é coletar amostras do fluxo de bits dos dados recebidos do canal e recuperando tanto os dados transmitidos quanto o *clock*. Quando o receptor recupera os bits seriais transmitidos, eles são enviados ao bloco desserializador, cuja tarefa é converter os dados seriais recebidos de volta à sua forma paralela original para futuras interfaces (RATAN, 2014, p. 6).

Figura 15 - Diagrama de bloco Típico de links de alta velocidade.



Fonte: Assaad (2009, p. 11)

O canal transporta o sinal de dados do transmissor para o receptor e pode ser elétrico, óptico ou uma combinação de ambos. Para comunicações com canal de longa distância, ele é uma fonte dominante de ruído, instabilidade de fase e *jitter*. No entanto, para comunicações de curta distância é considerado uma fonte desprezível de ruído e instabilidade (ASSAAD, 2009).

Portanto, deve-se entender o funcionamento de cada bloco representado na Figura 15, para compreender a sua função para contornar os problemas das comunicações de alta velocidade.

5.1 Serializador

O circuito serializador converte os dados do barramento paralelo de entrada em um formato de fluxo de bits serial. É um bloco totalmente digital e antecede o circuito do driver do canal TX. Dentro do driver TX exige-se a adição de um amplificador de carga de saída de 50Ω , ou, em certos casos, a adição de um circuito sofisticado capaz de fornecer sinal compatível a um óptico. Na maioria dos sistemas de comunicação, os dados são codificados primeiro. O processo de codificação pode incluir compactação, criptografia, verificação de erros e enquadramento. Basicamente, um serializador é essencialmente um circuito Multiplexador cujo *clock* de acionamento para o processo de serialização é o sinal gerado pelo TX PLL. Serializadores também podem ser implementados usando *shift registers* (ASSAAD, 2009).

5.2 Driver Amplificador

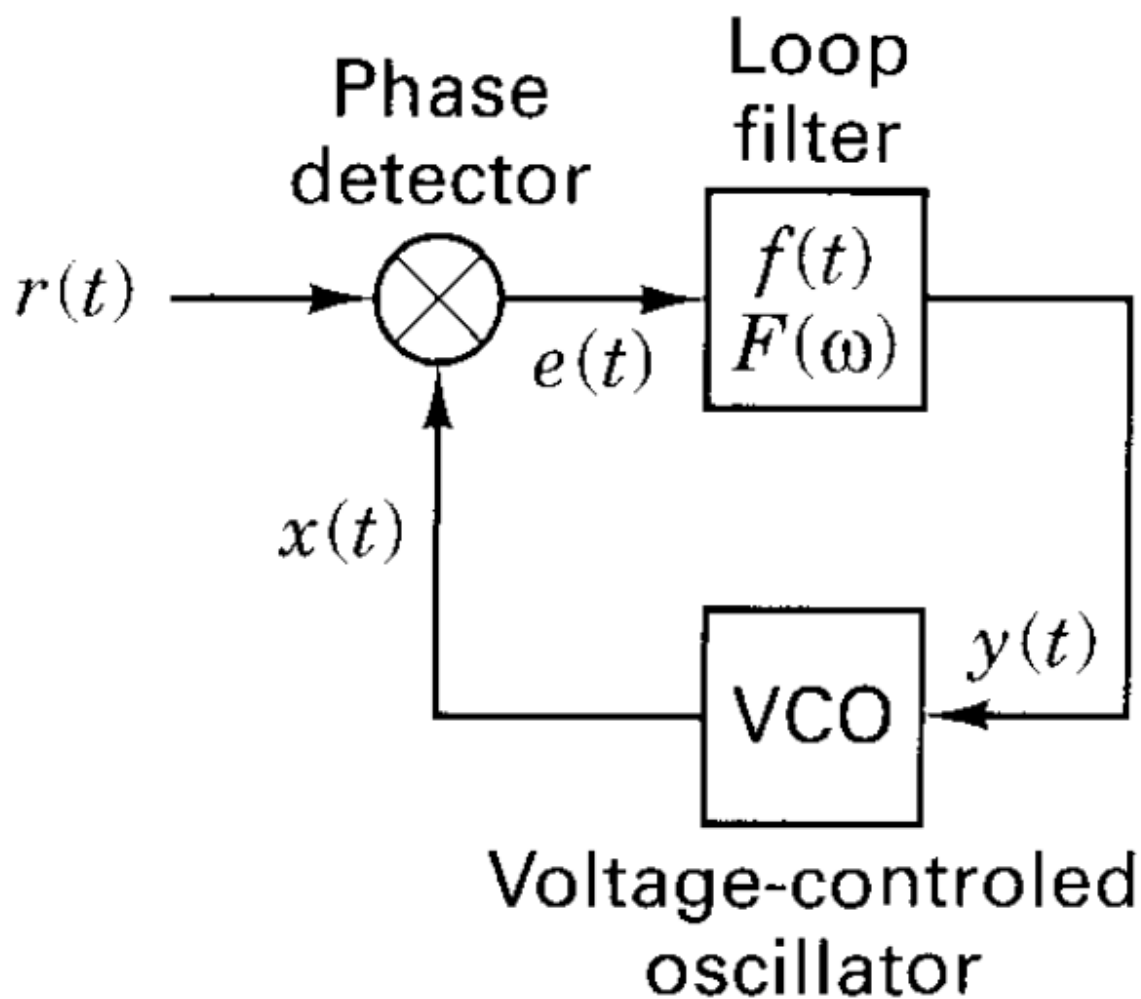
Os amplificadores de driver são encontrados nas extremidades TX e RX. O DA (Driver Amplifier) é usado para amplificar o fluxo de bits serial de entrada antes de ser enviado ao receptor através do canal. Outra tarefa importante cumprida pelo DA é que ele fornece terminações de impedância que terminam a entrada ou saída do canal com impedância de 50Ω .

5.3 Phase-Locked Loop (PLL)

Um PLL é um sistema de feedback negativo cujo único objetivo é usar um *clock* de referência e gerar um *clock* local no chip em uma frequência desejada, estando o *clock* de saída em fase ao *clock* de entrada. Os PLLs são usados em todos os sistemas modernos de alta velocidade. Os cristais piezoelétricos são usados exclusivamente com *clocks* de referência para quase todos os sistemas de interface no chip, pois possuem a mais alta pureza espectral e podem emitir sinais de *clock* verdadeiramente periódicos e sem *jitter*, tipicamente até 200 MHz (ASSAAD, 2009).

Devido à demanda insaciável por sinalização robusta e de alta velocidade, um mero oscilador de cristal não é suficiente para atender aos requisitos necessários sendo necessário um PLL. Portanto, este sistema produz sinais de *clock* com ruído de tempo mínimo, isto é, com o menor *jitter* possível (no domínio do tempo) e ruído de fase (no domínio da frequência). No nível de bloco, um PLL típico possui 3 componentes básicos: *phase-detector*, *loop filter* e um *voltage-controlled oscillator* (VCO) (SKLAR, 1988b, p. 603). Na Figura 16 é ilustrado um esquema de um PLL básico.

Figura 16 - Esquemático de um PLL básico.



Fonte: Sklar (1988b, p. 604)

Pela Figura 16, o *phase-detector* é um dispositivo que produz uma medida da diferença de fase entre um sinal de entrada e a referência. Como o sinal de entrada e a referência mudam entre si, o erro de fase torna-se um sinal variável no tempo no *loop filter*. O *loop filter* comanda a resposta do PLL às variações no sinal de erro. Um *loop filter* bem projetado deve ser capaz de rastrear alterações na fase do sinal de entrada, mas não responder impulsivamente ao ruído do receptor. O VCO é o dispositivo que produz a réplica da transportadora (SKLAR, 1988b, p. 604).

O VCO é um oscilador sinusoidal cuja frequência é controlada por um nível de tensão na entrada do dispositivo. Sua frequência de saída é uma função linear de sua tensão de entrada sobre alguma faixa da tensão de entrada e saída. Uma tensão de entrada positiva faz com que a frequência de saída do VCO seja maior que seu valor não controlado (ω_0), enquanto uma tensão

negativa fará com que seja menor. O bloqueio de fase é obtido alimentando um versão filtrada da diferença de fase (ou seja, o erro de fase) entre a entrada sinal $r(t)$ e a saída do VCO, $x(t)$, de volta à entrada do VCO, $y(t)$. Na Figura 16, o *phase-detector* é mostrado como multiplicador, o *loop filter* é descrito por sua função de resposta ao impulso $f(t)$, com Transformada de Fourier $F(w)$, e o VCO já está indicado (SKLAR, 1988b, p. 604).

No caso de receptores digitais modernos, o detector de erros pode ser matematicamente geralmente muito mais complicado do que o multiplicador simples mostrado na Figura 16. Por exemplo, o detector de erros pode ser um conjunto de correlacionadores de filtros correspondentes, cada correspondente a um deslocamento de fase ligeiramente diferente, alimentando uma função de ponderação ou decisão. A saída da função de ponderação seria a estimativa do *phase-error*. Esta função pode ser matematicamente muito complexa, mas seria facilmente aproximada usando tecnologia digital moderna. O VCO pode não parecer sinuoidal, mas pode ser implementado como uma memória somente de leitura cujos ponteiros são controlados por uma combinação de um *clock* e a saída do estimador de erros. Não obstante, os princípios básicos das diferentes topologias ainda são ilustrados no modelo simples da Figura 16 (SKLAR, 1988b, p. 604).

