UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CORREÇÃO DE PCR EM PROCESSADORES DE FLUXOS DE TRANSPORTE MPEG-2

HEITOR JUDISS SAVINO

Manaus - Amazonas
25 de setembro de 2012

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CORREÇÃO DE PCR EM PROCESSADORES DE FLUXOS DE TRANSPORTE MPEG-2

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

HEITOR JUDISS SAVINO
Orientador: Prof. Dr. Eddie Batista de Lima Filho

Manaus - Amazonas
25 de setembro de 2012

Ficha Catalográfica
(Catalogação realizada pela Biblioteca Central da UFAM)

Savino, Heitor Judiss

S267c Correção de PCR em processadores de fluxos de transporte MPEG-2 / Heitor Judiss Savino. - Manaus: UFAM, 2012.
109 f.; il. color.

Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Amazonas, 2012.
Orientador: Prof. Dr. Eddie Batista de Lima Filho

1. TV Digital 2. Métodos de correção de PCR 3. MPEG-2 4. Processamento de imagens – Técnicas digitais I. Lima Filho, Eddie Batista de (Orient.) II. Universidade Federal do Amazonas III. Título

CDU 004.932(043.3)

UNIVERSIDADE FEDERAL DO AMAZONAS

UFAM

FOLHA DE APROVAÇÃO

Correção de PCR em processadores de fluxos de transporte MPEG-2

HEITOR JUDISS SAVINO

D. Sc. EDDIE BATISTA DE LIMA FILHO – Orientador
Universidade Federal do Amazonas

D. Sc. WALDIR SABINO DA SILVA JÚNIOR
Universidade Federal do Amazonas

Ph. D. LUCAS CARVALHO CORDEIRO
Universidade Federal do Amazonas

D. Sc. CELSO BARBOSA CARVALHO
Universidade Federal do Amazonas

Manaus - Amazonas, 25 de setembro de 2012

Dedico esta dissertação a
Deus, fonte de tudo,
meus pais, exemplo de luta,
meu irmão, meu primeiro amigo,
minha noiva, que me faz crescer e sonhar
e meu orientador, exemplo profissional e acadêmico.

Agradecimentos

Dedico meus sinceros agradecimentos:

– primeiramente a Deus, por guiar e iluminar meus passos, dando-me coragem e persistência.

– ao Professor Doutor Eddie Batista de Lima Filho, pela amizade, incentivo, orientação, e por tudo o que me ensinou, colaborou e lutou pelo meu crescimento e formação profissional, pessoal e acadêmica.

– a meus pais, Nestor e Marina, por me tornarem quem sou, guiando meus primeiros passos, a meu irmão Bruno e minha noiva Maria. A estes, pela confiança em mim depositada, incentivo, apoio e por fazerem parte da minha vida.

– à equipe do Laboratório de TV Digital do Centro de Ciência, Tecnologia e Inovação do Pólo Industrial de Manaus – CT-PIM, Cláudio, Anastacio, César, Ribeiro, Daniel, Eddie, André Luiz, Volnei, Carlos Eduardo, Marcelo, George e Cenival, pelo incentivo e suporte prestado durante estes dois anos.

– ao Centro de Ciência, Tecnologia e Inovação do Pólo Industrial de Manaus – CT-PIM, pelo financiamento, espaço físico e oportunidade de realização deste trabalho.

– ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, pelo apoio prestado nas participações em congressos.

– por fim, aos professores e funcionários do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, que possibilitam a mim e a todos os colegas do programa, a oportunidade de formação acadêmica, ampliando nossos horizontes.

Resumo

Atualmente, o fluxo de transporte MPEG-2 é amplamente utilizado como camada de transporte para a multiplexação de programas que carregam áudio, vídeo e dados, em redes de televisão digital. As informações transmitidas são divididas em pacotes de tamanho fixo, multiplexadas no tempo e então enviadas ao receptor. Para que os dados de um mesmo programa sejam apresentados corretamente e em sincronismo (*e.g.* áudio e vídeo), o fluxo de transporte permite a incorporação de bases de tempo, que são utilizadas na sincronização do sistema de recepção. Entretanto, essa informação, conhecida como *Program Clock Reference* (PCR) e inserida periodicamente no feixe multiplexado, pode sofrer erros, observados na forma de *jitter*, devido à cadeia de processamento entre transmissor e receptor. Dessa forma, para se evitar que o erro inserido pelo processamento do fluxo de transporte prejudique a apresentação da informação enviada, métodos de correção de PCR são geralmente utilizados. Este trabalho apresenta um estudo sobre os métodos de correção de PCR mais utilizados na literatura e introduz duas novas contribuições. A primeira consiste em uma estrutura de acesso compartilhado para o bloco de correção, que tem o potencial de proporcionar características reduzidas de complexidade computacional (quantidade de operações aritméticas) e requisitos de hardware, quando utilizada em conjunto com métodos tradicionais de correção de PCR. A segunda consiste em um método inteligente para a correção de PCR, cuja principal característica é a redução do erro inserido pelo processo de correção. A sua arquitetura é baseada na operação cooperativa dos dois principais agentes envolvidos nesse processo: os módulos de adaptação de taxa e de correção de PCR. Apesar de apresentar maior complexidade computacional, o seu desempenho superior facilita o processo de sincronização, executado pelo receptor, o que o torna bastante adequado a sistemas que exigem bases de tempo precisas. As metodologias propostas são utilizadas em conjunto, proporcionando um esquema integrado e eficiente para o processamento de fluxos de transporte MPEG-2.

Palavras-chave: Correção de PCR, TV Digital, MPEG-2, semáforo, fluxo de transporte.

Abstract

Currently, the MPEG-2 transport stream is widely used as multiplexing system for sending audio, video, and data, in digital television networks. The transmitted information is divided into fixed-length packets, which are multiplexed in time and then sent to the receiver. In order to ensure the correct presentation of data related to a given program (*e.g.*, audio and video), the transport stream can provide time bases, which are used for synchronizing the reception system. However, that information, which is known as Program Clock Reference (PCR) and is also periodically embedded in the multiplexed data stream, may suffer from inaccuracies, due to the processing chain between transmitter and receiver, which can be perceived in the form of jitter. In order to prevent that errors caused by the transport stream processing chain, which compromises the proper presentation of the transmitted information, PCR correction methods are normally employed. This work presents a review of the PCR correction methods commonly used in the literature and introduces two new contributions. The first one consists of a shared access structure for the correction block, which has the potential to provide a reduced computational complexity (in terms of arithmetic operations) and hardware requirements, when used with traditional PCR correction methods. The second one devises an intelligent method for PCR correction, whose main feature lies on the reduction of the jitter caused by the correction process. Its architecture is based on the cooperative operation of the main modules involved in this process: the rate adapter and the PCR corrector. In spite of presenting a higher complexity, its superior performance favors the synchronization process, performed by the receiver, which makes it suitable for precise time-bases demanding systems. The proposed methodologies are combined, which provides a complete and efficient framework for processing MPEG-2 transport streams.

Keywords: PCR correction, digital television, MPEG-2, semaphore, transport stream.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Siglas

AAC *Advanced Audio Coding* - Compressão de Áudio Avançada

AVC *Advanced Video Coding* - Compressão de Vídeo Avançada

ASI *Asynchronous Serial Interface* - Interface Serial Assíncrona

ATSC *Advanced Television System Comitee* – Comitê do Sistema Avançado de Televisão

DVB-T *Digital Video Broadcasting – Terrestrial* - Radiodifusão de Vídeo Digital – Terrestre

DTS *Decoding Time Stamp* - Referência de Tempo de Decodificação

ES *Elementary Stream* - Fluxo Elementar

FPGA *Field-Programmable Gate Array* - Arranjo de Portas Programável em Campo

HD *High Definition* - Alta Resolução

ISDB-T *Integrated System Digital Broadcasting – Terrestrial*

ISDB-Tb *Integrated Services Digital Broadcasting – Terrestrial Brazil* - Serviços Integrados de Radiodifusão Digital – Terrestre Brasil

MPEG *Moving Pictures Experts Group* - Grupo de Especialistas em Imagens com Movimento

MPTS *Multiple Program Transport Stream* - Fluxo de Transporte com Múltiplos Programas

NCO *Numerically Controlled Oscillator* - Oscilador Controlado Numericamente

OFDM *Orthogonal Frequency Division Modulation* - Múltiplas Portadoras Ortogonais entre Si

PAT *Program Association Table* - Tabela de Associação de Programa

PCM *Pulse-Code Modulation* - Modulação por Código de Pulsos

PCR *Program Clock Reference* - Referência de Relógio de Programa

PES *Packetized Elementary Stream* - Fluxo Elementar Empacotado

PID *Program Identifier* - Indentificador de Programa

PLL *Phase Locked Loop* - Malha de Detecção de Fase

PMT *Program Map Table* - Tabela de Mapa de Programa

PSI *Program Specific Information* - Informação Específica de Programa

PTS *Presentation Time Stamp* - Referência de Tempo de Apresentação

SBTVD Sistema Brasileiro de Televisão Digital

SD *Standard Definition* - Resolução Padrão

SPI *Synchronous Parallel Interface* - Interface Paralela Síncrona

SSI *Synchronous Serial Interface* - Interface Serial Síncrona

STC *System Time Clock* - Relógio Local do Receptor

TP *Transport Packet* - Pacote de Transporte

TS *Transport Stream* - Fluxo de Transporte

Introdução

O Fluxo de Transporte (*Transport Stream* – TS) MPEG-2 [4] é um formato padronizado para a transmissão de vídeo, áudio e dados associados, comumente utilizado como camada de transporte em sistemas de radiodifusão digitais, como os padrões europeu, americano e japonês.

