Bruno Fernandes Chimieski
Jonas Cella
Tiago Baptista Noronha

Implementação e Otimização do Sistema Corretor de Erros Baseado em Lógica Majoritária Para Codificação de Canal no Sistema ISDB-T de TV Digital

Porto Alegre – RS

11 de dezembro de 2009

Bruno Fernandes Chimieski
Jonas Cella
Tiago Baptista Noronha

Implementação e Otimização do Sistema Corretor de Erros Baseado em Lógica Majoritária Para Codificação de Canal no Sistema ISDB-T de TV Digital

Trabalho de conclusão de curso a ser apresentado à Faculdade de Engenharia da Pontifícia Universidade Católica do Rio Grande do Sul para a obtenção do título de Engenheiro de Computação.

Orientador:
Prof. Dr. Fabrício de Oliveira Ourique

Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Engenharia

Porto Alegre – RS

11 de dezembro de 2009

Agradecimentos

Aos meus pais, que sempre me apoiaram nas minhas decisões, tanto nos estudos quanto na vida. Ao orientador e aos colegas de curso e de trabalho, que ajudaram dando novas idéias de resolver os problemas, sentando e tomando os problemas como se fossem deles para ajudar na solução, criticando construtivamente e contribuindo para que o curso e os trabalhos não fossem tão penosos. Aos amigos e familiares, dispostos à comemorar as alegrias e vitórias, porém sempre presentes para aturar as queixas e angústias ao longo do curso.

Jonas Cella

Agradeço, a Deus pelo dom da vida e pela dádiva de ter conhecido pessoas tão especiais. Aos meus pais por todo apoio e dedicação, e por terem me ensinado as primeiras lições. A todos meus familiares por acreditarem em meus sonhos e pelo apoio prestado em minha caminhada. Em especial aos meus afilhados João Vítor, Humberto e Larissa Noronha por me motivarem a acreditar no futuro. Agradeço com todo coração a minha namorada, Susana de Oliveira Elias, por estar sempre ao meu lado me dando força e ajudando a superar todos os problemas do meu dia-a-dia, e compartilhando todos os momentos de alegria. Agradeço também ao grupo do CPTW, principalmente à equipe DVBENTER, meus amigos Bruno Hexsel e Diego von Brixen Montzel Trindade, por todo coeficiente de gaiatagem inserido em minha vida. E por fim, aos co-autores desta obra e colegas de curso, o meu muito obrigado pela paciência e dedicação prestadas.

Tiago Baptista Noronha

Gostaria de agradecer primeiramente a meus pais que com imensurável carinho e suporte me apoiaram ao longo dessa extenuante porém gratificante jornada. Sem eles eu nada seria! Agradeço aos meus colegas de trabalho, Jonas Cella e Tiago Baptista Noronha, pelo empenho demonstrado, pois sem isso não teríamos chegado à conclusão do presente trabalho. À equipe do Centro de Pesquisas em Tecnologias Wireless por depositarem sua confiança em nossos esforços para atingir esse objetivo. Ao professor, Dr. Fabrício de Oliveira Ourique, por sua fundamental orientação ao longo do desenvolvimento deste trabalho. Aos demais colegas de curso pela união e camaradagem demonstrados ao longo do curso. Por fim, agradeço à Deus por todas as bençãos recebidas ao longo de toda a minha vida. Somente graças a ele foi possível a conclusão de mais essa etapa importante em minha vida.

Bruno Fernandes Chimieski

"Se não receio o erro, é só porque estou sempre pronto a corrigi-lo."

Bento Jesus Caraça

Resumo

O presente trabalho apresenta a implementação e otimização de um sistema de correção de erros baseado em lógica majoritária para o padrão ISBD-T de TV digital. A codificação de canal em questão é utilizada no sistema com o intuito de proteger os dados do TMCC, que servem para informar os parâmetros de transmissão ao receptor. Adicionalmente, é proposta uma otimização no receptor com a adição de blocos de pré e pós-processamento com o objetivo de aumentar a robustez e tolerância a falhas do sistema. Os resultados obtidos demonstram que o código corretor funciona conforme as regras contidas na norma ARIB-B31, e que suas otimizações não interferiram significativamente na área de FPGA consumida e no tempo de processamento e, além disso, em alguns casos, fizeram com que o código corretor pudesse corrigir mais erros do que sua capacidade teórica.

Palavras-chave: VHDL, Código Corretor de Erros Cíclico, ISDB-T, Lógica Majoritária, TV Digital, Codificação de Canal, TMCC, FPGA.

Abstract

This article presents an implementation and optimization of an error correction system based on a majority logic decision for the ISDB-T digital TV broadcasting system, which was recently adopted in Brazil as digital television standard. The channel coding is used in order to protect TMCC data which is used to inform transmission parameters to the receiver. Furthermore, it is proposed a receiver optimization through the addition of pre and post processing blocks on the receiver side in order to execute procedures to increase reliability and fault tolerance within the system. Obtained results show that the error correcting code works as its definition at japanese regulamentation (ARIB-B31), and its optimizations did not affect substantially the FPGA area consumption and timming constraints, and besides it has been observed that the correction system was able to correct more errors than its theoric capability in some cases.

Keywords: VHDL, Cyclic Errors Corrector Code, Majority Logic, ARIB, ISDB-T, Digital TV, Channel Coding, TMCC, FPGA.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

- ARIB - *Association of Radio Industries and Business*
- ATSC - *Advanced Television System Commitee*
- AWGN - *Additive White Gaussian Noise*
- BCH - *Bose-Chaudhuri-Hochquenghem*
- BER - *Bit Error Rate*
- CRC - *Cyclic Redundancy Check*
- DVB-T - *Digital Video Broadcasting - Terrestrial*
- FPGA - *Field Programmable Gate Array*
- ISDB-T - *Integrated Services Digital Broadcasting - Terrestrial*
- NHK - em japonês, *Nippon Hoso Kyokai*, em inglês, *Japan Broadcasting Corporation*
- MPEG - *Moving Picture Experts Group*
- OFDM - *Orthogonal Frequency-Division Multiplexing*
- SBTVD - Sistema Brasileiro de Televisão Digital
- TCC - Trabalho de Conclusão de Curso
- TMCC - *Transport and Multiplexing Configuration Control*
- TV - Televisão
- VHDL - *Very-high-speed integrated circuits Hardware Description Language*

1 Introdução

Desde o início da humanidade o homem sente a necessidade de se comunicar, primeiramente pela voz, depois pela escrita. Ao longo da evolução outros meios foram surgindo, porém, sempre calçados na base fundamental, a fala e a escrita. A imprensa, o rádio e a televisão são todos meios de comunicação criados pelo ser humano para, de uma forma ou de outra, se comunicar.

Como consequência, na década de 20 surgiram as primeiras televisões, e desde então nunca pararam de evoluir. Consideradas a extensão natural do rádio, pois acrescentam imagem à voz, em menos de um século foram transformadas, de grandes caixas com imagens em preto e branco e baixissíma definição, em dispositivos móveis que cabem no bolso de uma calça e exibem conteúdos com média e alta resolução (RESENDE, 2004).

Existem diversos fatores que contribuíram para essa evolução dos meios de comunicação, e entre eles, talvez o mais importante seja a evolução dos circuitos e componentes eletrônicos. Hoje, com algumas centenas de dólares, é possível comprar um computador pessoal que possui mais velocidade de processamento, memória e capacidade de armazenamento do que um computador comprado em 1965 por 1 milhão de dólares (HENNESSY; PATTERSON, 2003).

Devido às revoluções na indústria de semicondutores e meios de transmissão, hoje temos à disposição inúmeros meios de comunicação. Como exemplo, podemos citar a televisão digital, que é alvo de muitos estudos dada a vasta gama de aplicações, programações e serviços que podem ser diponibilizados aos usuários. Porém, para que o estágio atual pudesse ser alcançado, diversas tecnologias tiveram que ser desenvolvidas e aperfeiçoadas. Padrões de digitalização e compressão de vídeo e voz, tecnologias de codificação de canal e transmissão, métodos de correção de erros foram e continuam sendo estudados e desenvolvidos para atender os requisitos mínimos propostos pelos sistemas de TV digital(RESENDE, 2004).

No campo dos códigos corretores de erros podemos citar diversos tipos para diver-

sas utilidades, que variam desde aplicações espaciais até controle de erros em ambientes industriais. No sistema de recepção de TV digital japonês (ISDB-T), existe o código que será alvo deste estudo. O sistema japonês foi adotado pelo Brasil como base para o Sistema Brasileiro de TV Digital (SBTVD). O SBTVD difere do ISDB-T quanto à compressão digital de áudio e vídeo, pois utiliza o padrão MPEG-4 (*Moving Picture Experts Group*) ao invés do MPEG-2, utilizado no ISDB-T(DTV, 2009). Apesar de utilizarem tipos de compressão diferentes, a camada física do sistema, onde o código corretor de erros implementado neste trabalho se encontra, não sofreu alterações.