5.4 Equalizador

A equalização é um método de combate aos efeitos da Interferência Inter-Simbólica (ISI) causada pela limitação da banda do canal. Os equalizadores são implementados tipicamente como filtros adaptativos lineares ou não lineares ou usando técnicas de processamento de sinais. A equalização é realizada antes de se transmitir o dado no canal, realizando basicamente a passagem do sinal TX através de um filtro cuja função de transferência é o inverso da função de transferência de canal. Por outro lado, a equalização no RX é usada para desfazer a distorção ocorrida no sinal recebido devido à perda e dispersão do canal. A maioria dos esquemas de equalização de RX são adaptáveis e implementados usando técnicas de processamento digital de sinal para cancelar a perda do canal dos bits de dados recebidos (SKLAR, 1988b, p. 150).

5.5 Clock and Data Recovery (CDR)

Um CDR, como o nome sugere, é responsável por extrair as informações do *clock* do transmissor do sinal recebido. Nos sistemas de comunicações de alta velocidade modernos, o mecanismo de recuperação de *clock* é essencial na extremidade do receptor. Isto deve-se, pelo fato de as informações do *clock* TX geralmente serem incorporadas ao fluxo de bits recebidos na entrada do receptor. No fundo, um CDR é essencialmente um circuito PLL modificado, no qual o *phase-detector* agora precisa coletar o fluxo de dados recebido e extrair dele os dados e as informações de fase (LI, 2015). Na Figura 17 é ilustrado um esquema básico de um CDR.

5.7 Esquemas de Codificação

Os pulsos de *non-return-to-zero* (NRZ) são comumente usados ??como função básica para a transmissão discreta de dados. A resposta do canal ao pulso NRZ é definida como a resposta do pulso, sendo tradicionalmente usada para analisar e modelar os efeitos de um canal na transmissão de dados e no design de equalizadores no caso de canais com grande atenuação na frequência de interesse. Além da sinalização NRZ, os projetistas também podem implementar técnicas avançadas de modulação para uma sinalização mais rápida e robusta. Dentre essas técnicas, pode-se citar o PAM multinível, como o PAM-4, ao qual possui eficiências espectrais muito mais altas e pode transmitir 2 bits por símbolo. Isso permite a transmissão de uma quantidade equivalente de dados na metade da largura de banda do canal.

Nos links seriais modernos, juntamente com os esquemas de sinalização, uma certa quantidade de codificação também está presente no fluxo de dados. Os esquemas de codificação mais usados são 8B/10B e 16B/20B sendo codificações poderosas por melhorar o BER. A única desvantagem da codificação é que ela adiciona ainda mais complexidade ao projeto do transceptor, pois o circuito do codificador/decodificador precisa ser projetado e mais bits precisam ser enviados através do mesmo canal de largura de banda limitada.

6 FORWARD ERROR CORRECTION (FEC)

Em sistemas de transmissão em alta velocidade muitos tipos de erros podem ocorrer, originando-se de diversas fontes no circuito, como por exemplo: ruído térmico, *crosstalk*, *jitter* de fase, atenuação do sinal, dessincronização entre transmissor e receptor. Diante disso, várias técnicas foram desenvolvidas para controlarem os erros em sistemas de comunicações. Basicamente existe dois tipos de mecanismos que realizam o controle de erros: Detecção de erro e Correção de erro (SKLAR, 1988b).

A detecção de erro é feita pela introdução de redundância no dado, sendo possível verificar se houve um erro na transmissão. Este tipo de sistema normalmente é usado com a técnica de retransmissão em cenários onde há uma baixa taxa de erros recebidos. Entretanto, há vários esquemas de detecção além da retransmissão, como por exemplo: paridade, *checksum*, *Cyclic Redundancy Check* (CRC), distância de Hamming baseado no esquema do *checksum* e polaridade (AGRAWAL, 2011).

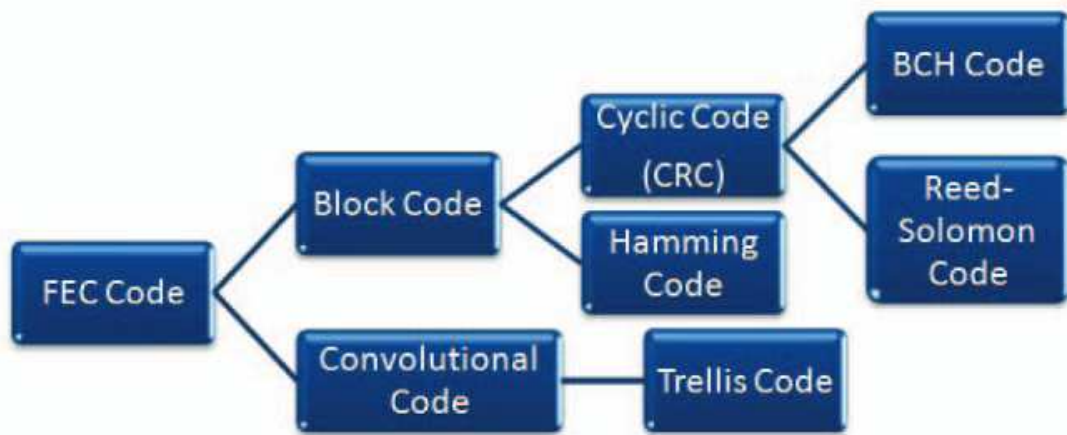
Para cenários com altas taxas de erros é preferível técnicas de correção de erro, uma vez que retransmissões geram grande perda na performance. Para empregar correções de erro deve-se utilizar o FEC como técnica de implementação nos sistemas de comunicação. Esta ferramenta insere no dado a ser transmitido redundâncias de acordo com um algoritmo. O dado ao chegar no receptor, é decodificado pelo algoritmo correspondente e os erros durante a transmissão são detectados. Portanto, a latência da decodificação do FEC é constante e mais viável em cenários com alta taxa de erros. Porém, existe a possibilidade de implementar sistemas com retransmissão e um FEC, para contornar o problema da mudança da condição de transmissão do canal (Huang; Zhang, 2011).

O sistema implementado nesse trabalho usa a técnica de correção de erros, especificamente um FEC implementado com o algoritmo Reed-Solomon. Desta forma, na próximas seções terão o objetivo de dar uma ideia geral do FEC e focar mais na descrição do algoritmo Reed-Solomon.

6.1 Tipos de FECs

A maioria dos sistemas FEC podem serem divididos em duas categorias: *Block Codes* e *Convolutional Codes*. Na Figura 18 é ilustrado um esquemático de alguns tipos de FECs e suas classificações.

Figura 18 - A relação entre diferentes tipos de FEC.



Fonte: Huang e Zhang (2011)

6.1.1 Block Codes

Os códigos de bloco funcionam em blocos de tamanho fixo de bits ou símbolos de tamanho predeterminado, permitindo uma transmissão digital confiável através de canais de comunicação sujeitos a ruído do canal. O código de bloco adiciona "r" bits de verificação sobre a mensagem original com k bits, alternado a mensagem que será transmitida em uma sequência de comprimento fixo de n bits, chamados palavra de código. As palavras de bits de verificação, ou seja, a redundância inserida permite ao receptor detectar os bits de erro através de algum algoritmo de decodificação e corrigi-lo sem retransmitir o pacote (Huang; Zhang, 2011).

6.1.1.1 Cyclic Codes

Os códigos de correção de erro cíclicos são códigos de bloco lineares, que possuem estruturas algébricas convenientes para detecção e correção de erros eficientes. Se $C = [c_{n-1}c_{n-2}\dots c_1c_0]$ for uma palavra de código de um código cíclico, $[c_{n-2}c_{n-3}\dots c_0c_{n-1}]$, obtido por deslocamento cíclico de todas as elementos à esquerda também são palavra de código. Em outras palavras, toda mudança cíclica de uma palavra de código resulta em outra palavra de código. Essa estrutura cíclica é muito útil em operações de codificação e decodificação, pois é muito fácil de implementar em hardware (Huang; Zhang, 2011).

No transmissor, uma função é usada para calcular um valor para os bits de verificação com base nos dados a serem transmitidos. Esses bits de verificação são transmitidos juntamente com os dados ao receptor. O receptor executa o mesmo cálculo nos dados recebidos e os compara

com os bits de verificação que recebeu. Se corresponderem, considera-se que nenhum erro de bit ocorreu durante a transmissão. Embora seja possível que certos padrões de erro não sejam detectados, uma seleção cuidadosa da função do gerador minimizará essa possibilidade. Usando diferentes tipos de polinômios do gerador, é possível usar os bits de verificação para detectar tipos diferentes de erros, como todos os bits únicos, erros, todos os erros de bit duplo, qualquer número ímpar de erros ou erro de burst menor que um valor específico (GUPTA; VERMA, 2012).

Um código cíclico que visa à detecção de erros é chamado código *Cyclic Redundancy Check* (CRC). Com os caracteres simples de implementar, fáceis de analisar e bons em detectar erros comuns causados por ruídos, os CRCs se tornam populares e amplamente utilizados na detecção de erros (Huang; Zhang, 2011).

6.1.1.2 *Hamming Codes*

Os códigos Hamming detectam e corrigem um único erro de bit no bloqueio de dados. Nesses códigos, cada bit é incluído em um conjunto exclusivo de bits de paridade. A presença e a localização de um erro de paridade único são determinadas pela análise de paridades de combinações de bits recebidos, para produzir uma tabela de paridades em que cada uma corresponde a uma combinação de erro de bit específica. Esta tabela de erros é conhecida como síndrome do erro. Se todas as paridades estiverem corretas de acordo com esse padrão, pode-se concluir que não há um único erro de bit na mensagem (pode haver vários erros de bit). Se houver erros nas paridades causados por um único erro de bit, o bit de dados incorreto pode ser encontrado adicionando as posições das paridades erradas.