Um TS MPEG-2 pode passar por vários níveis de processamento antes de ser finalmente transmitido. Por exemplo, é possível a realização de processos de remultiplexação, demultiplexação ou até mesmo um simples rearranjo do conteúdo transportado, que podem causar alterações no posicionamento das referências de tempo carregadas.

Além disso, para a comunicação entre equipamentos de radiodifusão de TV Digital, o padrão europeu provê três tipos de interfaces físicas para a transmissão de dados [5]: a Interface Serial Assíncrona (*Asynchronous Serial Interface* - ASI), a Interface Paralela Síncrona (*Synchronous Parallel Interface* - SPI) e a Interface Serial Síncrona (*Synchronous Serial Interface* - SSI). Em alguns casos, há a necessidade de conversão de um tipo de interface para outra, o que pode causar erros de sincronização, devido à alteração da posição de pacotes com informações relacionadas ao sincronismo.

A Referência de Relógio de Programa (*Program Clock Reference* – PCR) é a informação de tempo utilizada, pelo TS MPEG-2, para a sincronização do receptor, de modo que áudio, vídeo e dados sejam corretamente apresentados. Em resumo, esta informação representa o estado atual do contador de 27 MHz do transmissor e fornece toda a informação de tempo necessária ao sincronismo do sistema. Quando este valor é igual ao do contador do receptor, o sistema está perfeitamente sincronizado; caso contrário, uma malha de detecção de fase (*Phase Locked Loop* - PLL) é usada na adaptação do relógio local, através da diferença entre os valores de PCR [4].

Entretanto, a cadeia de processamento entre transmissor e receptor pode causar desvios na informação de PCR, o que é conhecido como *jitter* [6] e pode comprometer o processo de sincronização. De modo a manter o erro dentro de limites aceitáveis, um módulo de correção de PCR é normalmente empregado, com o objetivo de compensar o desvio devido às novas localizações, no TS, dos pacotes de PCR. O padrão MPEG-2 estabelece um limite máximo de $\pm 500ns$ [4] para o *jitter*, de modo a assegurar o

correto funcionamento do receptor.

Alguns trabalhos na literatura [3, 7, 8, 9, 10, 11] fornecem soluções para a correção de PCR, através da modificação dos seus valores, de modo a compensar as mudanças ocasionadas no módulo de adaptação de taxa. Dois esquemas básicos podem ser identificados: um conjunto de módulos de correção dedicados [3, 7, 12], onde cada contador é associado a um programa no TS e é diretamente carregado com o respectivo valor de PCR, e o esquema baseado em um único contador [8, 9], do qual referências intermediárias são tomadas para se realizar a compensação. Ambos apresentam vantagens e desvantagens, sendo que a melhor escolha depende das características da aplicação desejada.

Este trabalho aborda a correção de PCR, através do estudo das técnicas existentes na literatura e da proposição de novos métodos. Primeiramente, uma nova estrutura de acesso ao módulo de correção, baseada em variáveis de semáforos para o compartilhamento, é apesentada, o que proporciona as principais vantagens presentes nos esquemas de correção tradicionais: um único contador e a possibilidade dos valores de PCR serem corrigidos diretamente por este, evitando-se adições e subtrações. Como consequência, há uma redução no número de operações necessárias durante o processo de correção de PCR e uma minimização dos recursos computacionais necessários (*e.g.* número de células de FPGA).

Até o momento, as soluções propostas apenas modificam os valores de PCR, para que os mesmos reflitam as suas novas posições no TS, porém, nenhuma delas aborda a diferença de fase existente na adaptação de taxa, isto é, os tempos de chegada e saída de pacotes, e a granularidade da correção de PCR, o que é dado pelo período dos gatilhos do relógio de 27 MHz. Como consequência, mesmo quando a correção do PCR é perfeitamente realizada, ainda há um acréscimo no *jitter* de saída.

Com base nisso, propõe-se um novo método de correção de PCR, no qual o algoritmo de adaptação de taxa opera de forma cooperativa com o módulo de correção de PCR, de tal modo que o sistema é capaz de reduzir o *jitter* inserido no TS adaptado. Tal abordagem consiste, na verdade, em um esquema completamente novo para o processamento de Fluxos de Transporte MPEG-2, com a correção de PCR integrada à adaptação de taxa, de modo que pacotes com PCR sejam transmitidos apenas quando o *jitter* previsto for suficientemente pequeno.

O restante do trabalho está organizado da seguinte forma. O Capítulo 1 descreve o padrão de TV Digital e introduz a multiplexação de sinais. No Capítulo 2, aborda-se o padrão MPEG-2 como camada de transporte. O Capítulo 3 descreve a correção de PCR e os métodos tradicionais, destacando suas principais características. No Capítulo 4, a nova estrutura de contador controlado por semáforos é apresentada. Em seguida, no Capítulo 5, a estrutura de correção de PCR integrada à adaptação de taxa é descrita

e, no Capítulo 6, abordam-se os resultados de simulações para os métodos propostos. Finalmente, no Capítulo 7, as considerações finais do trabalho são apresentadas.

Capítulo 1

Revisão da Literatura

O problema da correção de PCR está localizado na área de processamento de Fluxos de Transporte MPEG-2, cujas técnicas são amplamente empregadas em sistemas de televisão digital. Sendo assim, imprecisões nas bases de tempo de programas presentes no TS podem afetar diretamente o funcionamento de receptores de TV Digital, comprometendo as informações enviadas na rede de comunicação. Esta seção aborda a constituição básica do sistema de TV Digital adotado no Brasil, de modo a expor e embasar a necessidade de um esquema eficaz para a correção de PCR.

1.1 TV Digital

Por TV Digital, entende-se o tipo de transmissão de mídia através de pacotes de dados com valores discretos, o que permite a utilização de ferramentas computacionais para o tratamento destes sinais e acarreta uma série de vantagens. Por exemplo, é possível utilizar a compressão para se aumentar a capacidade de transmissão de dados do sistema, melhorando assim a qualidade do conteúdo a ser transmitido. Algoritmos de correção de erros também podem ser utilizados, de modo que a informação recebida seja verificada e corrigida, reduzindo-se a degradação do sinal. A TV Digital também envolve a utilização de técnicas de modulação digital, com uma banda dividida em múltiplas portadoras ortogonais entre si (*Orthogonal Frequency Division Multiplexing* – OFDM) [13, 14], como descrito na norma ABNT/NBR 15601 [1].

Dentre os padrões de TV Digital mais utilizados atualmente, estão o americano (*Advanced Television System Comitee* - ATSC) [15], o europeu (*Digital Video Broadcasting – Terrestrial* - DVB-T) [16] e o japonês (*Integrated System Digital Broadcasting – Terrestrial* - ISDB-T) [17]; no Brasil, adotou-se uma modificação do padrão ISDB-T, denominada ISDB-Tb [1]. É comum, a esses sistemas, a utilização dos algoritmos do padrão MPEG-2 para a compressão de áudio [18] e vídeo [19] e para a formação da

camada de transporte [4].

O padrão do Sistema Brasileiro de TV Digital SBTVD, assim como o japonês, possui canais com largura de banda de 6 MHz, dividida em 13 segmentos. Tal organização permite uma transmissão hierárquica de sinais, cujo objetivo é proporcionar serviços para dispositivos fixos, móveis e portáteis, em um esquema unificado de recepção.

Assim, cada camada de transmissão utiliza um determinado número de segmentos OFDM, que são codificados de forma diferente. Por exemplo, utiliza-se um único segmento para a transmissão do sinal de TV Digital para receptores móveis, como celulares e sistemas multimídia automotivos [1].

Como mostrado na Figura 1.1, o sistema de transmissão do SBTVD está dividido em codificação de dados, multiplexação e codificação de canal. A codificação de dados compreende a compressão de vídeo e áudio [20], que são multiplexados com outros dados a serem enviados, para que seja possível a ocupação de um único caminho de transmissão. A codificação de canal, por sua vez, visa garantir que a informação seja enviada de forma adequada, de modo que o receptor seja capaz de recuperá-la.

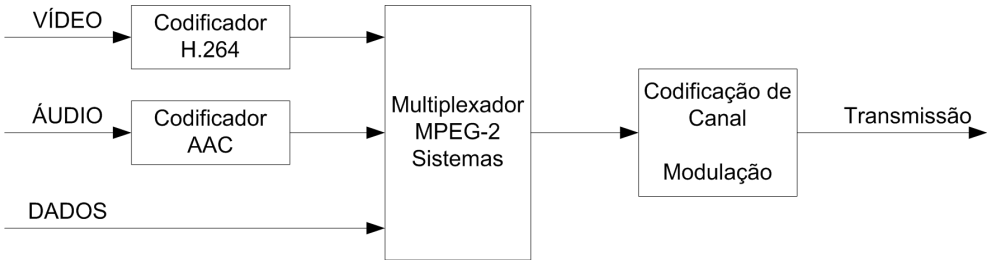


Figura 1.1: Diagrama de formação de sinal do SBTVD [1].