As otimizações aqui apresentadas não interferem nos demais blocos do sistema, uma vez que o código corretor de erros possui velocidade de processamento muito maior do que a do bloco TMCC, que é o bloco responsável por repassar as informações à serem corrigidas ao código corretor e retirar a palavra corrigida no término da computação. Essa diferença entre as velocidades de processamento garante que, mesmo acrescentando outros blocos ao sistema de correção, o tempo máximo para o cálculo será atendido.

2 Aspectos Teóricos

O presente capítulo destina-se a fornecer o embasamento teórico necessário para o entendimento posterior dos algoritmos implementados e das otimizações do corretor de erros alvo desse trabalho. Ou seja, trata de assuntos relacionados com o sistema ISBD-T, codificação de canal e teoria de erros. Nesse capítulo também é feita uma breve introdução sobre os blocos que compõem um sistema básico de comunicação digital.

2.1 Sistema de Comunicação Digital

Embora a comunicação digital, ou também chamada de transmissão digital, se refira à transmissão de informação que se encontra na forma digital, não significa que apenas informação gerada nessa forma possa se utilizar de um sistema de transmissão digital. Pode-se usar modulação digital em sinais produzidos em forma analógica, como por exemplo, dados, áudio e vídeo.

Uma das principais vantagens no uso de técnicas de transmissão digital é devido ao fato de as informações serem representadas na forma binária, o que possibilita o uso de técnicas computacionais executadas por microprocessadores. Essas técnicas são chamadas de Processamento Digital de Sinais, e oferecem uma vasta gama de aplicações, como filtragens, cancelamento de interferências, cancelamento de ruído, modulação e outros processamentos (RUSCHEL, 1996).

A Figura 1 mostra um sistema de comunicação digital básico. Dentro de cada bloco da figura existem outros n blocos que executam tarefas de acordo com o propósito geral do sistema.

Entretanto, apesar de parecerem bastante genéricos e poderem ser implementados de diversas maneiras, as funcionalidades básicas são as seguintes:

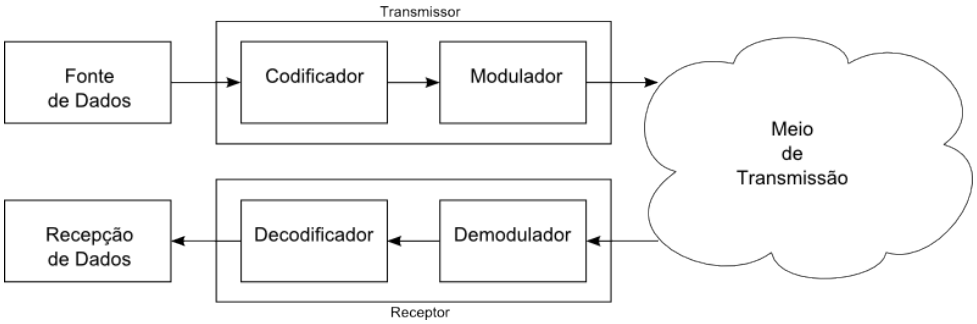

Figura 1: Sistema Digital Básico.

2.1.1 Fonte de Dados

Fornece os dados para o sistema. Consiste em qualquer fonte de informação que se deseja, de alguma forma, codificar e transmitir. Essas informações podem estar tanto no formato digital quanto no formato analógico.

2.1.2 Codificador

O codificador transforma a mensagem em símbolos capazes de serem computados por microprocessadores. Essas mensagens podem ser o resultado de uma conversão analógica-digital como no caso de uma transmissão de voz.

2.1.3 Modulador

O modulador mapeia a informação no formato digital, ou seja, os símbolos, em sinais que podem ser eficientemente transmitidos pelo canal de comunicação (WICKER, 1994). Em outras palavras, o modulador se encarrega de representar os símbolos por sinais elétricos adequados para o meio de transmissão escolhido (RUSCHEL, 1996).

2.1.4 Meio de Transmissão

É o meio físico através do qual a informação irá trafegar até chegar ao destino. Pode-se citar diversos meios, como por exemplo, ar, fibra óptica, cabos metálicos etc. Cada meio de transmissão possui características próprias e que podem ser um fator de peso na implementação de uma aplicação.

2.1.5 Demodulador

Faz o trabalho inverso do modulador, convertendo os sinais analógicos em informações digitais que poderão ser manipuladas por microprocessadores.

2.1.6 Decodificador

Será o elemento responsável pela produção de um sinal digital regenerado a partir do sinal distorcido entregue pelo meio de transmissão (RUSCHEL, 1996).

2.1.7 Recepção de Dados

É o ponto final na cadeia de comunicação. É nesse ponto onde as informações transmitidas são efetivamente utilizadas. Pode ser um sistema de televisão ou rádio digital, uma rede de computadores *wireless* ou cabeada, ou seja, qualquer aplicação que transmita dados de um ponto à outro.

2.2 Sistema ISDB-T

Depois de explicados os elementos principais de um sistema de comunicação, trata-se nesse capítulo do sistema de comunicação no qual está inserido o desenvolvimento desse trabalho, o sistema de transmissão de TV digital baseado na norma japonesa, o ISDB-T. Esse sistema, já adotado no Brasil e em outros países da América Latina, teve suas pesquisas iniciadas na década de 70 nos laboratórios da rede de televisão japonesa NHK, usando como base o sistema europeu de transmissão, o DVB-T. Portanto, o ISDB-T, assim como o DVB-T utiliza a técnica de modulação OFDM, que demonstra uma melhor ocupação espectral em relação aos sistemas de portadora única.

O ISDB-T possui modificações em relação ao DVB-T que o tornam mais robusto e adaptável, uma vez que é possível escolher entre mais de um milhar de esquemas de configuração, dependendo das necessidades da emissora. Para tanto, é preciso enviar quais são as configurações que estão sendo utilizadas na transmissão.

Para desempenhar esse papel são definidas algumas portadoras que carregam as configurações do sistema por meio de um sinal conhecido como TMCC (*Transmission and Multiplexing Configuration Control*). Dada a importância do sinal de TMCC para a demodulação do sinal, o padrão define um código cíclico de diferenças para a correção de

erros nas informações do TMCC. Cabe lembrar, que o reenvio dos dados é inviável em sistemas de *broadcasting*, como é o caso da transmissão de TV.

Além disso, para uma melhor proteção dos dados, ainda são definidas regras de embaralhamento com o intuito de espalhar as possíveis falhas ao longo do tempo e do espectro. Uma vez que a natureza das principais fontes de ruído é em rajada, o uso da técnica de embaralhamento aumenta a eficiência dos processos de correção.

Também existem os sinais de estimação de canal, conhecidas como pilotos, que são informações previamente conhecidas pelo receptor. Essas pilotos são espalhadas pelo espectro e permitem que o sistema de recepção possa tomar ações objetivando reduzir possíveis distorções existentes.

2.3 Codificação de Canal

Conforme descrito anteriormente, em um sistema de comunicação, pretende-se enviar dados de um transmissor para um receptor através de um meio de transmissão, também chamado de canal de transmissão. No entanto, seja qual for o tipo de canal, com ou sem fio, sempre haverá degradação de informação. Tal fenômeno ocorre devido ao ruído e às interferências presentes em qualquer tipo de canal prático. Portanto, como não se pode evitar a presença de erros na informação transmitida alterando-se as características do canal, passa-se a aplicar maneiras de diminuir esses efeitos indesejáveis inserindo redundâncias de informação nos dados enviados.

Mesmo assim, os erros não são totalmente evitados através de tais técnicas, ou seja, em um projeto prático, determina-se qual será a taxa máxima de erro aceitável para que o sistema consiga trabalhar sem perder suas funcionalidades. Se for definido que um sistema consegue tolerar até um erro a cada um milhão de *bits* enviados, diz-se que a taxa máxima de erro do mesmo será $1 \cdot 10^{-6}$.

Cabe salientar que o tipo de serviço provido pelo sistema é um fator determinante para o cálculo dessa taxa máxima de erro. Transmissões de dados de voz permitem uma presença maior de erros em relação à transmissão de dados entre computadores, devido à maior capacidade do ouvido humano em recuperar informação a partir do contexto da conversação (DECASTRO,).

Dessa forma, denomina-se codificador de canal o elemento responsável por manter determinado sistema digital em funcionamento garantindo que a taxa máxima de erros

definida não seja ultrapassada.

Sempre se pode inserir algum tipo de codificação de canal quando se pretende enviar dados a uma taxa menor ou igual à capacidade máxima do canal em questão, sendo essa capacidade máxima do canal calculada pela expressão $C = B * log(1 + P/N)$, extraída do Teorema Fundamental de *Shannon* - onde C é a capacidade, B é a largura de banda, N é a potência do ruído no canal e P é a potência do sinal. Caso a taxa a ser utilizada for maior que a capacidade do canal, não haverá codificação que impeça a presença de erros na informação por melhor que esta seja (DECASTRO,).