Embora os códigos de Hamming sejam fáceis de implementar, surge um problema se mais de um bit na mensagem recebida estiver incorreto. Em alguns casos, o erro pode ser detectado, mas não pode ser corrigido. Em outros casos, o erro pode ser detectado, resultando em uma interpretação incorreta das informações transmitidas. Portanto, são necessários esquemas de detecção e correção de erros mais robustos que possam detectar e corrigir vários erros nas mensagens transmitidas.

6.1.1.3 *Bose-Chaudhuri Hocquen-hem (BCH) Codes*

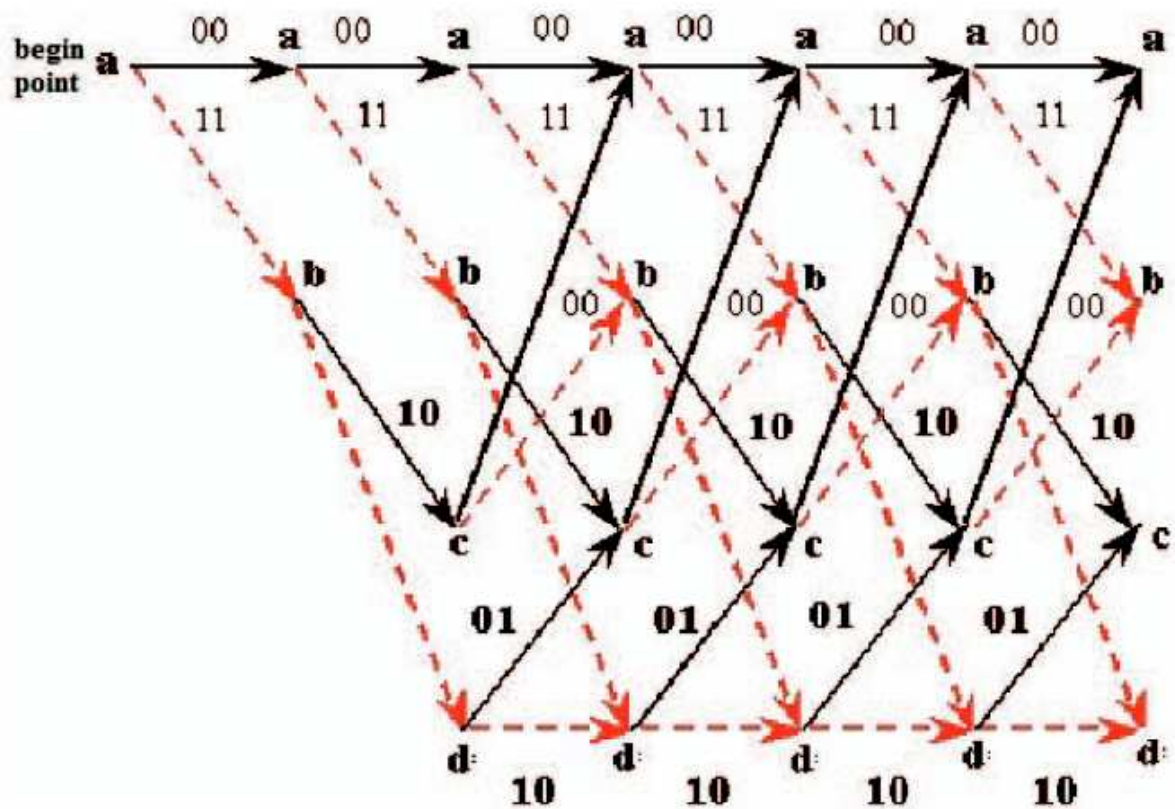
Os códigos de correção de erros BCH são códigos cíclicos. A principal vantagem é a facilidade com que eles podem ser decodificados, por meio de um método algébrico conhecido como decodificação de síndrome. Isso permite que um hardware eletrônico muito simples execute a tarefa, evitando a necessidade de um computador e implicando que um dispositivo de decodificação pode ser pequeno e com baixo consumo de potência (Huang; Zhang, 2011).

6.1.2 Convolutional Codes

Os códigos convencionais são adequados para a transmissão de fluxos de bits ou símbolos e possuem um curto tempo de atraso. Diferentemente do código de bloco, o código de n bits gerado pelo código convolucional é determinado e calculado não apenas pelo sinal de entrada de k bits durante o tempo atual, mas também pelos pacotes $N-1$ transmitidos no passado. Assim, o código convolucional geralmente representa como $[n, k, N - 1]$. Um código convolucional pode ser transformado em um código de bloco, se desejado.

Para diminuir a complexidade do processo de codificação do código convolucional, os códigos Trellis são amplamente aplicados. O algoritmo Viterbi é uma representação típica do código Trellis. Uma ilustração do código Trellis é representada na Figura 19

Figura 19 - Diagrama do código Trellis para o codificador



Fonte: Huang e Zhang (2011)

6.2 Propriedades dos códigos FEC

Quatro propriedades básicas com base nas quais um FEC pode ser selecionado são descritas a seguir:

- **Ganho de codificação:** Expressa em dB, a diferença entre $\frac{Eb}{N_0}$ necessária para obter uma determinada probabilidade de erro de bit com e sem codificação.
- **Taxa:** É a razão entre o número de bits de mensagem transmitidos e o número total de bits transmitidos $(\frac{k}{n})$.
- **Penalidade de energia:** é o resultado do envio de uma grande constelação que inclui a paridade extra ou bits de verificação do erro código de correção. $Penalidade = \frac{2B-1}{2b-1}$.
- A complexidade do envolvido na codificação e decodificação aumenta o tempo de design e a latência.

6.3 Código Reed-Solomon

Os códigos Reed-Solomon (R-S) são cíclicos e não binários com símbolos construídos com m -bits, em que m é um número maior do que 2. Para um R-S da forma (n,k) , em que k representa o número de símbolos no dado e n o número de símbolos de código que a codificação insere (SKLAR, 1988b, p. 438). Para um código R-S (n,k) tem-se as seguintes propriedades:

$$0 < k < n < 2^m + 2 \quad (24)$$

$$(n, k) = (2^m - 1, 2^m - 1 - 2 * t) \quad (25)$$

Em que t representa a capacidade de símbolos que o código pode corrigir e $n - k = 2 * t$ é o número de paridade. Por meio de uma codificação R-S é possível obter a menor distância Hamming para um código linear com o mesmo comprimento do *encoder* e *decoder*. Em particular, um código C pode recuperar k bits de erro se, e somente se, a distância mínima de Hamming entre duas de suas palavras do código é pelo menos $k + 1$ (SKLAR, 1988b, p. 438).

Diz-se que um código C corrige k erros, caso para cada palavra w no espaço Hamming subjacente H , existir no máximo uma palavra de código c (de C), de modo que a distância de Hamming entre w e c seja no máximo k . Em outras palavras, um código corrige k -erros se, e somente se, a distância mínima de Hamming entre duas de suas palavras de código é pelo menos $2k + 1$. Isso é mais facilmente entendido geometricamente como qualquer esfera fechada de raio k centrada em palavras de código distintas sendo disjuntas (SKLAR, 1988b, p. 437).

Para o R-S, a mínima distância é dado por:

$$d_{min} = n - k + 1 \quad (26)$$

O código é capaz de corrigir até t símbolos de erros no bloco, em que t é dado por:

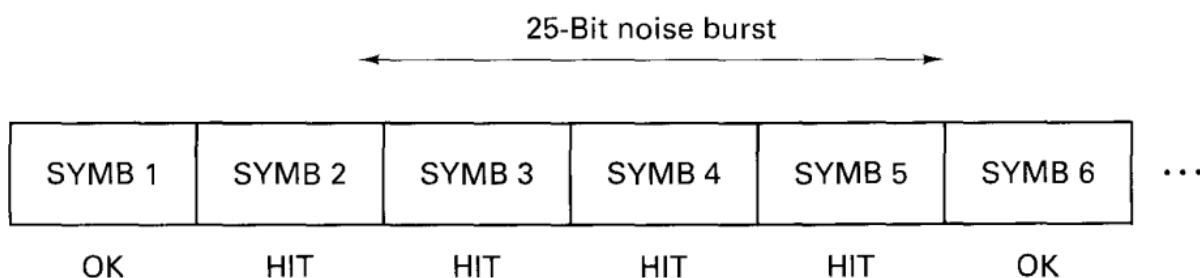
$$d_{min} = \left\lceil \frac{d_{min}-1}{2} \right\rceil = \left\lceil \frac{n-k}{2} \right\rceil \quad (27)$$

em que $\lceil x \rceil$ significa o maior inteiro não maior do que x . Portanto, pela equação 27 é fácil perceber que um R-S requer $2t$ símbolos de paridade onde portanto $m = n - k$ (SKLAR, 1988b, p. 438).

6.3.1 Performance de códigos R-S contra Burst Noise

Considere um código R-S $(n, k) = (255, 247)$, em que cada símbolo é composto por $m = 8$ bits (esses símbolos são geralmente chamados de bytes). Como $n - k = 8$. Pela equação 27, esse código pode corrigir quaisquer erros de 4 símbolos em um bloco de 255 bits. Imagine a presença de um *burst noise*, com duração de 25 bits e perturbando um bloco de dados durante a transmissão, conforme ilustrado na Figura 20.