Para a codificação de vídeo [21], nas resoluções padrão (*Standard Definition* - SD) e em alta resolução (*High Definition* - HD), a informação é constituída por um sinal de luminância e dois sinais de crominância [22], quantizados [23] com 8 bits. A codificação é dada segundo a norma MPEG-4/AVC [24], ou seja, utiliza o compressor de vídeo H.264.

Na codificação de áudio [25], o sinal não comprimido está no formato de modulação por código de pulsos (*Pulse-Code Modulation* - PCM), com amostragem a uma taxa de 32 KHz, 44,1 KHz ou 48 KHz e quantizado com 16 bits ou 20 bits. A compressão e a transmissão do áudio são realizadas conforme a norma MPEG-4 [18], utilizando-se a compressão de áudio avançada (*Advanced Audio Coding* - AAC).

Após a etapa de codificação de dados, as informações de TV estão prontas para o empacotamento/multiplexação, de modo a serem finalmente transmitidas. Como a transmissão do sinal é feita de modo sequencial, os pacotes de áudio, vídeo e dados devem ser multiplexados no tempo, de modo a formar um único feixe de dados. Nesta

etapa do sistema de TV Digital [26], comum aos demais padrões, como o DVB-T e o ATSC, utiliza-se o MPEG-2 Sistemas [4], o que resulta no TS MPEG-2.

O feixe de transporte é então enviado ao bloco de codificação de canal, onde ocorre a inserção de dados auxiliares, a codificação contra erros e a modulação, para finalmente ser enviado através do transmissor OFDM.

1.2 Multiplexação de Sinais

Conforme mencionado na seção anterior, antes da etapa de codificação do canal, os fluxos elementares de áudio e vídeo devem ser multiplexados, formando um único feixe de dados, de modo a serem transmitidos ou armazenados.

O MPEG-2 Sistemas [4] define uma estrutura para o Fluxo de Transporte, através da quebra dos pacotes de fluxos elementares, de um determinado programa, em pacotes de tamanho fixo, com 188 bytes. Um programa é definido por um conjunto de vídeos, faixas de áudio e outros dados multiplexados, que possuem uma mesma base de tempo. Esta base é responsável pela temporização entre todas as informações, assim como por proporcionar sincronismo ao decodificador. Um TS MPEG-2 pode transportar vários programas ao mesmo tempo, conforme mostrado na Figura 1.2, e deve disponibilizar a informação necessária para identificar cada um destes, de modo que, no momento da decodificação, o receptor possa filtrar os pacotes referentes ao programa que se deseja apresentar.

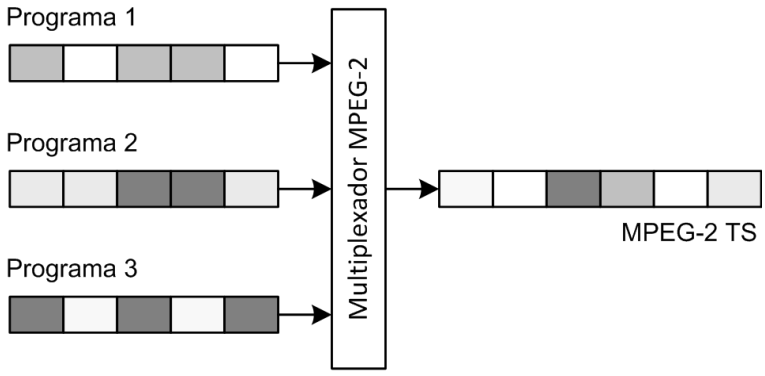


Figura 1.2: Programas multiplexados em Fluxo de Transporte MPEG-2.

Primeiramente, cada pacote do Fluxo de Transporte possui um identificador de pacote (*Packet Identifier* – PID), que é comum a todos os pacotes que carregam uma dada informação. Por exemplo, todos os pacotes de vídeo de um determinado programa são identificados com o mesmo PID.

A informação contendo a estrutura do TS, que possibilita o acesso ao conteúdo de um programa, é denominada Informação Específica de Programa (*Program Specific*

Information – PSI) [4], que é disponibilizada em forma de tabela no corpo de pacotes específicos. A primeira tabela a ser acessada (identificada pelo PID 0) é a Tabela de Associação de Programas (*Program Association Table* – PAT). Esta tabela informa quantos programas estão contidos no TS e apresenta, no seu corpo, uma lista de PIDs, que apontam para pacotes com outras tabelas, contendo informações mais detalhadas a respeito da estrutura de cada programa.

As tabelas apontadas pela PAT são as Tabelas de Mapa de Programa (*Program Map Table* – PMT), que, por sua vez, contém os PIDs dos pacotes de vídeo, áudio e dados, associados ao programa indicado pela PMT. Ainda, esta tabela indica o PID do pacote que contém uma das informações de maior interesse presentes no TS, e que está diretamente relacionada à base de tempo de um programa: a Referência de Relógio de Programa, ou seja, o PCR.

É através do PCR que se recupera a taxa de bits de um Fluxo de Transporte e se realiza o sincronismo do Relógio Local do Receptor (*System Time Clock* - STC). É importante que o STC esteja corretamente sincronizado, isto é, com as mesmas características do relógio no transmissor, pois como este é utilizado durante a decodificação dos dados e no sincronismo entre vídeo e áudio, a sua falha resultaria na apresentação incorreta do conteúdo do TS.

Nesse ponto, evidencia-se a importância da correção de PCR no ambiente de TV digital. Sem uma base de tempo correta, a apresentação do conteúdo dos programas fica comprometida, o que pode resultar na diminuição da qualidade do serviço.

Capítulo 2

A camada de transporte MPEG-2

Neste capítulo, será descrita a formação de pacotes MPEG-2, indicando-se as suas aplicações e o processo de sincronismo do relógio local, de modo a se garantir a correta apresentação do conteúdo multiplexado. Além disso, será mostrado como é medido o *jitter* e como o sistema de recepção é influenciado por este.

2.1 O fluxo de transporte MPEG-2

MPEG é a sigla de *Moving Pictures Experts Group*, que trata da transmissão digital de dados multimídia. O padrão de TV Digital Brasileiro, assim como vários outros padrões utilizados em outros países, adotam as recomendações desenvolvidas pelo MPEG, tanto para compressão de dados (MPEG-4 AVC e AAC) como para a camada de transporte (MPEG-2 TS).

O padrão de Fluxo de Transporte MPEG-2 [4] é aplicado à transmissão e ao armazenamento de mídias, como áudio e vídeo comprimidos e outros tipos de dados, podendo carregar inclusive aplicações interativas. Sendo assim, a estrutura do TS MPEG-2 pode ser descrita independentemente dos dados a serem carregados.

Na Figura 2.1, a formação do Fluxo de Transporte MPEG-2 pode ser visualizada. Inicialmente, os sinais-fonte de áudio e vídeo são comprimidos, pois as suas taxas de dados originais inviabilizariam a transmissão por um sistema comercial. Por exemplo, a taxa de um vídeo HD, sem compressão, é de aproximadamente $1,5$ Gbit/s. Após o processo de compactação, esse valor cai para menos de 15 Mbit/s. Os sinais comprimidos, chamados de fluxos elementares (*Elementary Stream* – ES), assim como os outros tipos de dados, são divididos em pacotes de tamanho variável, formando os Fluxos Elementares Empacotados (*Packetized Elementary Stream* – PES). Logo após, esses pacotes são quebrados em unidades menores, com tamanho fixo de 188 bytes, e multiplexados, o que dá origem ao Fluxo de Transporte [27].

Além da multiplexação dos dados de um programa, o Fluxo de Transporte também deve fornecer informações adicionais para a sua recepção, como as referências de relógio e as tabelas de identificação e acesso aos programas.

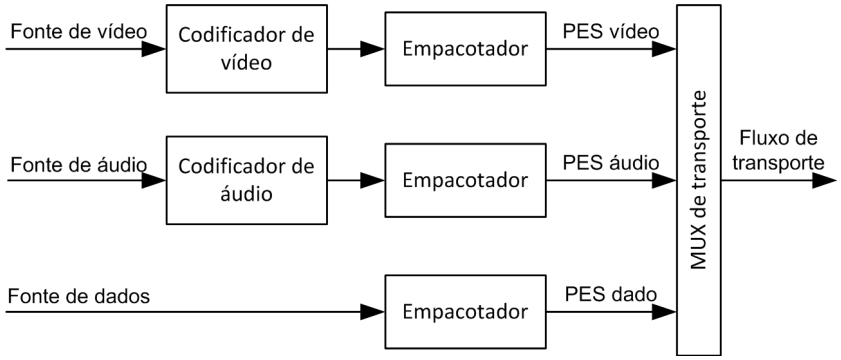


Figura 2.1: Diagrama de formação do Fluxo de Transporte MPEG-2.