2.4 TMCC

O TMCC, *Transmission and Multiplexing Configuration Control*, é o sinal utilizado para informar ao receptor quais são as configurações do sinal que está sendo transmitido. Será através desse sinal que o demodulador saberá qual a configuração hierárquica da transmissão, além de informações sobre os parâmetros dos segmentos OFDM. Para que o receptor saiba como obter essas informações corretamente, a localização das portadoras de TMCC e a modulação utilizada, DBPSK, são previamente definidas pela norma japonesa (ARIB, 2005).

O sinal de TMCC possui tamanho igual a 204 *bits*, visto que cada quadro de transmissão OFDM compõe-se de 204 símbolos (RESENDE, 2004). A tabela a seguir ilustra o conteúdo desses 204 *bits*.

bit 0	Referência para modulação diferencial
bits 1 até 16	Sinal de Sincronização
bits 17 até 19	Identificação do Tipo de Segmento
bits 20 até 121	Informação de TMCC
bits 122 até 203	*Bits* de Paridade

Tabela 1: Tabela com os campos de informação do TMCC

Cabe salientar que a porção do TMCC que é protegida pelo código corretor de erros desenvolvido nesse trabalho é a denominada TMCC *Information*, de tamanho igual a 102 *bits*. Portanto, será essa a porção de informação do TMCC na qual nos ateremos a detalhar a sua composição, visto que uma das propostas de otimização do algoritmo de correção envolve a interpretação do comportamento dos valores desse campo.

2.4.1 TMCC *Information*

O TMCC *Information* contém dados que auxiliam o receptor na demodulação e decodificação das informações por ele recebidas. Essas informações são mostradas na tabela a seguir e descritas nos próximos itens.

Bits	Descrição	
bits 20 até 21	Identificação do Sistema	
bits 22 até 25	Indicador de Troca de Parâmetros de Transmissão	
bit 26	Flag para Broadcast de Alarme de Emergência	
bit 27	Informação atual	Indicador de recepção parcial
bits 28 até 40		Informação dos parâmetros de transmissão da camada hierárquica A
bits 41 até 53		Informação dos parâmetros de transmissão da camada hierárquica B
bits 54 até 66		Informação dos parâmetros de transmissão da camada hierárquica C
bit 67	Próxima Informação	Indicador de recepção parcial
bits 68 até 80		Informação dos parâmetros de transmissão da camada hierárquica A
bits 81 até 93		Informação dos parâmetros de transmissão da camada hierárquica B
bits 94 até 106		Informação dos parâmetros de transmissão da camada hierárquica C
bits 107 até 109	'1' para todos os bits	
bits 110 até 121	'1' para todos os bits	

Tabela 2: Tabela com a Informação do TMCC

2.4.2 Identificação do Sistema

Composto por dois *bits*, têm a finalidade de identificar o sistema em utilização, de acordo com a descrição da tabela a seguir.

bits 20 até 21	Significado
00	Sistema baseado na especificação ISDB-T
01	Sistema para ISDB-T$_{SB}$
10, 11	Reservados

Tabela 3: Tabela da Identificação do Sistema

2.4.3 Indicador de Troca de Parâmetros de Transmissão

Esse campo, composto por quatro *bits*, faz uma contagem regressiva de '1111' a '0000', para indicar quando são alterados os parâmetros de transmissão e para ajustes de tempo. A contagem regressiva inicia sempre que há a necessidade de troca de parâmetros e, quando a contagem atinge '0000', o contador retorna para seu valor normal ('1111') no quadro seguinte. É no momento dessa transição que ocorre a troca de parâmetros de transmissão. A tabela 4 representa essa contagem regressiva.

bits 22 até 25	Significado
1111	Valor Normal
1110	15 frames para a troca
1101	14 frames para a troca
1100	13 frames para a troca
⋮	⋮
0010	3 frames para a troca
0001	2 frames para a troca
0000	1 frame para a troca

Tabela 4: Tabela mostrando o processo de contagem regressiva

2.4.4 *Flag* de Recepção Parcial

O conteúdo dessa *flag* indica se o segmento central da banda de transmissão é utilizado para recepção parcial. *Flag* em '0' significa que não é utilizada a recepção parcial, quando em '1', a recepção parcial é disponível. A recepção parcial é utilizada principalmente por dispositivos móveis e essa *flag* indica se o conteúdo para tais dispositivos está acessível ou não. Caso essa *flag* esteja habilitada, a camada hierárquica A deve possuir apenas 1 segmento.

2.4.5 Parâmetros de Transmissão

Esses são os valores que informam os parâmetros de transmissão referentes a cada um dos níveis hierárquicos A, B e C, tanto nos campos de informação atual, como nos campos de próxima informação, se houver. Existem quatro campos de conteúdo a serem descritos a seguir:

Descrição	Número de Bits
Esquema de Modulação de Portadora	3
Taxa de Código-Convolucional	3
Comprimento de *Interleaving* de tempo	3
Número de Segmentos	4

Tabela 5: Parâmetros de Transmissão

2.4.6 Modulação de Portadora

O conteúdo desse campo de configuração indica o esquema de modulação que é utilizado em cada nível hierárquico, conforme mostrado na tabela a seguir.

bits 28-30, 41-43, 54-56, 68-70, 81-83, 94-96	Significado
000	DQPSK
001	QPSK
010	16QAM
011	64QAM
100-110	Reservados
111	Camada hierárquica não usada

Tabela 6: Esquemas de Modulação

2.4.7 Taxa de Codificação Convolucional

Indica os valores de taxa de codificação convolucional conforme mostrado na tabela a seguir.

bits 31-33, 44-46, 57-59, 71-73, 84-86, 97-99	Significado
000	1/2
001	2/3
010	3/4
011	5/6
100	7/8
101-110	Reservados
111	Camada hierárquica não usada

Tabela 7: Taxa de Codificação Convolucional

2.4.8 Comprimento do *Interleaving* de Tempo

Os valores são de acordo com a tabela a seguir.

bits 34-36, 47-49, 60-62, 74-76, 87-89, 100-102	Significado
000	0 (Modo 1), 0 (Modo 2), 0 (Mode 3)
001	4 (Modo 1), 2 (Modo 2), 1 (Mode 3)
010	8 (Modo 1), 4 (Modo 2), 2 (Mode 3)
011	16 (Modo 1), 8 (Modo 2), 4 (Mode 3)
100-110	Reservados
111	Camada hierárquica não usada

Tabela 8: *Interleaving Length*

2.4.9 Número de Segmentos

Esse campo indica o número de segmentos de cada camada hierárquica, conforme a tabela a seguir.

bits 37-40, 50-53, 63-66, 77-80, 90-93, 103-106	Significado
0000	Reservado
0001	1 segmento
0010	2 segmentos
0011	3 segmentos
0100	4 segmentos
0101	5 segmentos
0110	6 segmentos
0111	7 segmentos
1000	8 segmentos
1001	9 segmentos
1010	10 segmentos
1011	11 segmentos
1100	12 segmentos
1101	13 segmentos
1110	Reservado
1111	Camada hierárquica não utilizada

Tabela 9: Número de Segmentos

2.5 Teoria de Erros

Conforme citado na seção referente à codificação de canal, em um sistema de comunicação prático, sempre ocorre a degradação da informação. Essa ocorrência de erros pode ser medida através da frequência com que esses erros ocorrem. Essa taxa de *bits* com erro, chamada de BER, representa a razão entre a quantidade de *bits* com erro sobre o total de *bits* da informação transmitida. A BER pode ser considerada como uma das medidas de desempenho mais importantes, do ponto de vista do usuário, ao lado do *throughput*

(vazão dos dados).

Quando se pretende implementar um sistema de comunicação nos moldes desse trabalho, um dos requisitos básicos a ser definido é justamente o valor da BER. A exigência desse valor varia conforme o tipo de dado que será transmitido, audio, vídeo, dados, etc. Além disso, a característica do sistema ser ou não de tempo real, a capacidade do canal, e outros diversos fatores determinam que, para garantir o funcionamento adequado do sistema, se utilize uma codificação de canal adequada.

Para fazer essa codificação de canal, pode-se utilizar tanto códigos detectores como corretores de erro, conforme a exigência de desempenho do sistema. Como o nome já enuncia, códigos detectores realizam apenas a verificação da ocorrência dos erros, o que pode ser útil nos casos em que a retransmissão de informação não interfira consideravelmente no desempenho da comunicação. Já no caso dos códigos corretores de erro, a retransmissão dos dados pode ser dispensada visto que os erros serão corrigidos no receptor, exceto nos casos em que o algoritmo não tiver o poder de corrigir dada a elevada degradação dos dados imposta pelo canal.

O caso desse trabalho, refere-se aos códigos corretores de erro. Tais códigos podem ser subdivididos em diversos tipos como, por exemplo, os de bloco, os cíclicos, os convolucionais, entre outros. O código corretor desenvolvido nesse trabalho classifica-se como cíclico com a utilização de lógica majoritária.