Figura 20 - Bloco de dados deteriorado por 25 bits de *noise burst*



Fonte: Sklar (1988b, p. 441)

Neste exemplo, observe que um *burst noise* dura 25 bits contíguos perturbando exatamente 4 símbolos. O decodificador R-S para o código $(255, 247)$ corrigirá erros de 4 símbolos sem considerar o tipo de dano sofrido pelo símbolo. Em outras palavras, quando um decodificador corrige um byte, substitui o byte inteiro incorreto pelo correto. Esta substituição pode ocorrer pelo fato de um bit ou todos os 8 bits estarem incorretos. Isso dá ao código R-S uma tremenda vantagem sobre o *burst noise* em relação aos códigos binários.

6.3.2 Codificador Reed-Solomon

O polinômio gerador para um código R-S é descrito da seguinte forma:

$$g(x) = g_0 + g_1X + g_2X^2 + \dots + g_{2t-1}X^{2t-1} + X^{2t} \quad (28)$$

em que $2t$ é número de símbolos de paridade, portanto o grau do polinômio é igual ao número de símbolos de paridade. Desde que o polinômio gerador é de grau $2t$, deve-se encontrar as raízes do polinômio gerador $g(x)$ como: $\alpha, \alpha^2, \dots, \alpha^{2t}$. Considerando o exemplo de um R-S $(7, 5)$, pode-se descrever o polinômio gerador em termos de $2t = n - k = 7 - 5 = 2$ raízes como se segue:

$$g(X) = (X - \alpha)(X - \alpha^2) = X^2 - (\alpha^2 + \alpha)X + \alpha^3 = X^2 - \alpha^4X + \alpha^3 = X^2 + \alpha^4X + \alpha^3 \quad (29)$$

Uma importante propriedade da palavra código é a de ser divisível pelo polinômio gerador. Considerando que o quociente e o resto da divisão entre uma palavra código e o polinômio gerador são representados respectivamente pelos polinômios $Q(x)$ e $P(x)$, pode-se afirmar que a relação:

$$C(x) = M(x)x^{n-k} + P(x) = Q(x)G(x) \quad (30)$$

é verdadeira. Os coeficientes da palavra código são representados também na forma polinomial em $C(x)$. O polinômio $C(x)$ de grau $n - 1$ é pertencente a $GF(2^m)$ se e somente se $C(x)$ for divisível por $G(x)$. Conforme mencionado anteriormente, para calcular o polinômio de paridade $P(x)$, o polinômio de mensagem $M(x)$ é deslocado $n-k$ posições e dividido por $G(x)$. O resto desta divisão é concatenado à mensagem para formar a palavra código. Adicionalmente, conforme pode ser observado em:

$$C(x) = x^{n-k}M(x) + M(x) \bmod G(x) \quad (31)$$

o polinômio $C(x)$ possui essencialmente dois termos. O primeiro corresponde ao polinômio mensagem, que é deslocado $n - k$ posições para esquerda e o segundo termo, corresponde ao resultado da paridade. Deste modo, a palavra código será um múltiplo inteiro do polinômio gerador.

6.3.3 Decodificador Reed-Solomon

Depois de transmitir uma palavra código em um canal de transmissão, os dados podem degradar-se. Deste modo, o polinômio que representa os símbolos da palavra código $C(x)$ modificados pelos erros $E(x)$ somados durante a transmissão, será referenciado a seguir como, $R(x) = C(x) + E(x)$. Este polinômio recebido pode ser escrito como, $R(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \dots + r_1x + r_0$ onde $E(x)$ é o polinômio correspondente aos símbolos de erro.

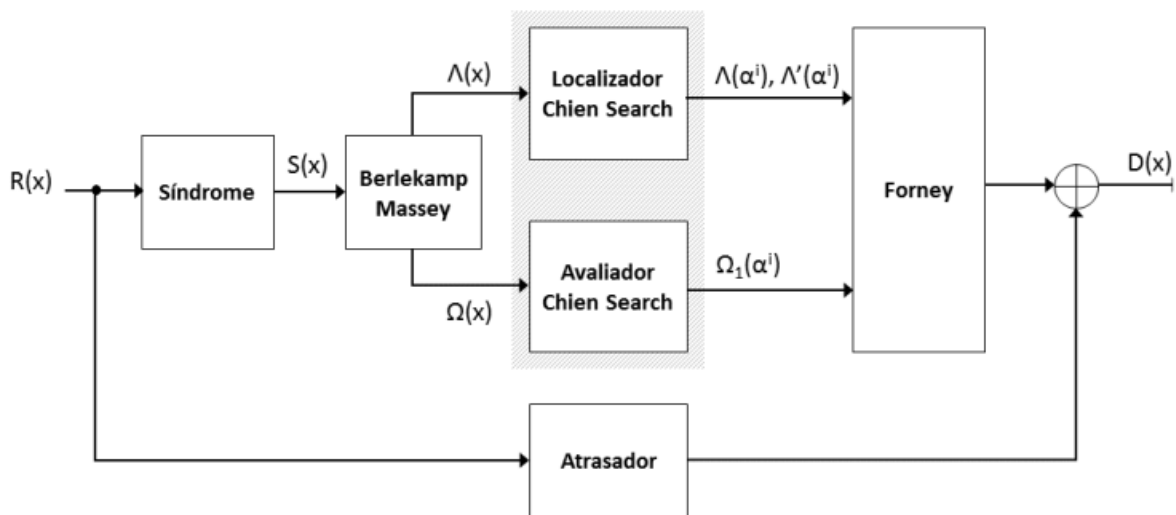
O processo de decodificação R-S consiste em encontrar o polinômio $E(x)$. Para isto, é

necessário que duas informações sejam extraídas do polinômio $R(x)$: a localização dos erros e suas correspondentes magnitudes. Estas localizações e magnitudes constituem o polinômio de erro $E(x)$. Um processo típico de decodificação R-S, compreende as seguintes etapas:

- **Síndrome:** Cálculo do polinômio de síndrome, $S(x)$.
- **Berlekamp Massey:** Cálculo dos coeficientes do polinômio localizador de erros $\Lambda(x)$, e cálculo dos coeficientes do polinômio avaliador de erros $\Omega(x)$
- **Chien Search:** Encontra a posição dos erros calculando as raízes do polinômio localizador de erros, $\Lambda(\alpha^i)$. Gera dois parâmetros intermediários para o cálculo do valor dos erros, $\Lambda'(\alpha^i)$ e $\Omega(\alpha^i)$.
- **Forney Algorithm:** Encontra o valor dos coeficientes do polinômio de erro $E(x)$ a partir dos parâmetros intermediários, $\Lambda'(\alpha^i)$ e $\Omega(\alpha^i)$, calculados no bloco *Chien Search*.
- **Correção:** Corrige os erros do polinômio $C(x)$ através do polinômio de erro $E(x)$.

Na figura 21, é ilustrado um diagrama do processo de decodificação.

Figura 21 - Diagrama em Blocos do Decodificador FEC Reed-Solomon



Fonte: Salvador (2015, p. 39)

6.3.3.1 Computação da Síndrome

A síndrome é o resultado da verificação dos bits de paridade no dado recebido para determinar se o dado recebido é um dado válido. Caso o dado for válido, a síndrome possui valor

0. Qualquer valor diferente de zero indica erros no dado recebido. A síndrome é construída de $n - k$ símbolos, $S_i (i = 1, \dots, n - k)$, portanto há dois símbolos em cada vetor para um código R-S (7,5). Os seus valores podem ser computados pelo sinal recebido. Pela expressão da palavra $U(X) = m(X)g(X)$, pode-se observar que todo $U(X)$ é um múltiplo do polinômio gerador $g(x)$ e todas as raízes de $g(x)$ devem ser raízes de $U(X)$ (SKLAR, 1988b, p. 455).

A mensagem recebida é descrita como sendo $r(x) = U(x) + e(x)$, portanto ao se avaliar as raízes do polinômio gerador na mensagem recebida deve-se obter um resultado nulo caso a palavra recebida não contém erros. Caso o resultado da equação $r(x)$ apresentar um resultado não nulo, houve algum erro na transmissão do dado (SKLAR, 1988b, p. 455). A computação de cada símbolo de síndrome pode ser descrito como:

$$S_i = r(X) |_{X=\alpha^i} = r(\alpha^i), i = 1, \dots, n - k \quad (32)$$

Se $r(x)$ for uma palavra válida, o símbolo de síndrome S_i vai para zero. Considerando que $R(x) = C(x) + E(x)$, os coeficientes de síndrome podem ser definidos como:

$$S_i = C(\alpha^i) + E(\alpha^i), i = 1, 2, 3, \dots, n-1. \quad (33)$$

Como $C(x)$ possui os coeficientes α^j como raízes, ou seja, $C(\alpha^j)$ é sempre 0, a equação de S_i pode ser simplificada e redefinida como:

$$S_i = E(\alpha_i) = \sum_{j=1}^t e_j \alpha^{ij}, i = 1, 2, 3, \dots, n-1, \quad (34)$$

onde e_i corresponde ao valor do erro do j -ésimo erro encontrado na i -ésima posição da palavra código e α_j^i corresponde ao j -ésimo erro encontrado na i -ésima posição da palavra código. O t , como definido anteriormente, corresponde ao número de símbolos que o R-S pode corrigir. Seja μ a quantidade de erros, $0 \leq \mu \leq t$, que estão nas posições i_1, i_2, \dots, i_μ (SALVADOR, 2015). Deste modo, pode-se definir um novo polinômio de erros que pode ser escrito como:

$$E(x) = e_{i1} \alpha^{i1} + e_{i2} \alpha^{i2} + \dots + e_{i\mu} \alpha^{i\mu} \quad (35)$$