2.1.1 Fluxos Elementares Empacotados – PES

A formação do fluxo de transporte inicia com a composição dos pacotes PES, que apresentam tamanho variável (até 64 kB) e são formados diretamente após a etapa de compressão. Há, então, relacionado a cada tipo de dado, um tipo de fluxo: vídeo, áudio ou outros dados.

A estrutura básica dos pacotes PES [2] pode ser vista na Figura 2.2. O pacote é composto basicamente por um cabeçalho fixo de 6 bytes, seguido pelo seu corpo. Há também a possibilidade de um cabeçalho opcional, de tamanho variável. Os dados úteis, resultantes da compressão, são carregados no corpo de cada pacote.

O cabeçalho de um pacote PES inicia com um prefixo de 3 bytes, fixado em 00 00 01, que é usado para se detectar o início de cada pacote. Em seguida, 1 byte é usado para definir o tipo de dado contido no corpo do pacote, indicando se este contém vídeo, áudio ou outros. Por último, no cabeçalho, tem-se 16 bits que indicam o tamanho do corpo do pacote. Cada pacote de PES pode ter um tamanho de até 64 kbytes, no entanto, em casos específicos, o seu comprimento pode assumir valores maiores que este. Para isto, basta que o valor do tamanho de pacote seja zero. Por consequência, o decodificador de MPEG usará outros mecanismos para encontrar o final do pacote, como o prefixo de início.

Após os 6 bytes iniciais de cabeçalho, segue-se uma extensão de cabeçalho opcional, que é utilizada para carregar informações detalhadas, de acordo com o tipo de PES transmitido. As informações mais específicas estão contidas em 11 *flags*, que indicam os dados presentes nos campos opcionais. Por exemplo, há informações como a Referência de Tempo de Apresentação (*Presentation Time Stamp* – PTS) e a Referência de Tempo

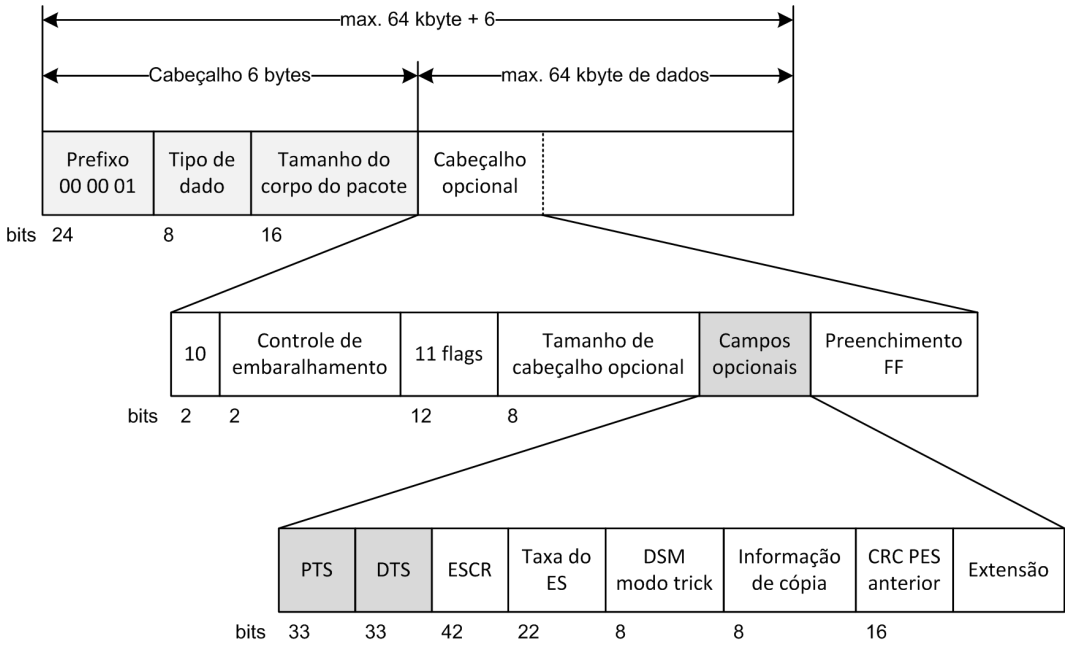


Figura 2.2: Estrutura de um pacote PES.

de Decodificação (*Decoding Time Stamp* – DTS), que são de grande importância para a sincronização entre áudio, vídeo e dados relacionados. Após os campos opcionais, o cabeçalho opcional pode ser completado com bytes de preenchimento (*stuffing*).

Como um único contador é usado na codificação, todas as referências de tempo, que indicam o tempo de apresentação e decodificação de áudio e vídeo, são geradas a partir do mesmo. Assim, as informações contidas nos campos opcionais do pacote de PES, indicadas na seção anterior (PTS e DTS), estão relacionadas à referência de relógio de programa (PCR) [4]. É a presença desta base de tempo que possibilita os recursos necessários para a sincronização do receptor e apresentação adequada do conteúdo transmitido.

2.1.2 A formação do TS MPEG-2

Apesar de bastante flexível, a estrutura de PES não é adequada à transmissão e difusão dos vários programas multiplexados, em um ambiente ruidoso [4]. Com este fim, definiu-se o TS MPEG-2, que apresenta suporte para a transmissão de múltiplos programas em um único feixe de dados, de forma a atender os requisitos de redes de TV digital. Para isto, os pacotes PES são divididos em pacotes menores, de tamanho fixo, contendo 184 bytes de corpo e 4 bytes de cabeçalho, o que forma pacotes de 188 bytes, denominados de Pacotes de Transporte (*Transport Packet* - TP).

Um fluxo de transporte com vários programas (*Multiple Program Transport Stream* – MPTS) pode conter vários programas. Um programa pode consistir em um ou mais

sinais de vídeo e áudio, como, por exemplo, as imagens de várias câmeras em diferentes ângulos durante a transmissão de um jogo de futebol, e o áudio em diferentes idiomas.

Para cada programa, há codificadores MPEG, que comprimem os fluxos elementares, e multiplexadores MPEG, que geram a estrutura de PES e os dividem em pacotes de TS. A taxa de dados de cada programa pode variar, de acordo com o conteúdo das mídias em cada momento. Sendo assim, como o multiplexador deve compartilhar o enlace de dados e se adaptar às variações de taxa dos fluxos elementares, tal operação acaba resultando em uma multiplexação estatística [28]. Os fluxos de transporte de cada programa são então combinados num único feixe de dados multiplexados de TS MPEG-2, como visto anteriormente na Figura 1.2, podendo facilmente alcançar taxas da ordem de 40 Mbit/s. É importante notar que as taxas de dados de cada programa podem variar, porém, para o ambiente de TV Digital, a taxa de dados total do TS deve permanecer constante. Isto é feito através da introdução de pacotes nulos, que são descartados no receptor. Juntamente com os programas, várias tabelas também são enviadas, indicando o conteúdo transmitido no feixe, de modo que o receptor possa acessar todos os dados disponibilizados pelo TS.

2.1.3 Pacotes de TS MPEG-2

Um fluxo de transporte MPEG-2 consiste em pacotes de 188 bytes, sendo 184 bytes de corpo de dados e 4 bytes de cabeçalho fixo. O corpo do mesmo é formado pelo conteúdo obtido a partir da quebra dos pacotes de PES, como mostrado na Figura 2.3, podendo conter informação relacionada a áudio, vídeo ou outros dados.

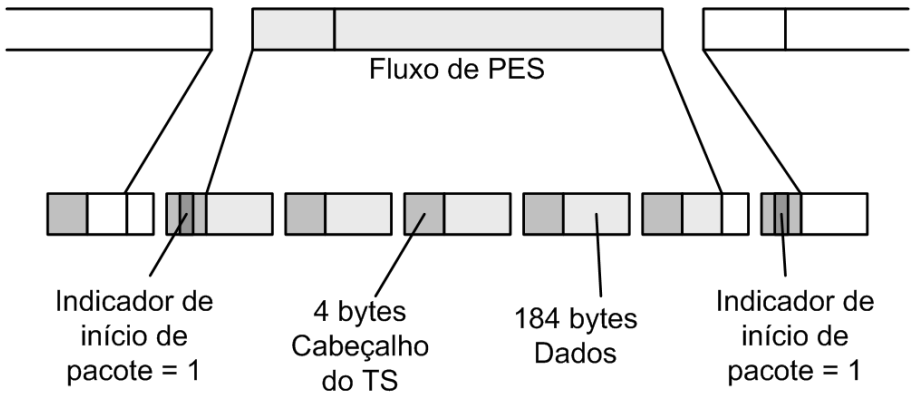


Figura 2.3: Formação dos pacotes de TS a partir da quebra de pacotes de PES [2].

O primeiro pacote de TS de um determinado PES, formado a partir de sua quebra, possui um indicador de início de pacote, e o último pode ser preenchido com dados nulos, de modo a completar os 188 bytes contidos em cada pacote.

A estrutura de um pacote de TS MPEG-2 pode ser vista na Figura 2.4. No cabeçalho fixo de 4 bytes, estão contidas informações importantes para a recepção dos pacotes. A primeira delas é o byte de sincronismo, que possui valor fixo 47_{hex} e é espaçado de 188 bytes do próximo byte de sincronismo no fluxo de transporte, indicando o início do pacote seguinte. Vale a pena mencionar que é possível haver outro byte com o mesmo valor 47_{hex} em algum lugar no TS, no corpo de dados do pacote. De acordo com o padrão MPEG-2, a sincronização, no decodificador, ocorre quando são detectados cinco bytes de sincronismo, distantes 188 bytes entre si, de modo que o início de cada pacote seja conhecido.