Os códigos cíclicos podem ser entendidos como um sub-conjunto dos códigos de bloco e tornaram-se bastante usados pela facilidade de implementação em *hardware*, baseada em operações de deslocamento cíclico através de registradores de deslocamento. Esses registradores estão entre os circuitos digitais mais simples, visto que consistem de uma coleção de *flip-flops* conectados em série, o que possibilita uma capacidade de trabalho em velocidades bastante elevadas, como as encontradas em diversos tipos de dispositivos atuais. (SMITH, 2005)

Além da presença dos registradores de deslocamento, o *hardware* também é composto de outros elementos também considerados simples, como portas ou-exclusivo, *switches*, ou ainda nos casos do uso de palavras não binárias, percebe-se a utilização de circuitos somadores e multiplicadores finitos (WICKER, 1994). No entanto, em contra-partida, a teoria que determina essa estrutura de *hardware* está relacionada com o campo da álgebra abstrata, que requer um estudo mais sofisticado em relação a outras áreas da matemática com as quais costumeiramente trabalhamos.

Para um melhor entendimento do funcionamento matemático dos códigos corretores cíclicos, segue uma definição destes:

Teorema 1. *Um código de bloco linear $C(n,k)$ é cíclico se para toda palavra do código $c = (c_0, c_1, ..., c_{n-2}, c_{n-1}) \in C$, então também é uma palavra do código $c' = (c_{n-1}, c_0, c_1,$ $..., c_{n-2}) \in C$.*

Em computação, diz-se que a palavra c' é um deslocamento cíclico para a direita da palavra c. Como c foi escolhido arbitrariamente dentre as palavras contidas em C, tem-se que todos os n deslocamentos cíclicos distintos de c devem também ser palavras de código em C.

Resumindo, um código C é dito cíclico quando o deslocamento de cada palavra de código também for uma palavra desse mesmo código. O funcionamento desses códigos está diretamente associado a um polinômio código $c(x) = c_0 + c_1x + c_2x^2 + ... + c_{n-1}x^{n-1}$ com toda palavra de código $c = (c_0, c_1, ..., c_{n-2}, c_{n-1})$ pertencendo a C.

Portanto, um código C de tamanho n pode ser representado como um conjunto de polinômios em K de grau, no máximo, $n - 1$. Os polinômios em K são adicionados e multiplicados normalmente, exceto no caso como $1 \oplus 1 = 0$, quando daí teremos $x^k \oplus x^k = 0$. Este conjunto de polinômios $K[x]$, assim como a coleção de palavras pertencentes a C, formam subespaços vetoriais dentro do espaço de todas as n combinações sobre o corpo de *Galois* $GF(2)$ (HSU, 2006).

Além disso, sejam $f(x)$ e $h(x)$ em $K[x]$, com $h(x)$ diferente de 0, pode-se então afirmar que existem polinômios únicos $q(x)$ e $r(x)$ em $K[x]$ tais que $f(x) = q(x)h(x) + r(x)$ com $r(x) = 0$ e o grau de $r(x)$ sendo menor que o grau de $h(x)$. O polinômio $q(x)$ é chamado de quociente e $r(x)$ de resto. O cálculo utilizado para se chegar à $q(x)$ e à $r(x)$ baseia-se na divisão de módulo 2 envolvendo os coeficientes.

O funcionamento dos códigos cíclicos exige a definição de um polinômio gerador de estrutura fixa e conhecida tanto pelo transmissor como pelo receptor. O seguinte teorema mostra que:

Teorema 2. *Se $g(x)$ é um polinômio de grau$(n - k)$ que seja um fator de $1 + x^n$, então $g(x)$ gera um código cíclico $C(n,k)$ cujo código polinomial $c(x)$ para uma dada palavra $d = (d_0, d_1, ..., d_{k-1})$ é gerado por $c(x) = d(x)g(x)$, onde $d(x) = d_0 + d_1x + ... + d_{k-1}x^{k-1}$ é o dado polinomial correspondente à palavra de dado.*

Outro importante fator a ser discutido é a utilização de um polinômio de verificação de paridade. Seja $h(x)$ um polinômio de grau k e também um fator de $1+x^n$. Então, $h(x)$

é o polinômio de verificação de paridade de um código cíclico $C(n,k)$. O polinômio de verificação de erro $h(x)$ e o gerador polinomial $g(x)$ de C são relacionados por $g(x)h(x) = 0mod(1+x^n)$ ou ainda $g(x)h(x) = 1+x^n$

Existe ainda o polinômio síndrome. Seja $r(x)$ a palavra polinomial recebida quando o polinômio de código $c(x)$ foi enviado. Então, $r(x) = c(x) + e(x)$, onde $e(x)$ é o polinômio de erro mais parecido. O polinômio de síndrome $s(x)$ é definido por $s(x) = r(x)mod\ g(x)$.

Assumindo que $g(x)$ possui grau $n-k$, então $s(x)$ terá grau menor do que $n-k$ e irá corresponder a uma palavra binária s de tamanho $n-k$. Como $c(x) = d(x)g(x)$, temos $s(x) = e(x)mod\ g(x)$.

Portanto, o polinômio de síndrome é dependente apenas do erro. O teorema a seguir especifica a condição para a existência de um único polinômio de síndrome.

Teorema 3. *Seja C um código cíclico com distância mínima d_{min}. Cada polinômio de erro de peso menor do que $\frac{1}{2}d_{min}$ possui um único polinômio síndrome.*

A partir disso, afirma-se que o problema de correção de erro é, então, a determinação do único $e(x)$ com o menor peso satisfazendo a equação $s(x) = e(x)modg(x)$.

Por fim, existem diversas matrizes geradoras G para esses códigos cíclicos. Um esquema simples é a matriz na qual as linhas são as palavras de código correspondentes ao polinômio gerador $g(x)$ e seus primeiros $k-1$ deslocamentos cíclicos: (HSU, 2006)

$$G = \begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

3 Implementação

3.1 Codificador

O código usado para a geração da paridade e posteriormente correção dos dados enviados está de acordo com as especificações estabelecidas em (ARIB, 2005). O codificador do sistema ISBD-T de transmissão de TV digital pertence à classe dos códigos cíclicos, possuindo assim, baixa complexidade de *hardware* e de operação, apesar de efetuar operações lógicas sobre um polinômio de grau 82.

O codificador desse sistema foi implementado em linguagem de descrição de *hardware* (VHDL), além disso foi escrito um *script* para Matlab®, a fim de agilizar o processo de testes e validação tanto do código como do sistema como um todo.

O princípio básico de funcionamento do codificador pode ser visualizado na Figura 2. A seguir será dada uma explicação geral do seu funcionamento.

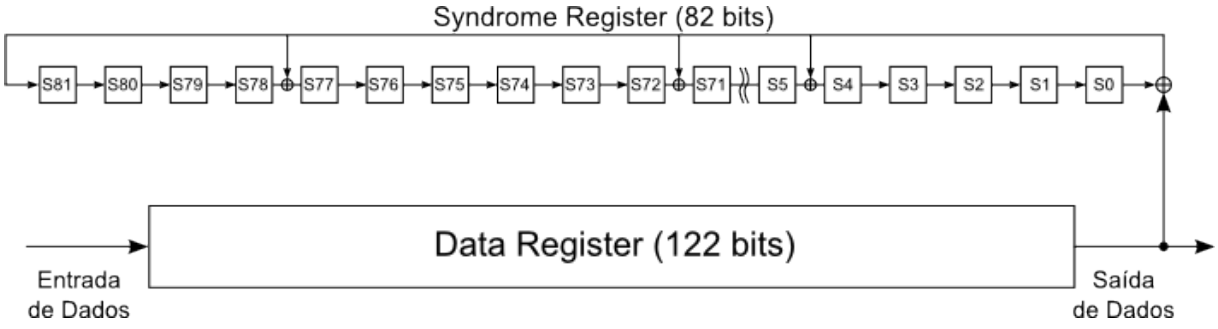

Figura 2: Representação gráfica do codificador.

Os dados provenientes do bloco que monta o pacote de TMCC chegam ao codificador de maneira paralela, ou seja, em um ciclo de *clock*, 122 *bits* são carregados para dentro do código codificador em um vetor chamado *data_register*. Assim que os dados forem carregados no vetor, o processo de geração de paridade tem início. O envio de dados por parte do bloco TMCC ao codificador é feito através de um protocolo que utiliza 4 sinais. A Figura 3 exemplifica o uso do protocolo.