Definindo $\alpha_j^i = \beta_i$, a equação 35 pode ser utilizada para gerar $2t$ novas equações a partir dos coeficientes de síndrome diferentes de 0. A seguir são representadas as i , para $0 < i_j \leq 2t$, equações geradas a partir de 35, da seguinte forma:

$$\begin{aligned}
s_1 &= R(\alpha) = E(\alpha) = e_{i1}\beta_1 + e_{i2}\beta_2 + \dots + e_{i\mu}\beta_\mu \\
s_2 &= R(\alpha^2) = E(\alpha^2) = e_{i1}\beta_1^2 + e_{i2}\beta_2^2 + \dots + e_{i\mu}\beta_\mu^2 \\
&\vdots \\
s_{2t} &= R(\alpha^{2t}) = E(\alpha^{2t}) = e_{i1}\beta_1^{2t} + e_{i2}\beta_2^{2t} + \dots + e_{i\mu}\beta_\mu^{2t}
\end{aligned} \tag{36}$$

Portanto, na decodificação existirão no máximo $2t$ equações com $2t$ incógnitas: t valores de erros e t localizações de erros. Entretanto, as $2t$ equações não podem ser resolvidas de modo usual pois são não-lineares (algumas incógnitas possuem expoentes). As técnicas que resolvem este sistema de equações são conhecidas como algoritmos de decodificação R-S. A solução dessas equações é bastante complexa, exigindo que polinômios intermediários sejam computados. Assim, conforme já mencionado, dois novos polinômios são determinados antes que as posições e valores dos erros possam ser encontrados (SALVADOR, 2015). O polinômio localizador de erros é definido como:

$$\Lambda(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \dots + \lambda_\mu x^\mu \tag{37}$$

e o polinômio avaliador de erros é definido como:

$$\omega(x) = 1 + ?_1 x + ?_2 x^2 + \dots + ?_? x^? \tag{38}$$

Existem dois métodos tipicamente utilizados para obter estes dois novos polinômios a partir do polinômio de síndrome:

- Berlekamp Massey
- Euclidiano

Ambos os métodos possuem uma complexidade similar. Entretanto, o algoritmo Berlekamp-Massey apresenta uma eficiência um pouco maior nas operações realizadas (SALVADOR, 2015).

6.3.3.2 Algoritmo Berlekamp-Massey

Uma vez que um vetor de síndromes diferente de zero foi calculado, podemos afirmar que a sequência recebida possui erros. Portanto, é necessário descobrir a localização dos erros. Um polinômio localizador de erros, $\Lambda(X)$, pode ser definido da seguinte maneira (SKLAR, 1988b, p. 457):

$$\Lambda(x) = (1 - \beta_1 x)(1 - \beta_2 x)(1 - \beta_3 x) \dots (1 - \beta_\mu x) \quad \Lambda(x) = 1 + \underset{1}{\Lambda}x + \underset{2}{\Lambda}x^2 + \dots + \underset{\mu}{\Lambda}x^\mu \quad (39)$$

As raízes de $\Lambda(x)$ são $\beta_1^{-1}, \beta_2^{-1}, \dots, \beta_\mu^{-1}$. Logo, o inverso das raízes de $\Lambda(X)$ indicam as localizações de erro de do padrão de erro $e(X)$. Para determinar os coeficientes $\Lambda_1, \Lambda_2, \dots, \Lambda_v$, é necessário utilizar uma técnica denominada modelagem auto-regressiva, que utiliza uma matriz de síndromes, onde as t primeiras síndromes são utilizadas para prever a próxima síndrome, como mostra a equação abaixo (SKLAR, 1988b, p. 457):

$$\begin{bmatrix} S_1 & S_2 & S_3 & \dots & S_{t-1} & S_t \\ S_2 & S_3 & S_4 & \dots & S_t & S_{t+1} \\ & & & \ddots & & \\ S_{t-1} & S_t & S_{t+1} & \dots & S_{2t-3} & S_{2t-2} \\ S_t & S_{t+1} & S_{t+2} & \dots & S_{2t-2} & S_{2t-1} \end{bmatrix} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ \vdots \\ \Lambda_2 \\ \Lambda_2 \end{bmatrix} = \begin{bmatrix} -S_{t-1} \\ -S_{t+2} \\ \vdots \\ -S_{2t-1} \\ -S_t \end{bmatrix} \quad (40)$$

Deve-se encontrar os termos $\Lambda_1, \Lambda_2, \dots, \Lambda_t$ através de técnicas matemáticas que podem serem encontradas em Sklar (1988b, p. 458). A partir dos coeficientes $\Lambda_1, \Lambda_2, \dots, \Lambda_t$, pode-se representar corretamente o polinômio da equação 39. Portanto as raízes do polinômio $\Lambda(x)$ corresponde a uma localização do erro (SKLAR, 1988b, p. 459).

Suponha um polinômio qualquer obtido pela equação 39 é obtido uma raiz na forma α^y . Isto significa que existem erros nas localizações de α^y , em que o índice j é completamente arbitrário. Portanto, pode-se obter o $\beta_l = \frac{1}{\alpha_l^j} = \alpha^y$ e consequentemente a equação 36 (SKLAR, 1988b, p. 459).

O objetivo do algoritmo Berlekamp-Massey é encontrar o mínimo grau do polinômio $\Lambda(x)$ cujos coeficientes satisfazem estas identidades de Newton (SALVADOR, 2015). O algoritmo procede como segue:

- A primeira fase do algoritmo BM visa determinar o grau mínimo do polinômio $\Lambda_{BM}^{(1)}(x)$ que satisfaz a primeira identidade de Newton.
- A segunda identidade de Newton é testada. Se o polinômio $\Lambda_{BM}(x)$ satisfaz a segunda identidade de Newton, então $\Lambda_{BM}^{(2)}(x) = \Lambda_{BM}^{(1)}(x)$. Caso contrário, o algoritmo adiciona um termo de correção em $\Lambda_{BM}^{(1)}(x)$ para o polinômio $\Lambda_{BM}^{(2)}(x)$ seja capaz de satisfazer as duas primeiras identidades de Newton.

- Na k-ésima iteração, o polinômio de grau mínimo será:

$$\bigwedge_{BM}^{(k)}(x) = 1 + \lambda_1^k x + \lambda_2^k x^2 + \dots + \lambda_{l_k}^k x^{l_k} \quad (41)$$

Onde l_k define a ordem do polinômio e $1 \leq l_k \leq k$, para os quais os coeficientes satisfaçam as seguintes identidades:

- o próximo passo consiste em definir um novo polinômio de grau mínimo:

$$\begin{aligned} s_1 + \lambda^{(k+1)} &= 0 \\ s_2 + s_1 \lambda^{(k+1)} &= 0 \\ s_3 + \lambda_1^{(k+1)} s_2 + s_1 \lambda^{(k+1)} &= 0 \\ &\vdots \\ s_{l_{k+1}} + \lambda_1^{(k+1)} s_{l_k} + \lambda_2^{(k+1)} s_{l_{k-1}} + \dots + \lambda_{l_k}^{(k+1)} s_2 + \lambda_{l_{k+1}}^{(k+1)} s_1 &= 0 \end{aligned} \quad (42)$$

Uma vez que o algoritmo BM alcança o *pass* $2t$, o polinômio $\bigwedge_{BM}^{2t}(x)$ é chamado como polinômio localizador de erro $\alpha(x)$.

Para encontrar o polinômio avaliador de erro, uma vez encontrado o polinômio localizador de erros, utiliza-se uma equação que relaciona os polinômios $S(x)$, $\bigwedge(x)$ e $\omega(x)$ (SALVADOR, 2015). Esta equação é chamada de equação chave e é definida como:

$$\bigwedge(x)S(x) = \omega(x) \bmod x^{2t+1} \quad (43)$$

Como os polinômios $S(x)$ e $\bigwedge(x)$ já foram determinados através da primeira etapa do algoritmo BM, a equação chave pode ser rearranjada na forma:

$$\omega(x) = \bigwedge(x)S(x) \bmod x^{2t+1} \quad (44)$$

6.3.3.3 Algoritmo Chien Search

Depois de obter os dois polinômios de localização e avaliação dos erros, precisa-se determinar as raízes do polinômio localizador de erros, as quais representam o inverso da posição dos erros. Portanto, para todos os elementos de $GF(2^m)$ deve-se calcular o valor de $\lambda(\alpha^i)$, onde $i = 1, 2, 3, \dots, 2^{m-1}$ (KAMAR et al., 2017).

Se algum elemento de GF se revelar raiz deste polinômio $\lambda(\alpha^i) = 0$, um erro ocorreu na

posição inversa de i , isto é, em (n_i) . Consequentemente, se esta condição não for satisfeita, significa que não existe erro na palavra código (KAMAR et al., 2017).