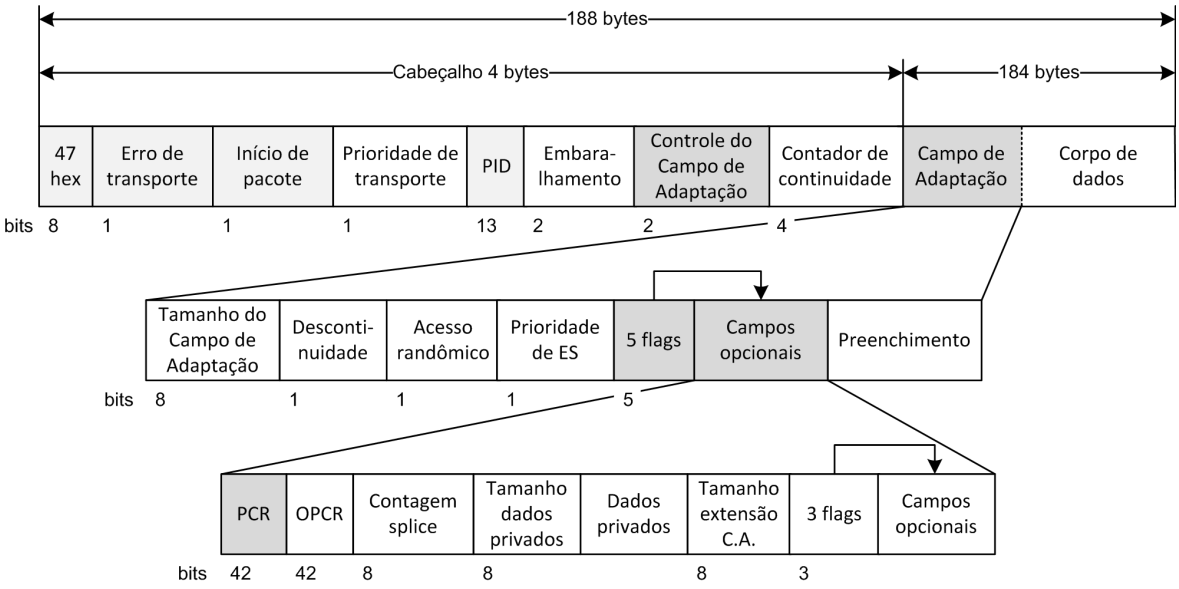


Figura 2.4: Estrutura dos pacotes de TS MPEG-2.

O bit que segue após o byte de sincronismo é o indicador de erro de transporte. Este bit é usado para indicar erro durante a transmissão. O demodulador é responsável pelo seu controle, sendo que o mesmo recebe valor 1 sempre que não é possível recuperar o conteúdo do pacote, através dos mecanismos de correção disponíveis.

Outra informação muito importante, contida no cabeçalho do TS, é o identificador de pacote (*Packet Identifier* – PID). O PID possui tamanho de 13 bits e identifica o conteúdo carregado no corpo do pacote. O seu valor, em hexadecimal, é referenciado nas tabelas transmitidas pelo TS, para que seu conteúdo possa ser adequadamente acessado e filtrado.

Em alguns casos, pode ser necessário transmitir informações adicionais no cabeçalho do pacote de TS. Assim, o cabeçalho pode ser estendido para o corpo do pacote, o que recebe o nome de Campo de Adaptação. Este é habilitado nos 4 bytes de cabeçalho, nos chamados bits de controle de campo de adaptação. No Campo de Adaptação, há

outras informações de grande importância habilitadas pelos 5 *flags* dentro deste, como é o caso do PCR, que é responsável pela sincronização do relógio local do decodificador.

Um dos mecanismos utilizados na correção de possíveis erros de transmissão é o código de correção de erros chamado Reed Solomon [29]. Para isto, aos 188 bytes do pacote de TS, adicionam-se 16 bytes de redundância, o que dá origem a pacotes com 204 bytes. Esses 16 bytes são um tipo de proteção a erros, calculados a partir do próprio conteúdo carregado pelo pacote de TS. Com isso, o demodulador é capaz de corrigir até 8 bytes errôneos, por pacote [2]; quando houver mais de 8 erros na recepção de um dado pacote, o indicador de erro de transporte é habilitado. Nestes casos, o pacote não é processado e o decodificador MPEG deve mascarar o erro, o que em muitos casos pode ser notado como um tipo de *bloco* na apresentação do vídeo.

2.2 A referência de relógio de programa - PCR

O PCR, que é uma cópia do valor do relógio de 27 MHz do codificador, no instante do seu envio, é uma informação inserida no campo de adaptação de alguns pacotes e tem grande importância para a sincronização do receptor. O PCR é formado, no total, por 42 bits [4], sendo 33 bits chamados de base, o que corresponde a 1/300 da frequência do STC, e 9 bits de extensão, na mesma frequência do relógio de 27 MHz. Sendo assim, o valor de PCR é dado por:

$$PCR(i) = PCR\_base(i) \times 300 + PCR\_ext(i) \tag{2.1}$$

onde $PCR\_base(i)$ é o valor da base e $PCR\_ext(i)$ é o valor da extensão, em um dado pacote i. Dado que o PCR é um retrato do valor do relógio do codificador, a base e a extensão podem ser calculadas como

$$PCR\_base(i) = (valor\_STC \div 300) \bmod 2^{33} \tag{2.2}$$

$$PCR\_ext(i) = valor\_STC \bmod 300. \tag{2.3}$$

Deste modo, entende-se a frequência do termo PCR_base como $27MHz \div 300$, isto é, $90kHz$.

A norma MPEG-2 Sistemas [4] indica que os valores de PCR devem aparecer num período máximo de 100 ms, de modo a garantir a correta sincronização do relógio de 27 MHz do receptor. Entretanto, as normas dos sistemas de televisão digital aconselham valores menores que 40 ms [30].

2.3 Sincronização

O sincronismo é realizado para se garantir que os conteúdos multiplexados (*e.g.* áudio, vídeo e dados) sejam corretamente decodificados e apresentados. A parte de sistemas do MPEG-2 é responsável por esta tarefa.

Em resumo, a sincronização consiste na recuperação do relógio do codificador. Quando esse objetivo é atingido, o receptor é capaz de reproduzir as mesmas características do STC do transmissor, no instante da codificação de um dado conteúdo. Para que essa tarefa seja executada, o demultiplexador, presente no sistema de recepção, seleciona os dados de PCR, diretamente do TS enviado pelo sistema de demodulação. Isso é possível devido ao conhecimento do PID dos pacotes que carregam tal informação, o que é obtido através das tabelas de PSI [31], presentes no TS.

O processo de sincronização do relógio, por meio dos valores de PCR, está mostrado na Figura 2.5. Para que o relógio do decodificador seja similar ao do codificador, o receptor utiliza um ajuste baseado em PLL, que adapta um Oscilador Controlado Numericamente (*Numerically Controlled Oscillator* - NCO). No momento em que um novo PCR chega ao decodificador, seu valor é comparado ao valor atual do relógio local. A diferença, ou erro, será um número em unidades de 27 MHz, que é processado por um filtro passa-baixas e então aplicado ao NCO.

Deste modo, os desvios do relógio de 27 MHz podem ser compensados através dessa comparação com o PCR atual, pois se os mesmos forem divergentes, o valor corrente é copiado para o STC e a taxa é ajustada pelo PLL, gerando o processo de estabilização do relógio local.

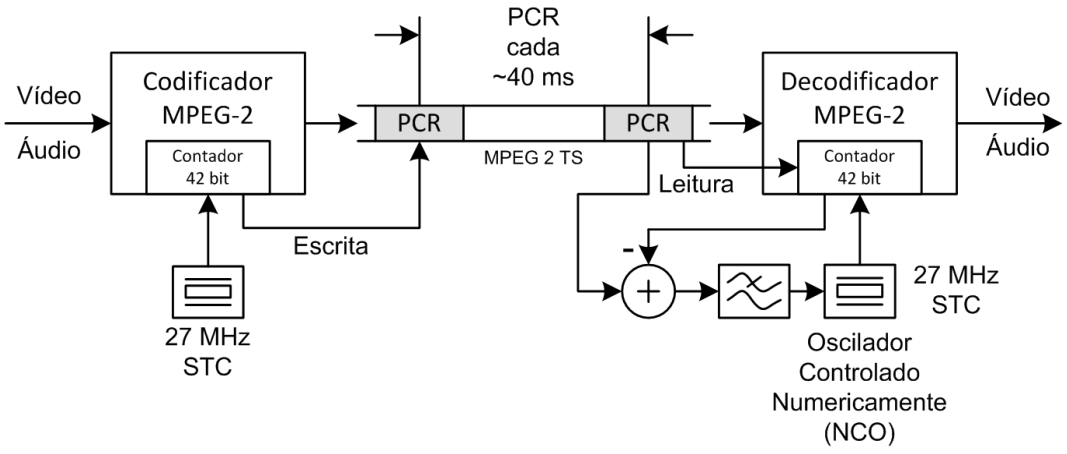


Figura 2.5: Sincronização através do PCR [2].