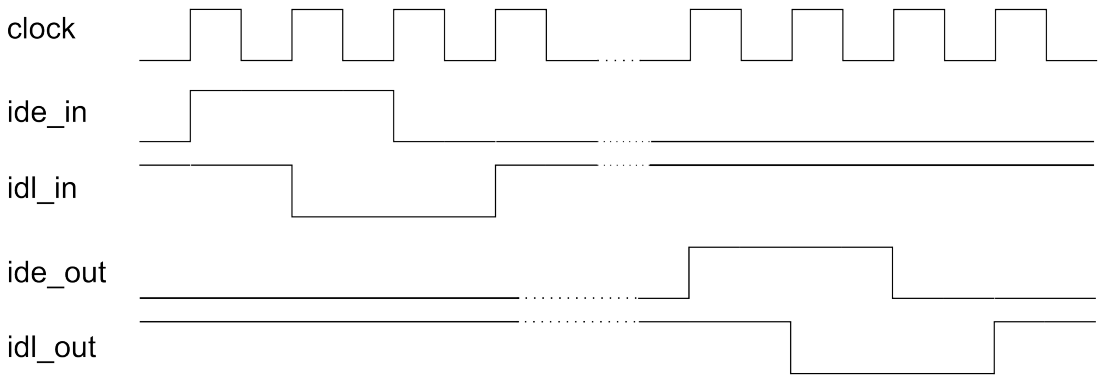

Figura 3: Protocolo de comunicação entre blocos.

O protocolo funciona com dois sinais na parte da recepção dos dados e dois sinais no envio dos dados ao próximo bloco. O fluxo de operação do protoloco é o seguinte:

- **ide_in (informa dado escrito - interface de entrada):** O sinal *ide_in* passa do nível lógico '0' para o nível lógico '1' quando o bloco que monta o pacote de TMCC escreve dados no barramento de 122 *bits*, e é mantido em '1' até que o codificador leia o dado do barramento e altere o sinal *idl_in* a fim de sinalizar que o dado foi lido. Uma vez identificada a leitura dos dados, o sinal volta ao nível lógico '0'.

- **idl_in (informa dado lido - interface de entrada):** Esse sinal serve para informar ao bloco anterior que o dado foi lido corretamente e o mesmo pode voltar a sua execução normal. O sinal *idl_in* está sempre em nível lógico '1' e assim que identifica que há dados disponíveis no barramento muda para o nível lógico '0', o codificador lê o dado do barramento e este sinal volta ao nível lógico '1'.

- **ide_out (informa dado escrito - interface de saída):** Similar ao *ide_in*, porém agora na interface de saída. Enquanto a paridade está sendo calculada esse sinal fica em nível lógico '0', e assim que ocorre o término da execução, o sinal é alterado para nível lógico '1' e os 204 *bits* (informação e paridade) são postos no barramento de saída de forma paralela. O sinal permanece em nível lógico '1' até que haja confirmação da leitura por parte do bloco seguinte através do sinal *idl_out*.

- **idl_out (informa dado lido - interface de saída):** Também possui funcionamento similar ao sinal *idl_in*, entretanto localiza-se na interface de saída. Uma vez que os 204 *bits* de informação e paridade são colocados no barramento e retirados pelo bloco posterior, o mesmo utiliza esse sinal para informar que a leitura foi feita corretamente e o codificador pode voltar a sua execução normal. O sinal permanece sempre em nível lógico '1', quando a leitura é feita pelo bloco posterior o sinal

é alterado para nível lógico '0' e logo em seguida para o nível lógico '1' caso não ocorram problemas durante a comunicação.

Esse tipo de comunicação, apesar de apresentar certa complexidade, garante que os dados foram entregues e lidos pelos blocos anteriores e posteriores, tornando o sistema mais robusto. Além disso, o custo de tempo e *hardware* para a implementação desse protocolo não será relevante quando comparado com o resto do sistema.

Uma vez estabelecida a comunicação na interface de entrada e feita a leitura dos dados, tem início então o processo de geração da paridade para a mensagem de 122 *bits*. Nesse ponto, os 20 primeiros *bits* da mensagem de TMCC são descartados, seguindo a norma ARIB-B31. Esses *bits* carregam informação de sincronia e sobre o tipo de segmentos, e não fazem parte do TMCC *Information*, logo não serão protegidos pela paridade.

Já no processo de geração da paridade, existe um vetor chamado *syndrome_register*, que é onde, após algumas operações, se encontrarão os 82 *bits* da paridade. A Figura 4 mostra como os dados são computados para gerar a paridade.

Figura 4: Operação de OR exclusivo para cada bit do TMCC.

Para cada *bit* do TMCC é feita uma operação de OU exclusivo (XOR) e o resultado dessa operação serve como realimentação para o vetor *syndrome_register* como mostrado na Figura 2. Cada realimentação no vetor *syndrome_register* corresponde a um termo do polinômio gerador.

Como havia sido comentado anteriormente, o vetor de *syndrome_register* possui 82 posições porque ele representa o resto da divisão entre a mensagem que se encontra no *data_register* e o polinômio gerador. O polinômio pode ser observado abaixo. Ele será utilizado tanto para o codificador quanto para o decodificador.

$$g(x) = x^{82} + x^{77} + x^{76} + x^{71} + x^{67} + x^{66} + x^{56} + x^{52} + x^{48} + x^{40} + x^{36} + x^{34} + x^{24} + x^{22} + x^{18} + x^{10} + x^4 + 1$$

Durante todo o processo de codificação, todos os *bits* da mensagem de TMCC armazenados no vetor *data_register* sofrem uma operação de *rotate logical right*, ou seja, a cada ciclo de *clock* o *bit* menos significativo é lido, usado nas operações de XOR necessárias e passa a se tornar o *bit* mais significativo da mensagem. A Figura 5 mostra como funciona essa operação.

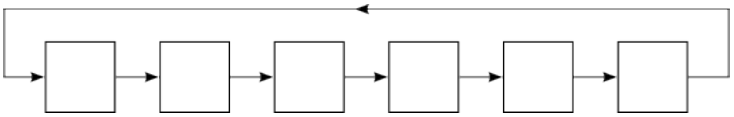

Figura 5: Operação de *rotate logical right*.

Após 102 ciclos de *clock*, todos os *bits* da mensagem de TMCC foram rotacionados e a mensagem encontra-se no vetor *data_register* em sua forma original, exatamente como havia sido lida na entrada do bloco. No vetor *syndrome_register* encontra-se o resto da divisão da mensagem de TMCC pelo polinômio, ou seja, nesse ponto da execução, o vetor *syndrome_register* contém os 82 *bits* de paridade.

Concluído o rotacionamento e consequentemente o processo de codificação, o sistema disponibiliza os dados no barramento e inicia a comunicação através do protocolo com o bloco seguinte. Os 204 *bits* disponibilizados no barramento são formados pela concatenação da mensagem de TMCC e paridade. A Figura 6 mostra como é feita a concatenação.

Figura 6: Concatenação dos 204 *bits* gerados.

3.2 Decodificador

Assim como o codificador, o decodificador do sistema ISDB-T de TV digital foi implementado seguindo as normas disponibilizadas a fim de manter compatibilidade com o sistema como um todo. O esquema de funcionamento do decodificador é bastante similar ao codificador, sendo que a principal diferença se encontra no uso de uma unidade de processamento de lógica majoritária, que pode ser vista como um circuito votador.

O decodificador aqui descrito foi implementado em linguagem de descrição de *hardware* (VHDL) para manter compatibilidade entre os demais blocos da cadeia de demodulação. Além disso, a linguagem VHDL possui elevado nível de abstração quando se

deseja trabalhar ao nível de *bits*, *bytes* e plataformas reprogramáveis (FPGAs).

Tal qual o codificador, a leitura de dados do barramento no decodificador é feita de forma paralela, porém agora são 273 *bits* que serão carregados para os registradores internos do bloco simultaneamente. Dentro desse pacote de 273 *bits* encontra-se a mensagem de TMCC transmitida, a paridade transmitida e 69 zeros a fim de ajustar o processo de decodificação. Além disso, os dados são lidos através do uso do mesmo protocolo utilizado no codificador exemplificado na Figura 3. A Figura 7 mostra como o pacote recebido é montado.

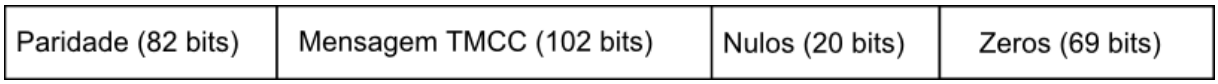

Figura 7: 273 *bits* de dados concatenados na decodificação.

A partir do momento que foi estabelecida a comunicação e os dados foram lidos do barramento, tem início o processo de decodificação. O decodificador possui 3 vetores principais e um sinal bastante importante. Os vetores e o sinal são:

- ***data_register*(273 posições)**: na leitura, é responsável por armazenar os dados a serem decodificados e na execução é responsável por conter os dados rotacionados.
- ***syndrome_register*(82 posições)**: assim como no codificador, esse vetor recebe realimentações e ao término da execução contém o resultado do resto da divisão da mensagem recebida pelo polinômio gerador.
- ***syndrome_sums*(17 posições)**: é através desse vetor que a lógica majoritária (votador) irá decidir se determinado *bit* está certo ou não.
- ***bit_correction***: aliado ao vetor *syndrome_sums* esse sinal faz a correção do *bit* caso o mesmo esteja errado.