6.3.3.4 Algoritmo de Forney

O algoritmo Forney é usado para encontrar o valor dos erros. Para tal, o polinômio $\omega(x)$ possibilita o cálculo do valor do erro através da equação (SALVADOR, 2015):

$$e_{j_l} = \frac{\omega(\beta_l^{-i})}{\Lambda'(\beta^{-i})} \quad (45)$$

Portanto, para um erro na posição α^i , a equação 45 fica:

$$e_{j_l} = \frac{\omega(\alpha_l^{-i})}{\Lambda'(\alpha^{-i})} \quad (46)$$

O polinômio $\Lambda'(x)$ corresponde a derivada do polinômio $\Lambda(x)$ em um ponto x . A derivada de uma função polinomial $f(x)$ pertencente a $GF(2^m)$ é uma função polinomial $f'(x)$ com os seus coeficientes ímpares, conforme observado nas equações (SALVADOR, 2015):

$$f(x) = f_1 + f_2x^2 + f_3x^3 + f_4x^4 + \dots + f_{n-1}x^{n-1} \quad (47)$$

e

$$f'(x) = f_1 + f_3x^3 + f_5x^5 + \dots + f_{n-1}x^{n-1} \quad (48)$$

Portanto, com o valor do erro é possível restaurar o dado somando o erro estimado (x) ao dado recebido. Desta forma, o dado ao final da operação de decodificação do FEC é da forma:

$$(x) = r(x) + (x) = U(x) + e(x) + (x) \quad (49)$$

7 ENTRELAÇADOR (*INTERLEAVING*)

Canais de transmissão que possuem *multipath fading*, *switching noise* e outros tipos de *burst noise* são exemplos de canais com memória. Assumindo que um canal possui memória, não pode-se caracterizar as falhas ocorrendo randomicamente isoladas bit a bit. Apesar de algumas técnicas de codificação para canais com memória terem sido propostas, essas técnicas são difíceis de proporcionar modelos precisos para a variação estatística no tempo desses tipos de canais. Portanto, técnicas de diversidade no tempo ou entrelaçamento resolvem o problema para modelar o canal, uma vez que só necessita-se conhecer o tempo ou o *span* do canal com memória (SKLAR, 1988b, p. 461).

Entrelaçar a mensagem codificada e desentrelaçar depois da recepção causa do espalhamento dos *burst erros* no tempo. Desta forma, o decodificador é capaz de trabalhar na recuperação desses erros caso serem randômicos no tempo. A ideia por trás do entrelaçador é separar os dados no tempo, transformando um canal com memória em um canal sem memória. Desta forma, os códigos corretores de erros são úteis em canais com *burst noise* uma vez que podem trabalhar com dados com erros até a sua capacidade de correção (SKLAR, 1988b, p. 461).

O entrelaçador espalha os símbolos por muitos comprimentos de bloco (para códigos de bloco) ou por muitos períodos de registradores (para códigos convolucionais). O espalhamento requerido é determinado pela duração do *burst erro*. Os detalhes da redistribuição dos bits devem ser conhecidos pelo receptor para realizar o desembaralhamento. São comumente conhecidos e usados dois tipos de entrelaçadores: entrelaçadores de bloco e convolucionais (SKLAR, 1988b, p. 461). Estes dois tipos são explicados nas duas seções decorrentes.

7.1 Entrelaçador de Bloco (*Block Interleaving*)

Um intercalador de bloco aceita os símbolos codificados em blocos do codificador, embaralha os símbolos e, em seguida, alimenta os símbolos reorganizados no modulador. A permutação usual do bloco é realizada preenchendo as colunas de uma matriz M-linha por N-coluna ($M \times N$) com a sequência codificada. Depois que a matriz é completamente preenchida, os símbolos são então introduzidos no modulador linha por linha e transmitidos pelo canal (SKLAR, 1988b, p. 463).

No lado do receptor, o desentrelaçador executa a operação inversa. Ele recebe os símbolos entrelaçados, desentrelaçando-os e os introduzindo no decodificador. Os símbolos são inseridos na matriz do desentrelaçador por linhas e removidos por colunas. A Figura 22 ilustra um

exemplo de um intercalador com $M = 4$ linhas e $N = 6$ colunas. As entradas na matriz ilustram a ordem em que os 24 símbolos de código são colocados no intercalador. A sequência de saída para o transmissor consiste em símbolos de código removidos da matriz por linhas, conforme mostrado na figura. As características mais importantes desse intercalador de bloco são (SKLAR, 1988b, p. 463):

Figura 22 - Exemplo de Entrelaçamento de Bloco.

(a) $M \times N$ entrelaçamento de bloco. (b) Cinco símbolos de *burst error*. (c) Nove símbolos de *burst error*. (d) Erro único periódico espaçado sequencialmente $N = 6$ símbolos separados.

$N = 6$ columns

$M = 4$ rows

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

Interleaver
output sequence: 1, 5, 9, 13, 17, 21, 2, 6, ...

(a)

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

(b)

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

(c)

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

(d)

Fonte: Sklar (1988b, p. 464)

- Qualquer *burst error* com de N erros de símbolo de canal contíguo resulta em erros isolados na saída do desentrelaçador que são separados um do outro por pelo menos M

símbolos.

- Qualquer *burst error* bN , em que $b > 1$, resulta em *bursts errors* de saída do desentrelaçador de não mais do que $[b]$ símbolos de erros. Cada rajada de saída é separada das outras rajadas por símbolos $M - [b]$. A notação $[x]$ significa o menor número inteiro não inferior a x , e $\lceil x \rceil$ significa o maior número inteiro não maior que x .
- O atraso de ponta a ponta do entrelaçador/desentrelaçador é de aproximadamente $2MN$. Para ser preciso, apenas as células de memória $M(N - 1) + 1$ precisam ser preenchidas antes que a transmissão possa começar (assim que o primeiro símbolo da última coluna da matriz $M \times N$ for preenchido). Um número correspondente precisa ser preenchido no receptor antes do início da decodificação. Assim, o atraso mínimo de ponta a ponta é de $(2MN - 2M + 2)$ tempos de símbolo, sem incluir nenhum atraso de propagação de canal.
- O requisito de memória são símbolos MN para cada local (entrelaçador e desentrelaçador). No entanto, como a matriz $M \times N$ precisa ser preenchida (principalmente) antes que possa ser lida, geralmente é implementada uma memória de símbolos $2MN$ em cada local para permitir o esvaziamento de uma matriz $M \times N$ enquanto a outra está sendo preenchida, e vice-versa.

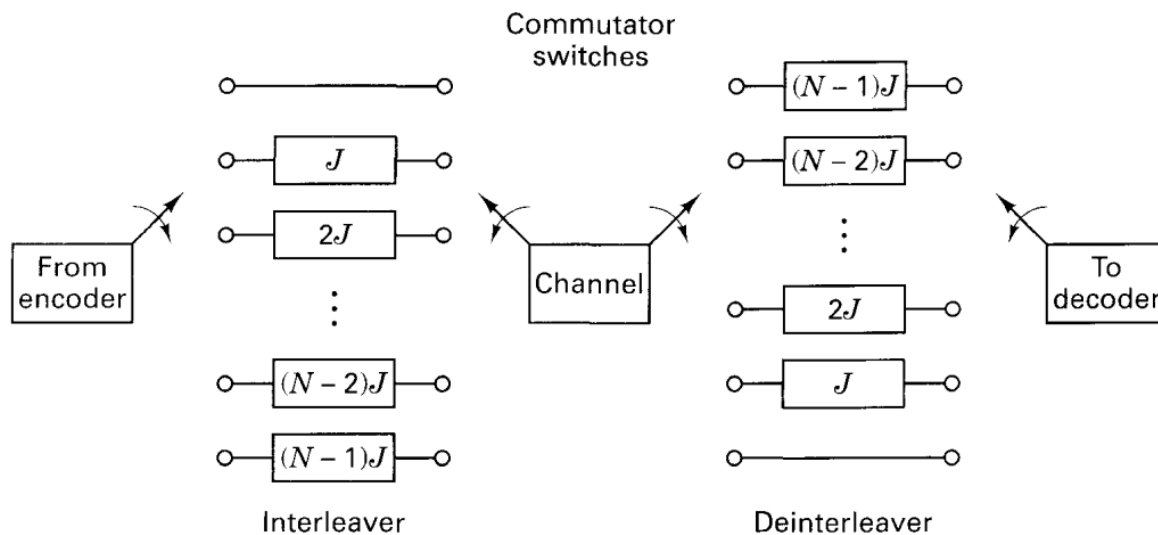
Normalmente, para uso com um código de correção de erro único, os parâmetros do intercalador são selecionados de modo que o número de colunas N ultrapasse o comprimento do *burst error*. A escolha do número de linhas M depende do esquema de codificação usado. Para códigos de bloco, M deve ser maior que o comprimento do bloco de código. Para códigos convolucionais, M deve ser maior que o comprimento da restrição. Assim, uma rajada de comprimento N pode causar no máximo um erro único em qualquer palavra de código de bloco: da mesma forma, com códigos convolucionais, haverá no máximo um erro único em qualquer comprimento de restrição de decodificação. Para códigos de correção de erros t , a escolha de N precisa apenas ultrapassar o comprimento esperado do *burst errors* dividido por t (SKLAR, 1988b, p. 464).

7.2 Entrelaçador Convolutacional (Convolutional Encoder)

Uma estrutura de um entrelaçador é ilustrada na Figura 23. Os símbolos de código são deslocados sequencialmente no banco de N registros. Cada registro sucessivo fornece J símbolos mais armazenamento do que o anterior. O registro zero não fornece armazenamento (o símbolo é transmitido imediatamente). Com cada novo símbolo de código, o comutador muda para um novo registro, e o novo símbolo de código é deslocado enquanto o símbolo de código mais antigo desse registro é deslocado para o modulador/transmissor. Após o registro $(N - 1)$, o comutador retorna ao registro zero e inicia novamente. O desentrelaçador executa a operação

inversa, e os comutadores de entrada e saída para serem entrelaçados e desentrelaçados devem ser sincronizados (SKLAR, 1988b, p. 466).