A recuperação do relógio, no decodificador, é uma tarefa comum para a exibição de qualquer programa existente no TS, dado que cada um pode ter sua própria base de tempo, que é compartilhada entre todos os seus fluxos elementares empacotados

(PES). É devido a isso que valores de PCR não são inseridos no instante da geração dos pacotes de PES, mas sim na camada de multiplexação do TS [2].

2.4 Imprecisões de relógio em sistemas baseados em MPEG-2

O relógio de 27 MHz, sincronizado com os valores de PCR extraídos do TS, pode apresentar desvios em sua frequência nominal, o que é conhecido como *jitter* e deve ser cuidadosamente controlado, de modo a se manter o STC estável. Em resumo, o *jitter* representa o erro existente no valor de PCR recebido, em relação ao esperado.

Como o valor de PCR é retirado de um relógio de 27 MHz, a taxa original do feixe de dados ($Taxa\_TS$), em bytes/s, pode ser estimada como

$$Taxa\_TS = \frac{(i'' - i') \times 27MHz}{PCR(i'') - PCR(i')}, \tag{2.4}$$

onde i'' e i' são os índices dos bytes contendo o último bit da sua respectiva base de PCR e $PCR(i'')$ e $PCR(i')$ são os valores completos das referências de relógio anterior e atual, que são dadas pela Equação (2.1).

Por outro lado, se a taxa média do TS for conhecida, é possível estimar o valor de PCR com

$$PCR\_estimado = \frac{N\_bytes \times 27MHz}{Taxa\_TS} + PCR(i''), \tag{2.5}$$

onde N_bytes é o número de bytes entre i'' e i'.

Finalmente, o *jitter* de PCR é calculado por

$$jitter(i) = \frac{PCR(i) - PCR\_estimado}{27MHz}, \tag{2.6}$$

ou seja, a diferença entre o valor de PCR lido do pacote i ($PCR(i)$) e o valor de PCR estimado ($PCR\_estimado$).

O padrão MPEG-2 Sistemas [4] define o limite máximo aceitável para o *jitter* como $\pm 500ns$. Isto se refere ao erro oriundo da posição que o valor de PCR ocupa no TS, não considerando o *jitter* introduzido pela cadeia de transmissão do fluxo de dados.

O *jitter* existente em uma base de tempo pode ter diversas fontes, como erros em algoritmos de codificação ou processamento de TS, arredondamentos de contagem de bytes ou gatilhos do relógio do sistema, reposicionamentos dos pacotes com PCR, devido à remultiplexação, e a própria adaptação de taxa, através da inserção de pacotes nulos, no TS. Como indicado na seção anterior, sempre que houver o reposicionamento

de pacotes com PCR, é necessário efetuar a correção dos respectivos valores de base de tempo, de modo que estes reflitam as suas novas posições no TS.

2.5 Consequências do *jitter*

Os valores de PCR, quando apresentam erros ocasionando o *jitter* de PCR, podem ocasionar diversas consequências, devido ao fato de estarem diretamente relacionados ao processo de sincronismo de relógio de $27MHz$. A falha natural, devido ao *jitter*, é a falta de estabilidade e o desvio da frequência do STC, o que altera as taxas de amostragem derivadas a partir deste relógio, como é o caso das taxas de amostragem de luminância (e.g. $27MHz \div 2 = 13,5MHz$) e de crominância (e.g. $27MHz \div 4 = 6,75MHz$), utilizadas em saídas de vídeo composto [31].

Algumas indicações de que o *jitter* de PCR está muito grande podem ser vistas como perda de cor em componentes de vídeo, falhas na apresentação dos quadros de vídeo e *blocagem* da imagem [2].

Capítulo 3

A correção de PCR em processadores de TS MPEG-2

Neste capítulo, será descrito o processo de correção de PCR quando há alteração, no TS, da posição de pacotes com PCR. Serão abordados os dois métodos mais utilizados: o método convencional, com contadores dedicados, e o método da compensação. Ainda, será abordada uma nova estrutura utilizando acumuladores. A análise desses métodos servirá de base para a compreensão das metodologias propostas, que são descritas nos Capítulos 4 e 5.

3.1 Processadores MPEG-2

A cadeia de processamento de um TS MPEG-2 pode ocasionar erros de PCR, devido ao reposicionamento de pacotes com referências de tempo, o que ocasiona *jitter* de PCR e prejudica o processo de sincronização do relógio local de 27 MHz.

Entre os tipos de processadores que causam o deslocamento desses pacotes, estão os remultiplexadores, demultiplexadores, conversores de interface [5] e adaptadores de taxa. Para se manter o *jitter* dentro do limite aceitável de $\pm$500 ns, é geralmente necessário empregar um módulo de compensação de PCR, cujo objetivo é refletir, no valor de PCR, a nova posição do pacote de transporte, dentro do TS.

A Figura 3.1 mostra o reposicionamento de pacotes de PCR quando são inseridos pacotes nulos, em um processo de Adaptação de Taxa. Neste exemplo, como a taxa de bits é incrementada, o tempo de transmissão de um pacote se torna menor. Logo, o processador de TS deve ocupar os vazios no feixe de dados, com pacotes nulos. Sempre que há pacotes a serem transmitidos, os mesmos são enviados na oportunidade mais próxima, porém, se o *buffer* de pacotes estiver vazio, ou menor que um número mínimo de pacotes, o processador envia pacotes nulos pela interface de saída. Deste modo,

nota-se, pela Figura 3.1, que há um reposicionamento de dados no TS, fazendo com que a distância temporal entre pacotes contendo informação de PCR seja alterada.

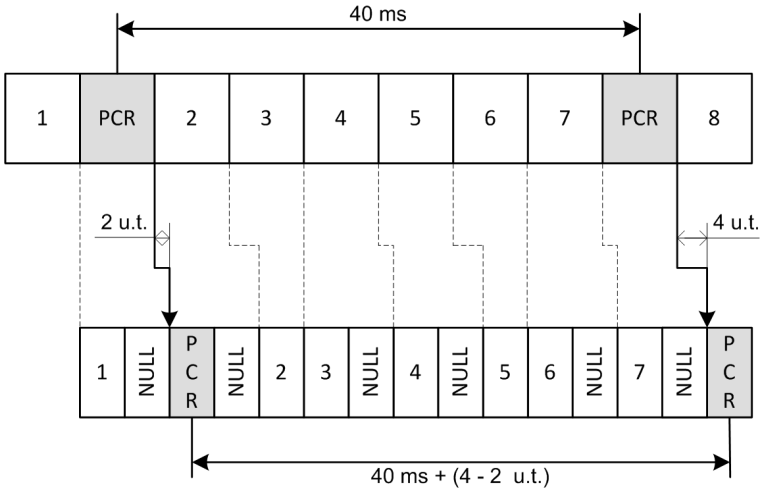


Figura 3.1: Reposicionamento de PCR por Adaptação de Taxa.

No exemplo da Figura 3.1, mostra-se como o tempo entre os pacotes com PCR foi incrementado em 2 unidades de tempo (u.t.). Se, por exemplo, estas duas unidades de tempo neste processador forem correspondentes a 400 períodos do relógio de $27MHz$, isto corresponderá a um erro de

$$Erro\ inserido = \frac{400}{27MHz} = 14,8\mu s, \tag{3.1}$$

o que é muito maior que os 500 ns aceitáveis por norma. Isto faz com que a correção de PCR seja extremamente necessária, de modo que o processo de sincronismo não seja prejudicado.

A Figura 3.2 mostra o resultado do *jitter* de PCR, quando nenhum processo de correção é realizado. Vários picos próximos ou maiores que 100 us podem ser visualizados, o que está fora da margem aceitável e pode comprometer o processo de sincronização.

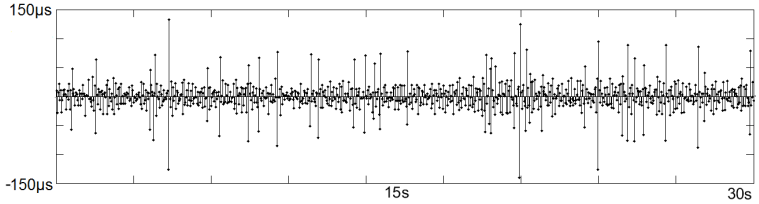


Figura 3.2: Jitter de PCR em Adaptação de Taxa sem correção de PCR.

3.2 Correção de PCR em processadores MPEG-2

Nos casos em que os processadores MPEG-2 alteram o TS, ocasionando o reposicionamento de pacotes com PCR e por consequência *jitter*, a fim de corrigir os valores de PCR, normalmente adiciona-se um termo de compensação aos valores de PCR originais, tal como

$$PCR_{corr} = PCR + \Delta PCR_{comp}, \tag{3.2}$$

onde PCR_{corr} é o novo valor de PCR, PCR é o valor original e ΔPCR_{comp} é o termo de compensação. Esta ideia geral é a base para os métodos descritos a seguir.