Na primeira etapa da correção, os dados contidos no vetor *data_register* são rotacionados 273 vezes para calcular o conteúdo do vetor *syndrome_register*. Durante esses 273 ciclos de *clock* nenhuma outra operação é feita. Assim que é concluída a operação de cálculo do vetor de síndrome, a mensagem de TMCC se encontra na sua forma original. A partir disso tem início a operação de correção propriamente dita.

Nesse ponto é interessante notar que o decodificador possui algumas diferenças quanto a operação de XOR entre o *bit* menos significativo, que varia a cada ciclo de *clock* devido

ao rotacionamento, em relação a mesma operação no codificador. As realimentações do vetor *syndrome_register* são iguais tanto no codificador quanto no decodificador uma vez que ambos utilizam o mesmo polinômio, porém, como comentado anteriormente. Se comparadas as Figuras 4 e 8 é possível observar que a operação de XOR acontece em locais diferentes no codificador e decodificador.

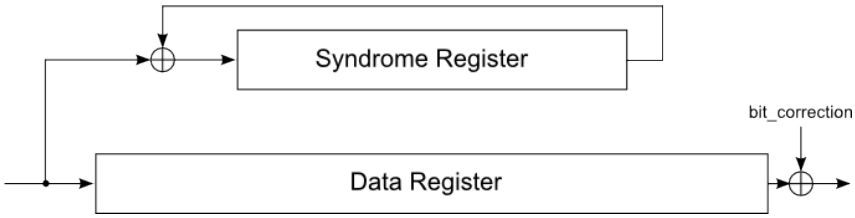

Figura 8: Operação de OU exclusivo no decodificador.

Na segunda etapa de correção, a mensagem de TMCC recebida será mais uma vez rotacionada 273 vezes, porém agora serão feitas correções em até 8 *bits* errados caso seja necessário. Durante a execução do segundo rotacionamento completo da mensagem de TMCC, o circuito de detecção e correção de erros permanece ativo, o que não acontece durando o primeiro rotacionamento.

Para cada *bit* da segunda etapa de correção são feitas operações de OU exclusivo (XOR) sobre o vetor *syndrome_sums* a fim de verificar a existência de erros. As operações consistem em: para cada posição do vetor *syndrome_sums*, operações de XOR entre determinadas posições do vetor *syndrome_register* são executadas. Supondo que A(x) sejam posições do vetor *syndrome_sums* e S(x) sejam posições do vetor *syndrome_register* as operações executadas são as seguintes:

A(1) = S(10)+S(5)
A(2) = S(64)
A(3) = S(76)+S(58)
A(4) = S(60)+S(54)+S(36)
A(5) = S(78)+S(56)+S(50)+S(32)
A(6) = S(65)+S(61)+S(39)+S(33)+S(15)
A(7) = S(46)+S(29)+S(25)+S(3)
A(8) = S(73)+S(37)+S(20)+S(16)
A(9) = S(79)+S(70)+S(34)+S(17)+S(13)
A(10) = S(71)+S(68)+S(59)+S(23)+S(6)+S(2)
A(11) = S(80)+S(69)+S(66)+S(57)+S(21)+S(4)+S(0)
A(12) = S(51)+S(49)+S(38)+S(35)+S(26)
A(13) = S(75)+S(44)+S(42)+S(31)+S(28)+S(19)
A(14) = S(81)+S(74)+S(43)+S(41)+S(30)+S(27)+S(18)

A(15) = S(63)+S(62)+S(55)+S(24)+S(22)+S(11)+S(8)
A(16) = S(72)+S(53)+S(52)+S(45)+S(14)+S(12)+S(1)
A(17) = S(77)+S(67)+S(48)+S(47)+S(40)+S(9)+S(7)

A partir dessas operações que acontecem a cada ciclo de *clock*, é possível saber se o *bit* na posição menos significativa do vetor *data_register* está certo ou errado. Para fazer essa verificação, conta-se o número de posições do vetor *syndrome_sums* em '1' e, se existirem mais de 8 posições em '1', significa que o *bit* que está sendo processado naquele momento (bit menos significativo do vetor *data_register*) está errado. Caso a soma de '1's seja menor que 8, nada é feito, e o circuito de correção não é ativado.

O circuito de correção implementado consiste de um sinal (*bit_correction*) que, dependendo da soma de '1's no vetor *syndrome_sums*, é ativado ou não. O sinal *bit_correction* altera somente o valor presente na posição menos significativa do vetor *data_register* no mesmo ciclo de *clock* em que o erro foi identificado. A Figura 9 mostra como é feita a correção a partir da lógica majoritária.

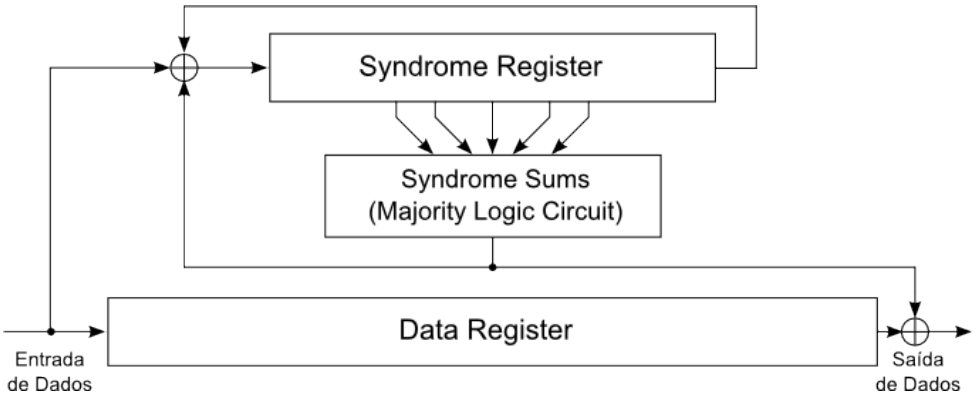

Figura 9: Circuito de correção.

Tendo sido feito o completo rotacionamento da mensagem de TMCC pela segunda vez, com os circuitos de detecção e correção ativos, os dados contidos no vetor *data_register* são considerados corrigidos e podem ser passados ao bloco seguinte. Apesar de o vetor *data_register* conter 273 *bits*, apenas 122 são disponibilizados no barramento. Esses 122 *bits* são a mensagem de TMCC contendo informações sobre as configurações do sistema.

A comunicação com o bloco seguinte também é feita através do protocolo descrito anteriormente e, uma vez terminada a execução e os dados garantidamente lidos pelo bloco seguinte, o decodificador fica à espera de uma nova mensagem para decodificar. Tendo em vista a diferença entre as velocidades do *clock* do decodificador e do bloco anterior e posterior ao decodificador, esse após processar e corrigir os dados, permanece um considerável tempo sem processamento até que chegue uma nova palavra para ser decodificada.

Aproveitando-se disso, foram feitas algumas otimizações na etapa de decodificação que podem ser observadas no próximo capítulo.

4 Otimização

4.1 Algoritmo Segundo a Norma ARIB

Segundo a norma (ARIB, 2005), o código corretor de erros é implementado usando uma variante do código cíclico diferencial (273,191). Uma vez que o pacote contém 122 *bits*, e somente 102 *bits* são efetivamente informações válidas, os 20 primeiros *bits* são descartados, gerando a variante (184,102) no modulador.

Devido ao fato das informações contidas no pacote de TMCC serem usadas para especificar parâmetros de transmissão e controle na operação do receptor, elas devem ser transmitidas com maior confiabilidade em relação a outros tipos de dados. Adicionalmente, as dificuldades envolvidas na recepção dos dados, o curto tempo de processamento disponível e a garantia de dados livres de erros, tornam o código (273,191) como sendo o código corretor de erro para as informações contidas no TMCC (ARIB, 2005).

4.2 Algoritmo Otimizado Através de Pré e Pós-Processamento

Nesta seção serão descritas as implementações dos blocos de pré e pós-processamento dos dados em relação ao decodificador desenvolvido. A proposta de otimização baseia-se na possibilidade de previsão dos valores a serem recebidos no TMCC pelo receptor. Essa previsibilidade é possível através do entendimento do significado e função dos campos de dados que são transmitidos pelas portadoras de TMCC. Portanto, essa otimização apenas é válida nos casos em que o corretor de erros abordado neste trabalho é utilizado no contexto do sistema de transmissão ISDB-T de TV digital seguindo as especificações da norma ARIB. Caso contrário, não há como haver uma previsibilidade dos dados a serem recebidos, a menos que ocorram alterações nos blocos de processamento existentes para que os mesmos atendam às especificações de outro sistema de comunicação.

Os blocos de pré e pós-processamento têm por objetivo corrigir *bits* que possam ter sofrido inversão antes que a palavra passe para o decodificador, ou seja, esses blocos visam diminuir o número de erros da mensagem, o que pode fazer a diferença entre tornar a mensagem recebida capaz de ser corrigida ou não pelo decodificador de erros. A partir da análise do conteúdo do TMCC, é possível identificar se há coerência entre a informação atual e a informação anterior que já foi decodificada.