Figura 23 - Implementação de um Entrelaçador Convolutacional com *Shift Register*



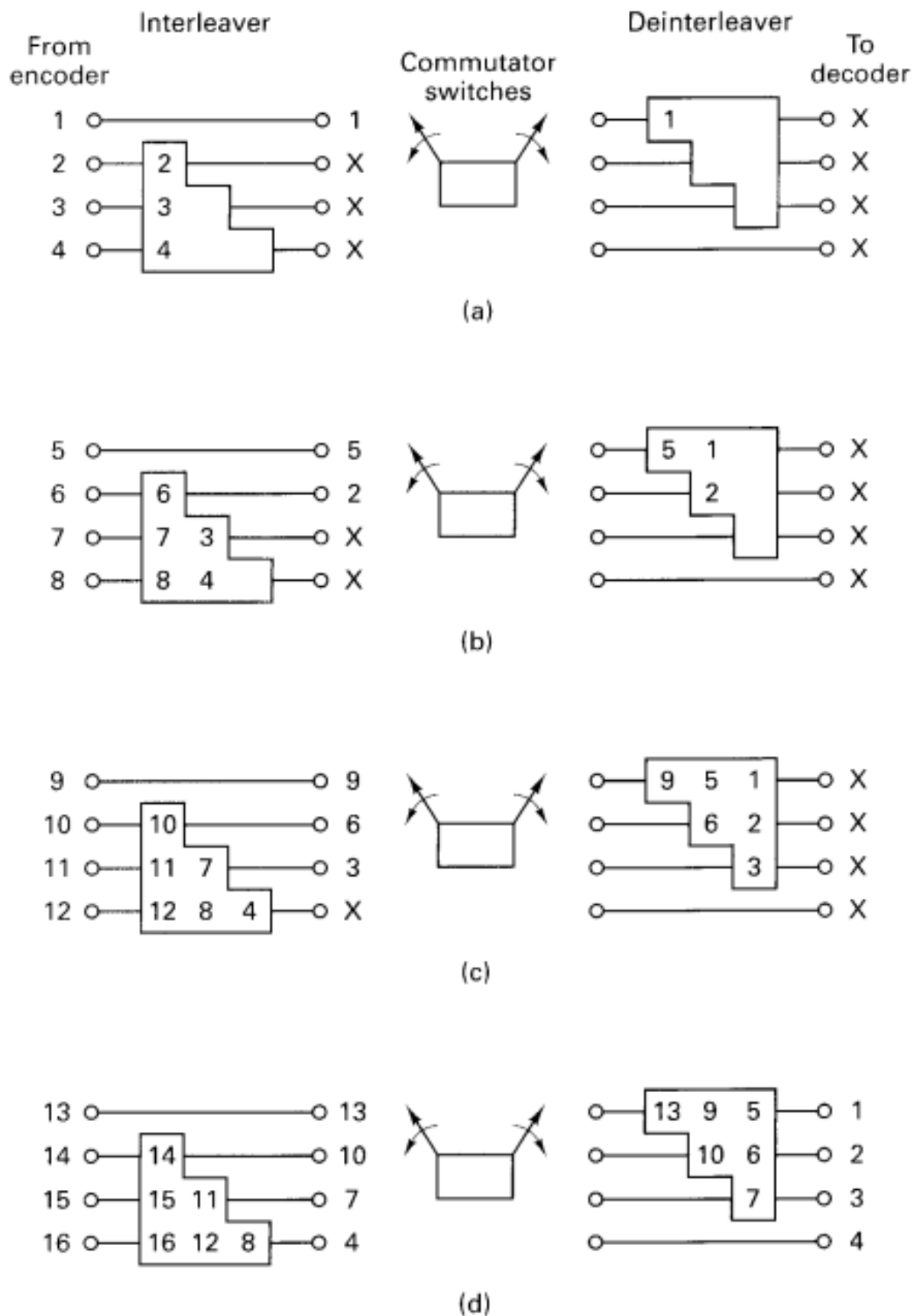
Fonte: Sklar (1988b, p. 466)

A Figura 24 ilustra um exemplo de um entrelaçador convolutacional simples de quatro registros ($J = 1$) sendo carregado por uma sequência de símbolos. O desentrelaçador sincronizado é mostrado inserindo simultaneamente os símbolos desentrelaçados no decodificador. A Figura 24(a) mostra os símbolos 1 a 4 sendo carregados. Os x' s representam estados desconhecidos (SKLAR, 1988b, p. 468).

A Figura 24(b) mostra os quatro primeiros símbolos deslocados nos registros e os símbolos 5 a 8 na entrada do entrelaçador. A Figura 24(c) mostra os símbolos 9 a 12 entrando no entrelaçador. O desentrelaçador agora está preenchido com símbolos de mensagem. Entretanto, nada útil ainda está sendo fornecido ao decodificador. Finalmente, a Figura 24(d) mostra símbolos 13 a 16 sendo inseridos no entrelaçador e na saída do desentrelaçador. Os símbolos 1 a 4 estão sendo passados para o decodificador (SKLAR, 1988b, p. 468).

O processo continua dessa maneira até toda a sequência de palavras de código, em sua forma pré-intercalada original. O desempenho de um entrelaçador convolutacional é muito semelhante ao de um entrelaçador de blocos. A vantagem importante da entrelaçação convolutacional sobre blocos é que, com a entrelaçação convolutacional, o atraso de ponta a ponta são símbolos $M(N - 1)$, onde $M = NJ$, e a memória necessária é $\frac{M(N-1)}{2}$ nas duas extremidades do canal. Portanto, há uma redução de metade no atraso e na memória sobre os requisitos de intercalação de blocos (SKLAR, 1988b, p. 468).

Figura 24 - Computação de um Entrelaçador Convolutivo



8 RESULTADOS E DISCUSSÕES

Neste capítulo será apresentado os resultados...

9 CONCLUSÕES

Conclui-se que...

REFERÊNCIAS

- ABINAYA, N. S.; PRAKASAM, P. Performance analysis of maximum length lfsr and bbs method for cryptographic application. In: *2014 International Conference on Electronics and Communication Systems (ICECS)*. [S.l.: s.n.], 2014. p. 1–5.
- AGGLETON, R.; ARDILA-PEREZ, L.; BALL, F.; BALZER, M.; BOUDOUL, G.; BROOKE, J.; CASELLE, M.; CALLIGARIS, L.; CIERI, D.; CLEMENT, E.; DUTTA, S.; HALL, G.; HARDER, K.; HOBSON, P.; ILES, G.; JAMES, T.; MANOLOPOULOS, K.; MATSUSHITA, T.; MORTON, A.; NEWBOLD, D.; PARAMESVARAN, S.; PESARESI, M.; POZZOBON, N.; REID, I.; ROSE, A.; SANDER, O.; SHEPHERD-THEMISTOCLEOUS, C.; SHTIPLIYSKI, A.; SCHUH, T.; SKINNARI, L.; SUMMERS, S.; TAPPER, A.; THEA, A.; TOMALIN, I.; UCHIDA, K.; VICHODIS, P.; VIRET, S.; WEBER, M. An FPGA based track finder for the l1 trigger of the CMS experiment at the high luminosity LHC. *Journal of Instrumentation*, IOP Publishing, v. 12, n. 12, p. P12019–P12019, dec 2017. Disponível em: <<https://doi.org/10.1088>>
- AGRAWAL, M. *Implementation of reed solomon error correcting codes*. 48 p. Dissertação (Bachelor of Technology in Electronics and Communication Engineering) — National Institute of Technology, Rourkela, Odisha, India, 2011.
- ASSAAD, M. *Design and Modelling of Clock and Data Recovery Integrated Circuit in 130 nm CMOS Technology for 10 Gb/s Serial Data Communications*. Tese (PhD) — University of Glasgow, College of Science and Engineering, School of Engineering, Glasgow, 2009. Disponível em: <<http://theses.gla.ac.uk/id/eprint/707>>.
- BRÜNING, L. R. *HIGH LUMINOSITY LARGE HADRON COLLIDER A DESCRIPTION FOR THE EUROPEAN STRATEGY PREPARATORY GROUP*. 2019. Disponível em: <<https://cds.cern.ch/record/1471000/files/CERN-ATS-2012-236.pdf>>. Acesso em: 17 set. 2019.
- COLLABORATION, C. *Technical Proposal for the Phase-II Upgrade of the Compact Muon Solenoid*. 2019. Disponível em: <<http://www.desy.de/~garutti/LECTURES/ParticleDetectorSS12/JournalClub/lhc-CMS.pdf>>. Acesso em: 20 set. 2019.
- DHINGRA, S. *Comparison of LFSR and CA for BIST*. Auburn: Dept. of Electrical and Computer Engineering, 2019. 47 p. Disponível em: <http://www.eng.auburn.edu/~agrawvd/COURSE/E7250_05/REPORTS_TERM/Dhingra_LFSR.pdf>. Acesso em: 19 set. 2019.
- FARRELL, J. C.; MOREIRA, P. G. *Essentials of Error-Control Coding*. Upper Saddle River: John Wiley & Sons, 2006.
- FERREIRA, B. C. *DETECÇÃO DE RAIOS CÓSMICOS COM CALORIMETRIA DE ALTAS*

ENERGIAS. 118 p. Dissertação (Mestrado em Engenharia Elétrica) — UNIVERSIDADE FEDERAL do Rio de Janeiro, Rio de Janeiro, 2009.

FLOYD, T. L. *Digital Fundamentals*. New Delhi: Pearson, 2009. 497 p.

GARVER, L. L. Transmission linear programming. *IEEE Transactions on Power Apparatus and Systems*, Rio de Janeiro, PAS-29, n. 9, p. 168–197, Dec. 1970.

GHATAK, A.; THYAGARAJAN, K. *An Introduction to Fiber Optics*. [S.l.]: Cambridge University Press, 1998.

GUPTA, V.; VERMA, D. C. Error detection and correction: An introduction. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 2, n. 11, p. 212 – 218, Nov. 2012.

HASSAN, G. M. *Scramble Image Based on LFBSR (Linear Feedback Shift Registers)*. Baghdad: Dept. of Electrical and Computer Engineering, 2019. 14 p. Disponível em: <<https://www.iasj.net/iasj?func=fulltext&aId=58271>>. Acesso em: 20 set. 2019.