3.2.1 Método dos Contadores Dedicados

No método dos contadores dedicados, a correção é feita pela substituição do valor de PCR dos pacotes, levando-se em consideração o tempo que este passa no processo de adaptação de taxa [7], de acordo com o termo ΔPCR_{comp} da Equação (3.2).

O primeiro passo, nesta abordagem, consiste em identificar e extrair a informação de PCR, dos pacotes com referências de relógio. Seu valor é então carregado em um contador de 27 MHz associado ao programa do qual o pacote faz parte. Assim que a adaptação de taxa é concluída e o pacote é agendado para envio, o conteúdo do contador referente à base de tempo do pacote é recuperado e usado para substituir o valor de PCR original.

O padrão MPEG-2 Sistemas [4] permite que o TS contenha mais de um programa, incluindo diferentes bases de tempo, o que é comumente conhecido como MPTS. Devido a isso, uma solução intuitiva para o problema da correção de PCR, em MPTS, seria utilizar um número de contadores de 27 MHz igual ao número de bases de tempo diferentes.

O sistema de correção geral, que utiliza um contador dedicado para cada uma das bases de tempo, detectadas nos respectivos programas contidos no TS a ser processado, pode ser visto no diagrama da Figura 3.3.

A principal desvantagem deste método é que muitos contadores podem ser necessários (*e.g.*, muitos recursos de FPGA usados para criar os contadores), dependendo do número de programas no TS de entrada. Além disso, como cada contador é dedicado a um programa específico, a maior parte dos contadores permanece livre quando há poucos pacotes com PCR no *buffer*, o que acarreta recursos inutilizados.

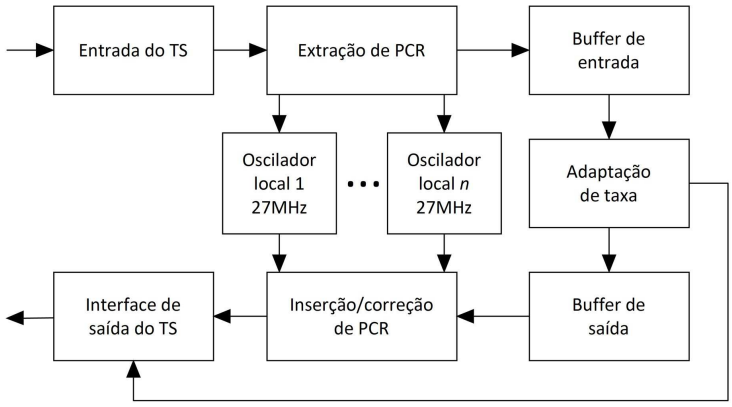


Figura 3.3: Diagrama em blocos do método dos contadores dedicados.

3.2.2 Contadores com Janelas Deslizantes

Esta abordagem observa o fato de que a maioria dos TSs disponibilizados comercialmente apresenta poucos pacotes com PCR próximos entre si [3], de tal modo que contadores não utilizados do método descrito na Seção 3.2.1 podem ser compartilhados.

Para se implementar este método e promover a reutilização dos contadores, duas janelas deslizantes são empregadas, uma para a entrada de PCR e outra para a saída. O sistema de janelas deslizantes, para um sistema com quatro contadores, está representado na Figura 3.4. Quando o primeiro pacote contendo informação com PCR chega ao processador, este é carregado no contador 1, no próximo gatilho de relógio, com o respectivo valor de PCR incrementado de 1 ($PCR + 1$). Em seguida, a janela deslizante é rotacionada para a posição 2 (sentido anti-horário) e, quando o próximo pacote com PCR for carregado, o contador 2 é assinalado para armazenar o novo valor de PCR. Há também uma janela deslizante controlando o processo de saída de pacotes com PCR, que é usada para se identificar o contador que contém a referência de tempo correta para o pacote que está sendo enviado, de acordo com a ordem de chegada deste. Quando o respectivo valor é recuperado do contador, a janela deslizante é também rotacionada, indicando o contador associado ao próximo pacote.

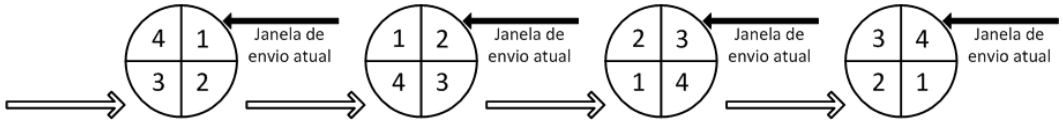


Figura 3.4: Sistema de Janelas Deslizantes com quatro contadores [3].

A vantagem desta abordagem esta na redução do número de contadores alocados, sendo este um processo dependente do comportamento do TS. Ainda, o sistema de janelas deslizantes consiste em poucos bits, com adições em módulo-2, o que o torna

simples de ser implementado. É importante notar que o número de contadores é dependente do número de bases de tempo e não do número de programas, devido ao fato de que múltiplos programas poderem compartilhar a mesma base de tempo.

3.2.3 O Método da Compensação

O método da compensação [8, 9] é baseado em um único contador de 27 MHz, utilizado na correção da informação de PCR de todos os programas em um MPTS, o que economiza um grande número de recursos. A abordagem também opera acrescentando-se aos valores de PCR o tempo que seus pacotes passam no processador, operando do modo descrito a seguir, segundo a Figura 3.5.

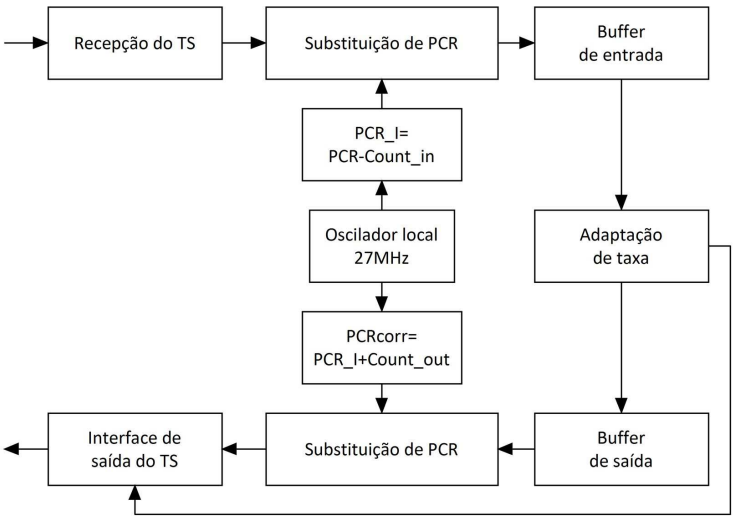


Figura 3.5: Diagrama em blocos do método da compensação.

Quando um pacote contendo PCR é detectado, o estado atual do contador ($Count\_in$) é tomado e subtraído do respectivo valor de PCR, sendo que o resultado desta operação (PCR_I), então, substitui o PCR original no pacote. Quando este pacote é agendado para ser enviado pelo sistema, o PCR armazenado é mais uma vez alterado: o estado atual do contador ($Count\_out$) é novamente tomado e adicionado ao valor do PCR. Desta forma, cada valor de PCR original é substituído por um novo valor de referência, calculado como

$$PCR\_I(i) = PCR(i) - Count\_in(i) \tag{3.3}$$

$$PCR_{corr}(i) = PCR\_I(i) + Count\_out(i) \tag{3.4}$$

$$PCR_{corr}(i) = PCR(i) + \Delta PCR_{comp}(i), \tag{3.5}$$

onde $\Delta PCR_{comp}(i) = Count\_out(i) - Count\_in(i)$ é o termo de compensação para $PCR(i)$.

O método da compensação pode ser visto como uma correção em duas etapas, na qual um contador de 27 MHz mede o tempo de processamento de pacotes no módulo de adaptação de taxa e compensa o atraso variável sofrido pelos mesmos. Este método é simples e utiliza uma quantidade de recursos mínima, porém, apresenta a desvantagem de realizar várias adições de 42 bits, o que aumenta a carga computacional do sistema.

3.2.4 O Método dos Acumuladores

O método dos acumuladores [12] é mais recente que os anteriores e não utiliza contadores de 27 MHz. Na verdade, o módulo de correção de PCR utiliza um acumulador gatilhado pela taxa de saída de bytes do TS. O acumulador é inicializado com o valor de PCR extraído do primeiro pacote com PCR detectado na interface de entrada do processador MPEG-2. Após isso, o sistema passa a acumular com uma taxa dada por

$$taxa\_acum = \frac{STC\_freq}{taxa\_saida} \tag{3.6}$$

onde $taxa\_acum$ representa o incremento que o acumulador realizará no valor de PCR carregado, a cada byte transmitido na interface de saída, a uma frequência dada por $taxa\_saida$, que é a taxa de saída em bytes por segundo, e STC_freq é a frequência do relógio do sistema, ou seja, 27 MHz.

É importante notar que esse método inicializa o acumulador uma única vez, ou quando descontinuidades de PCR são detectadas, de modo que, em si, o processo consiste na reestampa do PCR, com base no primeiro valor carregado. No caso de múltiplos programas, vários acumuladores devem ser utilizados. O diagrama em blocos está mostrado na Figura 3.6.