4.2.1 Pré-Processamento de Pacote

O bloco de pré-processamento fica colocado à frente do decodificador de erros. Esse bloco é responsável por, através da análise dos dados recebidos da portadora de TMCC, fazer uma pré-correção. Essa pré-correção se baseia no fato de que determinados campos de informação do TMCC, do *bit* 20 até o *bit* 121, podem ter seus valores pré-determinados a partir do último pacote com decodificação bem sucedida, segundo regras de funcionamento do sistema ISDB-T.

Partindo do uso da implementação do decodificador não otimizado, que é capaz de corrigir até 8 *bits* com erro, conclui-se que se uma mensagem com 9 erros ou mais for recebida, o decodificador não será capaz de corrigir essa informação. No entanto, se o bloco de pré-processamento conseguir corrigir alguns *bits* que estão errados simplesmente analisando o contexto da mensagem recebida, a palavra pode vir a apresentar 8 ou menos erros e, portanto, passa a ser corrigível pelo decodificador. Ou seja, com a adição do bloco de pré-processamento torna-se possível ter um bloco de correção do TMCC com maior confiabilidade. A Figura 10 mostra o esquema do bloco de pré-processamento junto com o código corretor.

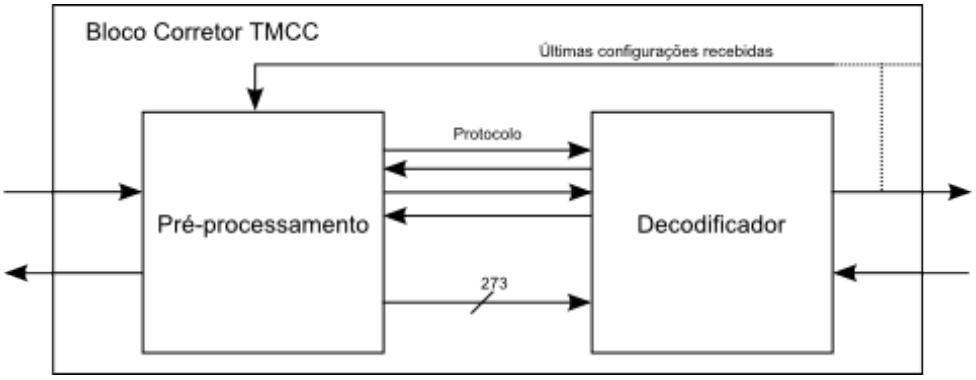

Figura 10: Bloco de pré-processamento.

Para manter a compatibilidade com o sistema já implementado, foi adotado o mesmo protocolo de comunicação descrito no capítulo anterior, sendo que a utilização do bloco

de pré-processamento é transparente para os componentes do sistema no que diz respeito a comunicação.

A entidade do código de pré-processamento implementado em VHDL é a seguinte:

```
entity TMCC_decoder_pre_proc is
   port(
      clock    : in  std_logic;
      sreset   : in  std_logic;

      current_information : in std_logic_vector(39 downto 0);
      next_information    : in std_logic_vector(39 downto 0);
      ind_trans_par_sw    : in std_logic_vector( 3 downto 0);

      data_in  : in  std_logic_vector(272 downto 0);
      data_out : out std_logic_vector(272 downto 0);

      ide_in   : in  std_logic;
      idl_in   : out std_logic;
      ide_out  : out std_logic;
      idl_out  : in  std_logic
    );
end TMCC_decoder_pre_proc;
```

Conforme explicado no capítulo referente ao TMCC, esse se divide em alguns tipos de informações de configuração para uso do receptor. Um dos campos, por exemplo, que pode ser analisado pelo pré-processamento é o sinal de troca de parâmetros de configuração, ou simplesmente *countdown*, visto que a partir desse é possível saber se está para acontecer mudanças nos parâmetros de transmissão do canal. Isso implicará na ocorrência de alterações ou não de conteúdo nos campos *informação atual* e *próxima informação*. As regras utilizadas pelo pré-processamento são as seguintes:

- O campo de *Identificação do Sistema* deve sempre estar em '00' para o sistema ISDB-T.
- O *countdown* tem seu valor normal em 15.
- O campo de *informação atual* não deve mudar na operação normal do sistema.
- Quando inicia-se o processo de configuração do sistema, o valor do *countdown* é decrementado a cada pacote.

- Durante o processo de configuração, os campos de *informação atual* e *próxima informação* não devem ter seus valores alterados.
- Quando o *countdown* atinge o valor 0, seu próximo valor será 15.
- Durante a transição do *countdown* o valor do campo de *próxima informação* é transferido para o campo de *informação atual*.
- Os últimos 15 *bits* do TMCC devem estar todos em '1'.

Caso a informação recebida não siga essas regras, pode ser um sinal de que erros tenham sido inseridos pelo canal, nesse caso o bloco pré-corretor altera as informações recebidas pelas informações esperadas antes de enviá-la para o decodificador.

4.2.1.1 Análise da capacidade de correção do pré-processamento

A fim de avaliarmos a capacidade de correção adquirida ao utilizarmos o bloco de pré-processamento, devemos definir os 3 estados possíveis de operação do bloco, que são consequência das regras citadas na seção anterior.

- Operação normal - ocorre quando o *countdown* está com o valor 15. Durante esse estado não devem ser alterados os seguintes *bits*: os 2 *bits* de *Identificação do Sistema*, os 3 *bits* mais significativos do *countdown* (uma vez que os únicos valores possíveis para novos pacotes são 15 ('1111') e 14 ('1110')), os 20 *bits* de *informação atual*, e os 15 últimos *bits* de informação. Totalizando 40 *bits* que podem ter seus erros eliminados antes do decodificador.
- Operação antes da mudança de configuração - ocorre quando o *countdown* inicio sua contagem. Durante esse estado não devem ser alterados os seguintes *bits*: os 2 *bits* de *Identificação do Sistema*, os 20 *bits* de *informação atual*, os 20 *bits* de *próxima informação*, e os 15 últimos *bits* de informação. Além dos 4 *bits* do *countdown* poderem ser previstos. Totalizando 61 *bits* que podem ser previamente corrigidos.
- Operação durante a mudança de configuração - ocorre quando o *countdown* passa de 0 para 15. Durante esse estado não devem ser alterados os seguintes *bits*: os 2 *bits* de *Identificação do Sistema*, e os 15 últimos *bits* de informação. Além de o *countdown* e o campo de *Informação atual* poderem ser previstos. Totalizando 41 *bits* que poder ser previamente corrigidos.

Segundo as definições anteriores, o estado de operação no qual o pré-processamento tem a capacidade de identificar menos erros é o de *operação normal*. Por se tratar do pior caso, o utilizaremos na análise da capacidade de correção. Além disso, esse estado em que o sistema opera na maior parte do tempo.

Como o decodificador por si só tem capacidade de corrigir até 8 *bits*, qualquer correção feita pelo pré-processamento, nesse caso, não acarretaria em ganhos para o decodificador. Por esse motivo analisaremos apenas casos em que há a ocorrência de mais de 8 erros.

Considerando a ocorrência de 9 erros, é necessário que pelo menos 1 dos *bits* em erro tenha ocorrido dentre os 40 protegidos pelo pré-processamento para que se respeite a capacidade de correção do decodificador. Para simplificar os cálculos consideraremos a possibilidade de que os 9 erros ocorram na área não protegida (144 *bits*), sendo assim o seu complementar será a probabilidade do pré-processamento tornar a informação decodificável.

O número de palavras de 184 *bits* onde os 9 erros encontram-se dentro dos 144 não protegidos é dado pela equação 4.1:

$$C_9^{144} = 5{,}6849 \cdot 10^{13} \tag{4.1}$$

O número total de palavras com 9 erros no pacote de 184 pode ser obtido pela equação 4.2:

$$C_9^{184} = 5{,}4612 \cdot 10^{14} \tag{4.2}$$

Logo, a probabilidade de uma palavra com 9 erros passar pelo pré-processamento e não poder ser corrigida é de:

$$\frac{C_9^{144}}{C_9^{184}} = 0{,}1041 \tag{4.3}$$

Assim sendo, através do complementar do valor calculado anteriormente, temos a probabilidade do sistema de correção corrigir uma palavra com 9 erros. De onde concluímos que o sistema de correção otimizado é capaz de corrigir 89,59% ($1-0{,}1041$) das mensagens com exatamente 9 erros.

Podemos generalizar o cálculo da probabilidade de correção para mais erros pela equação 4.4:

$$prob(n_{erros}) = 1 - \frac{\sum_{i=9}^{n_{erros}} C_i^{144} * C_{n_{erros}-i}^{40}}{C_{n_{erros}}^{184}} \tag{4.4}$$

A Figura 11 apresenta a probabilidade de correção de informação em função do número de erros ocorridos.