Huang, S.; Zhang, Z. Principles of fecs with evaluating different types of fec used in the internet and wireless networks. In: *2011 International Conference on Electronics, Communications and Control (ICECC)*. [S.l.: s.n.], 2011. p. 2181–2184.

KAMAR, S.; FOUADA, A.; ZEKRY, A.; EL-MAHDY, A. Fpga implementation of rs codec with interleaver in dvb-t using vhdl. *International Journal of Engineering & Technology*, v. 6, 11 2017.

KERL, J. *Computation in finite fields*. [s.n.], 2004. Disponível em: <<http://johnkerl.org/doc/ffcomp.pdf>>. Acesso em: 20 setembro 2018.

LI, Z. *DESIGN OF SERIALIZER AND DESERIALIZER OPERATING IN 65 nm CMOS TECHNOLOGY FOR HIGH-SPEED SERIAL LINK (HSSL) APPLICATIONS*. 31 p. Dissertação (Senior Thesis in Electrical Engineer) — University of Illinois, Urbana-Champaign, Illinois, EUA, 2015.

MACHADO, F. B.; MAIA, L. P. *Arquitetura de Sistemas Operacionais*. 4. ed. Rio de Janeiro: LTC, 2007. 105 p. p.

MOBS, E. The cern accelerator complex. complexe des accélérateurs du cern. Jul 2016. General Photo. Disponível em: <<https://cds.cern.ch/record/2197559>>.

MORENO, R. L. *Implementação em FPGA de uma Arquitetura Reed-Solomon para Uso em Comunicações Ópticas*. 68 p. Dissertação (Mestrado em Automação e Sistemas Elétricos Industriais) — UNIVERSIDADE FEDERAL DE ITAJUBÁ, UNIFEI, Itajubá - MG, 2010.

ORFANELLI, S. Pixel detector for cms upgrade. In: *Proceedings of Science, The 27th International Workshop on Vertex Detectors (VERTEX2018)*. [S.l.: s.n.], 2019. v. 348, n. 021.

RANDALL, L. *BATENDO À PORTA DO CÉU: O BÓSON DE HIGGS E COMO A FÍSICA MODERNA ILUMINA O UNIVERSO*. São Paulo: Companhia das Letras, 2013. 576 p.

RATAN, R. *DESIGN OF A PHASE LOCKED LOOP BASED CLOCKING CIRCUIT FOR*

HIGH SPEED SERIAL LINK APPLICATIONS. 124 p. Dissertação (Master of Science in Electrical and Computer Engineering) — University of Illinois, Urbana-Champaign, Illinois, EUA, 2014.

SALVADOR, A. H. *Implementação em VHDL de uma arquitetura paralela de um código de Reed-Solomon aplicado a Redes OTN*. 92 p. Dissertação (Dissertação(mestrado)) — Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e Computação, UNICAMP, Campinas - SP, 2015. Disponível em: <<http://repositorio.unicamp.br/handle/REPOSIP/258899>>.

SILVA, D. C. C. e. *Estudo da Codificação de Rede e Análise do seu Desempenho com Fonte de Tráfego HTTP*. 98 p. Dissertação (Mestrado em Telecomunicações) — Instituto Nacional de Telecomunicações, INATEL, Santa Rita do Sapucaí - MG, 2011.

SKLAR, B. *Digital Communications: Fundamentals and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN 0-13-211939-0.

SKLAR, B. *Digital Communications: Fundamentals and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN 0-13-211939-0.

SPRACE. *Sprace research group*. São Paulo: [s.n.], 2019. Disponível em: <<http://sprace.org.br>>. Acesso em: 15 set. 2018.

TEAM, L. D. *lpGBT Documentation*. [S.l.], 2019. Disponível em: <http://padley.rice.edu/cms/OH_GE21_LpGBT/lpGBT_20190215.pdf>. Acesso em: 20 Set. 2019.

WIKIPÉDIA. *GRANDE COLISOR DE HÁDRONS*. 2019. Disponível em: <https://pt.wikipedia.org/wiki/Grande_Colisor_de_Hádrons>. Acesso em: 15 set. 2019.

APÊNDICE A - LINUX

Neste capítulo será abordado o surgimento e a evolução do sistema operacional Linux. (GARVER, 1970).

APÊNDICE A.1 - HISTÓRICO DO LINUX

Atualmente, ...

Segundo Machado e Maia (2007), o sistema operacional (SO), possui inúmeras funções, as quais podem ser resumidas em duas:

- **Facilidade de acesso aos recursos:** consiste em ser totalmente transparente ao usuário a maneira como funciona um computador paralelo , ou seja, para um usuário não importa como um arquivo que está em um disquete será lido, mas sim que o mesmo será lido, resumindo, um usuário não precisa saber como será realizado essa ação e suas inúmeras etapas;

Figura 25 - Ilustração.



Fonte: Adaptado de Machado e Maia (2007)

APÊNDICE B - AINDA FALANDO DO LINUX

Neste capítulo será abordado o surgimento e a evolução do sistema operacional Linux.

APÊNDICE B.1 - MELHORIAS PARA O LINUX EM UM AMBIENTE CORPORATIVOS DE DUAS GRANDES FRNTES INTERPRETATIVAS

Atualmente, ...

- **Facilidade de acesso aos recursos:** consiste em ser totalmente transparente ao usuário a maneira como funciona um computador, ou seja, para um usuário comum não importa como um arquivo que está em um disquete será lido, mas sim que o mesmo será lido, resumindo, um usuário não precisa saber como será realizado essa ação e suas inúmeras etapas. (MACHADO; MAIA, 2007);

Figura 26 - Novo sistema operacional.



Fonte: Machado e Maia (2007)

Para facilitar a vida dos usuários, um exemplo de tabela longa.

Tabela 3 - Espaço de busca combinatório reduzido (*EBCR*) de 10, 5, 3 e 2 soluções com *gap* de 5% Para IEEE

Ramos	Número Máximo de linhas									
	poolreplace=0		poolreplace=1				poolreplace=2			
	5 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.
n_{1-2}	3	1	3	4	2	1	4	3	2	0
n_{1-3}	0	0	0	0	0	0	0	0	0	0

continua.

Tabela 3 - (Continuação da tabela da página anterior)

Ramos	Número Máximo de linhas									
	poolreplace=0		poolreplace=1				poolreplace=2			
	5 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.
n_{1-5}	1	1	1	1	1	1	1	1	1	1
n_{2-4}	0	0	0	0	0	0	0	0	0	0
n_{2-6}	0	0	0	0	0	0	0	0	0	0
n_{3-9}	0	0	0	0	0	0	0	0	0	0
n_{3-24}	1	1	1	1	1	1	1	1	1	1
n_{4-9}	0	0	0	0	0	0	0	0	0	0
n_{5-10}	0	0	0	0	0	0	0	0	0	0
n_{6-10}	1	1	1	1	1	1	1	1	1	1
n_{7-8}	3	2	3	2	3	3	2	3	2	3
n_{8-9}	0	0	0	0	0	0	0	0	0	0
n_{8-10}	0	0	0	0	0	0	0	0	0	0
n_{9-11}	0	0	0	0	0	0	0	0	0	0
n_{9-12}	0	0	0	0	0	2	0	0	0	0
n_{10-11}	1	0	1	0	1	1	0	1	0	1
n_{10-12}	1	1	1	1	1	1	1	1	1	1
n_{11-13}	1	1	1	1	1	1	1	1	1	1
n_{11-14}	0	0	0	0	0	1	0	0	0	0
n_{12-13}	0	0	0	0	0	0	0	0	0	0
n_{12-23}	0	0	0	0	0	0	0	0	0	0
n_{13-23}	0	0	0	0	0	0	0	0	0	0
n_{14-16}	1	1	1	1	1	1	1	1	1	1
n_{15-16}	0	0	0	0	0	0	0	0	0	0
n_{15-21}	0	0	0	0	0	0	0	0	0	0
n_{15-24}	0	0	0	0	0	0	0	0	0	0
n_{16-17}	0	0	0	0	0	0	0	0	0	0
n_{16-19}	0	0	0	0	0	0	0	0	0	0
n_{17-18}	0	0	0	0	0	0	0	0	0	0
n_{17-22}	0	0	0	0	0	0	0	0	0	0
n_{18-21}	0	0	0	0	0	0	0	0	0	0
n_{19-20}	0	0	0	0	0	0	0	0	0	0
n_{20-23}	1	1	1	1	1	1	1	1	1	1
n_{21-22}	0	0	0	0	0	0	0	0	0	0
n_{1-8}	0	0	0	0	0	0	0	0	0	0
n_{2-8}	0	0	0	0	0	1	0	0	0	0
n_{6-7}	0	0	0	0	0	2	0	0	0	0
n_{13-14}	0	0	0	0	0	1	0	0	0	0
n_{14-23}	1	0	1	0	1	1	0	1	0	1
n_{16-23}	0	0	0	0	0	0	0	0	0	0
n_{19-23}	0	0	0	0	0	0	0	0	0	0

continua.

Tabela 3 - (Continuação da tabela da página anterior)

Ramos	Número Máximo de linhas									
	poolreplace=0		poolreplace=1				poolreplace=2			
	5 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.	10 sol.	5 sol.	3 sol.	2 sol.
F.O	220.28	220.28	220.28	220.28	220.28	220.28	220.28	220.28	220.2	220.2

Fonte: Dados da pesquisa do autor.

Fim.

ÍNDICE REMISSIVO

computador, 17, 26
paralelo, 26

usuário, 26, 27
comum, 27