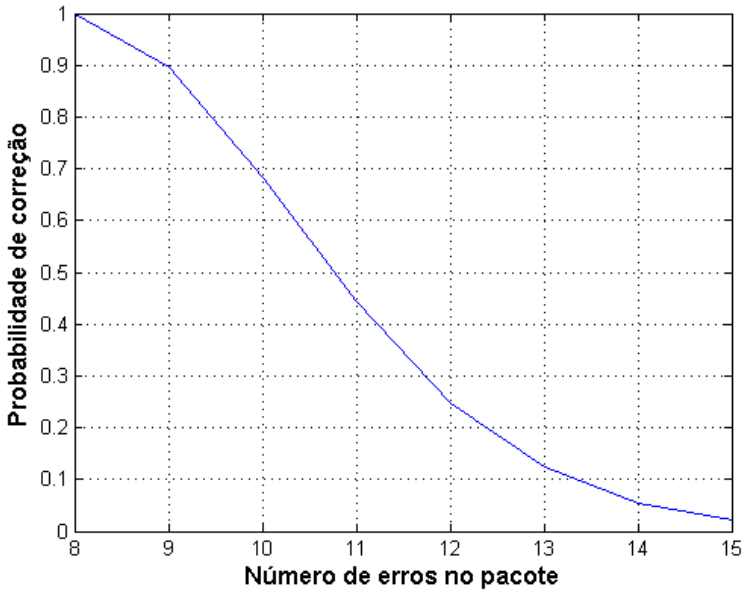

Figura 11: Probabilidade de Correção vs. Número de Erros Ocorridos.

Contudo, essa previsibilidade de recebimento de informação, que é o cerne da lógica do pré-processador, pode vir a não surtir os efeitos desejados caso ocorram mudanças bruscas de conteúdo do TMCC, situação esta que pode ocorrer, por exemplo, quando há mudança de canal. Por isso, quando for percebido pela realimentação do bloco de pós-processamento, que o decodificador não conseguiu corrigir um determinado número mensagens consecutivas, os blocos de pré e pós-processamento são desativados, e retomarão sua atividade assim que o sistema entrar em um estado de estabilidade.

4.2.2 Pós-Processamento de Pacote

O bloco de pós-processamento tem por finalidade mascarar possíveis erros de decodificação, tanto para o resto do sistema quanto para o bloco de pré-processamento, uma vez que esse é realimentado com a saída do pós-processamento. A partir das informações enviadas pelo bloco de pós-processamento ao bloco de pré-processamento a avaliação e posterior correção do TMCC é feita. Para tanto, posiciona-se o bloco de pós-processamento após a saída do decodificador de erros gerando, assim, uma realimentação no sistema. A

Figura 12 mostra o esquema do bloco de pós-processamento juntamente com o bloco de pré-processamento e o código corretor.

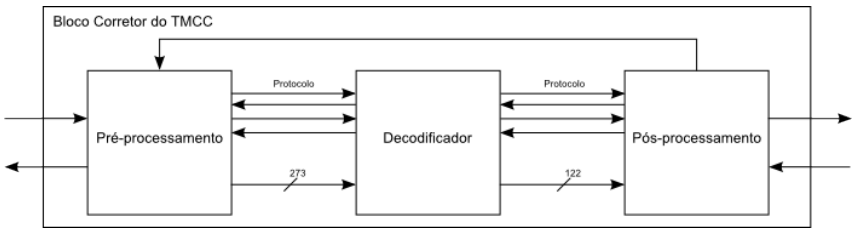

Figura 12: Bloco de pós-processamento.

A entidade do bloco de pós-processamento implementado em VHDL é a seguinte:

```
entity pros_proc is
   port(
      clock     : in  std_logic;
      reset     : in  std_logic;
      data_in   : in  std_logic_vector(121 downto 0);
      data_out  : out std_logic_vector(121 downto 0);

      ide_in    : in  std_logic;
      idl_in    : out std_logic;
      ide_out   : out std_logic;
      idl_out   : in  std_logic;

      info_erro : in  std_logic;

      current_information : out std_logic_vector(39 downto 0);
      next_information    : out std_logic_vector(39 downto 0);
      parameter_switching : out std_logic_vector( 3 downto 0)
   );
end pos_proc;
```

O bloco de pós-processamento, bem como o de pré-processamento, possui lógica para interpretar os dados provenientes do corretor. O funcionamento desse bloco e dado pelo sinal *info_erro*, cujo qual informa se o resultado do vetor *syndrome_register* é zero ou não. Se os dados foram completamente corrigidos o vetor *syndrome_register* assume o valor '0' em todas as posições, porém, se houver ao menos uma posição com valor '1', isso serve como indicativo de que a mensagem não pode ser completamente corrigida e os dados disponibilizados no barramento pelo decodificador podem não estar certos ou não foram completamente corrigidos

Uma vez identificada essa situação, o bloco de pós-processamento utiliza a última palavra válida para estimar os dados que deveriam ser enviados ao resto do sistema. Quando essa situação ocorre, o bloco de pós-processamento verifica as posições referentes ao sinal de indicação de troca de parâmetros de transmissão, dentro do vetor de 122 *bits*, que controla a troca dos parâmetros de transmissão do sistema. O sinal de indicação de troca de parâmetros de transmissão pode ser chamado também de *countdown*, assim como descrito anteriormente.

Se o *countdown*, for igual a 15, o bloco de pós-processamento mantém a última mensagem válida sem alterações e envia os dados para o próximo bloco. Se o *countdown* da última mensagem válida estiver em um valor qualquer entre 14 e 1, o bloco de pós-processamento decrementa 1 do *countdown* e envia a mensagem alterada tanto para o próximo bloco como também para o bloco de pré-processamento. Nesse caso, assim como no caso anterior, os sinais *current_information* e *next_information* são repassados sem alterações.

Porém, se o *countdown* da mensagem anterior estiver em 0, o bloco de pós-processamento altera o valor do próprio *countdown* para 15, e os dados contidos em *next_information* são copiados para o sinal *current_information* e enviados para o bloco seguinte assim como para o bloco de pré-processamento.

5 Conclusão

Uma vez concluída a etapa de estudo do sistema de correção de erros descrito em (ARIB, 2005), se deu início a implementação e validação do mesmo e de suas otimizações. Tal qual as otimizações, a implementação só foi possível a partir do uso de técnicas de descrição de sistemas digitais síncronos de alto desempenho e arquiteturas de FPGAs.

No que diz respeito à implementação do código corretor de erros baseado em lógica majoritária, é possível afirmar que tal implementação atende aos requesitos mínimos impostos pela norma de transmissão de TV digital japonesa. A partir da análise dos dados obtidos através de inúmeras simulações com valores aleatórios de erros, e também do uso do código desenvolvido em um transmissor de TV digital prototipado em FPGA, pode-se dizer que os resultados são satisfatórios, uma vez que houve completa compatibilidade com diversos receptores comerciais.

Quanto às otimizações propostas, observou-se que, através da adição dos blocos de pré e pós-processamento, há uma melhora na capacidade de correção do sistema como um todo. A lógica de pré e pós-processamento, aliadas a robustez da modulação DBPSK e ao sistema OFDM de modulação de dados, tornaram o sistema mais tolerante à falhas, fazendo com que mais de 8 *bits* errados pudessem ser recebidos e mesmo assim, corretamente decodificados.

Notou-se também, que, mesmo com a adição de tais blocos, a área de FPGA não foi significativamente alterada, bem como o tempo de processamento, tanto que, para o código decodificador e o resto do sistema, essas otimizações são completamente transparentes, uma vez que toda a análise dos dados do pacote e processamento pelos blocos adicionais ocorre em tempo praticamente nulo garantindo sincronismo e alta velocidade de processamento.

Referências

ARIB. *Transmission System for Digital Terrestrial Television Broadcasting ARIB Standard*. [S.l.], November 2005.

DECASTRO, F. C. C. *Comunicação Digital - Codificação digital*.

DTV. *Entenda a TV Digital*. 2009. [Online: acesso em 4 de novembro de 2009]. Disponível em: <http//www.dtv.org.br>.

HENNESSY, J. L.; PATTERSON, D. A. *Arquitetura de computadores: uma abordagem quantitativa*. [S.l.]: Campus, 2003.

HSU, H. P. *Teoria e Problemas de Comunicação Analógica e Digital*. [S.l.]: Bookman, 2006.

RESENDE, L. E. A. de. *Desenvolvimento de uma ferramenta de análise de desempenho para o padrão de TV Digital ISDB-T*. [S.l.]: PUC-RIO, 2004. Dissertação de Mestrado.

RUSCHEL, O. T. *Princípios da Comunicação Digital*. [S.l.]: EDIPUCRS, 1996.

SMITH, A. S. K. *Microeletrônica*. [S.l.]: Pearson Makron Books, 2005.

WICKER, S. B. *Error Control Systems for Digital Communication and Storage*. [S.l.]: Prentice Hall, 1994.