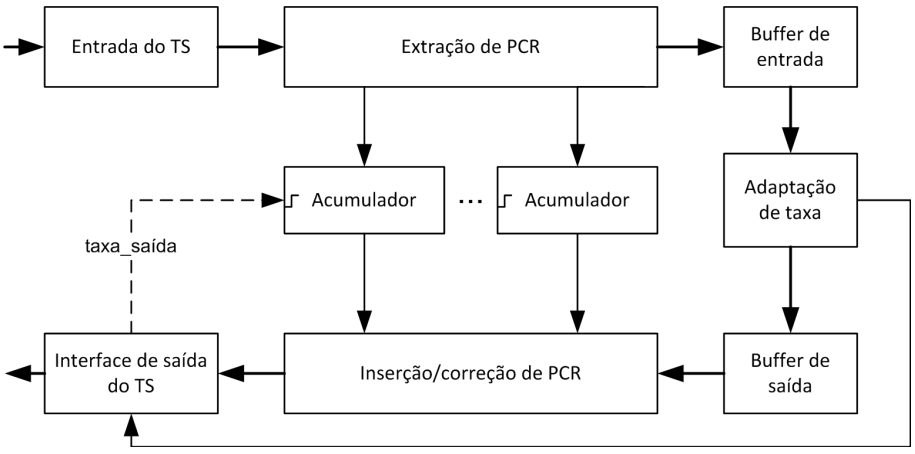


Figura 3.6: Diagrama em blocos do método dos acumuladores.

3.3 Considerações sobre a correção de *jitter*

Cada método descrito na Seção 3.2 apresenta características mais adequadas para determinados tipos de aplicações. As suas principais vantagens e desvantagens são destacadas na Tabela 3.1 e comentadas a seguir.

Tabela 3.1: Vantages e desvantagens dos métodos tradicionais.

Método	Vantagens	Desvantagens
Contadores Dedicados [7]	Desempenho rápido. Nenhuma operação aritmética.	Muitos recursos alocados e em grande parte do tempo inutilizados.
Contadores com Janelas Deslizantes [3]	Desempenho rápido. Nenhuma operação aritmética. Compartilhamento de recursos.	Tamanho da janela e número de contadores ainda permanece dependente do TS.
Compensação [8, 9]	Um único contador compartilhado.	Aumento do número de operações aritméticas.
Acumuladores [12]	Nenhum contador	Reestampa, altera a base de tempo. Maior granularidade e, por consequência, maior erro inserido.

O método dos Contadores Dedicados (ver seção 3.2.1) é o método mais rápido, dado seu menor custo computacional, pois não realiza nenhum tipo de operação aritmética e seu mecanismo de correção recai completamente sobre os contadores. No entanto, dado o número de programas desconhecido, devem ser alocados contadores suficientes, de modo a se atender qualquer TS. Por exemplo, isto implica em grande número de células de FPGA, que por vezes podem estar inutilizadas.

A estratégia para se reduzir os recursos necessários vem com o método dos Contadores com Janelas Deslizantes (ver seção 3.2.2), que, através de uma lógica simples, consegue compartilhar os contadores existentes. No entanto, o número de contadores permanece arbitrário, dependendo do TS a ser processado.

Com o método da Compensação (ver seção 3.2.3), é iniciado o uso de operações aritméticas para a correção de PCR, de modo que um único contador é empregado. Isto causa um aumento na carga computacional do processador, mas torna possível sua aplicação a qualquer TS, com qualquer quantidade de programas.

A abordagem dos métodos dos acumuladores mostra que o emprego de contadores pode ser descartado, através do uso da própria taxa de saída de bytes do processador, porém, a maior granularidade acarreta em maiores erros. Além disso, o mecanismo de correção não é atualizado a cada novo valor de PCR, tratando-se de uma reestampa,

o que pode comprometer as bases de tempo originais. Vale ressaltar que este método pode ser implementado com acumuladores dedicados ou de forma similar ao método da compensação.

Logo, é possível comentar que a escolha de um método de correção de PCR deve levar em consideração as características descritas na Tabela 3.2, como os recursos necessários, a complexidade computacional e o erro inserido pelo mecanismo de correção, aproveitando-se das características dos TSs a serem processadores e de estratégias de manejo de recursos.

Tabela 3.2: Características desejadas dos métodos.

Característica	*Descrição*
Redução de recursos necessários	Busca-se reduzir a quantidade de recursos necessário, em termos da quantidade de contadores ou acumuladores, que ocupam células de FPGA, evitando a alocação de recursos ociosos.
Complexidade computacional	Trata da redução do número de operações aritméticas exigidas pelo módulo de correção. A redução é comparada com a quantidade de somas exigidas pelo método da compensação [8, 9] (Seção 3.2.3).
Jitter inserido	O *jitter* inserido pelo processador de TS deve estar dentro dos níveis aceitáveis pela norma. Deste modo, busca-se um método de correção que seja capaz de controlar os erros de PCR inseridos, minimizando o impacto do reposicionamento dos pacotes no TS.

Capítulo 4

Uma estrutura de semáforos para a correção de PCR em processadores de TS MPEG-2

Pelo exposto no capítulo anterior, os métodos tradicionais apresentam vantagens e desvantagens, e a escolha de uma abordagem específica depende do hardware e do objetivo da aplicação final: poder de processamento ou diminuição dos recursos (*e.g.* número de células de FPGA). Se os contadores dedicados forem usados, haverá uma grande quantidade de recursos alocados para o módulo de correção de PCR, no entanto, eles podem não ser sempre utilizados, gerando recursos ociosos. Isto é devido a dois aspectos principais: é possível que diferentes programas usem a mesma base de tempo e o fato de que pacotes com PCR não estão normalmente próximos um do outro. Por outro lado, quando o método da compensação é utilizado, há um maior impacto na complexidade computacional, devido às operações de soma realizadas.

Sendo assim, uma abordagem que tirasse vantagem do comportamento típico do TS e utilizasse um único contador com um número mínimo de operações aritméticas (ou até mesmo nenhuma operação), seria então a melhor opção. Esta é a inspiração para uma nova metodologia aplicada à correção de PCR, conhecida como correção por um Contador Controlado por Semáforos.

4.1 A correção por um Contador Controlado por Semáforos

Este método apresenta as principais vantagens dos dois esquemas tradicionais: apenas um contador de 27 MHz é utilizado para se computar os ajustes de informação de PCR (extraída do respectivo pacote de TS), como no método da Compensação (ver Seção

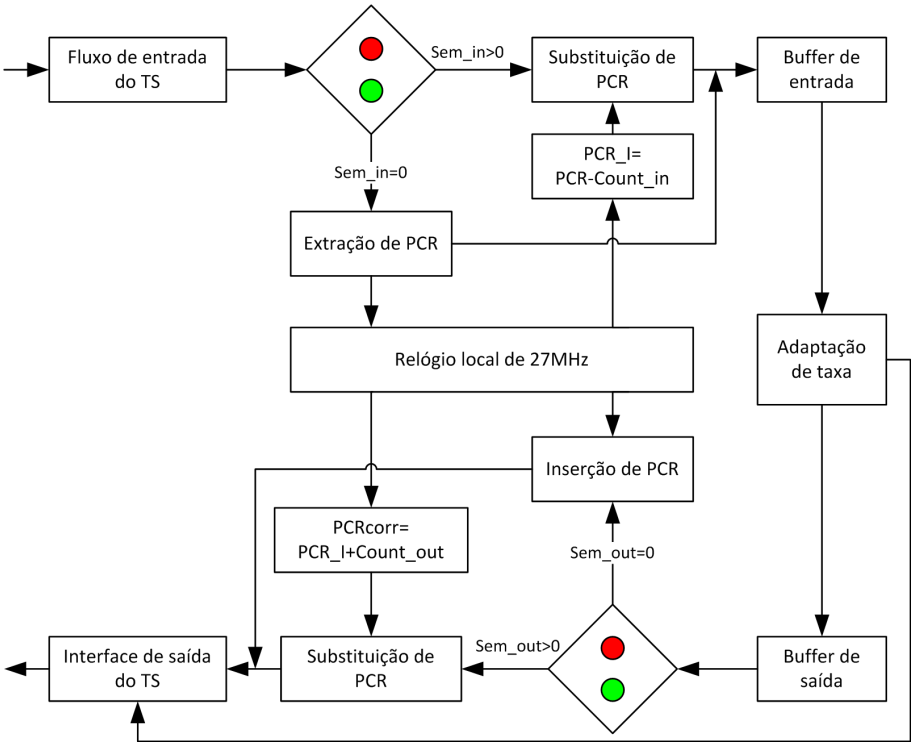


Figura 4.1: O diagrama de blocos do método do Contador Controlado por Semáforos.

3.2.3), e valores de PCR podem ser corrigidos sem nenhuma adição, sendo carregados diretamente no registro do contador, como no método dos Contadores Dedicados (ver Seção 3.2.1). O diagrama em blocos do método proposto está ilustrado na Figura 4.1.

4.2 A arquitetura do sistema de correção

De uma forma geral, os TSs normalmente encontrados em aplicações comerciais apresentam poucos pacotes com valores de PCR próximos, e este fato pode ser explorado para se reduzirem o número de operações aritméticas, quando há apenas um contador disponível no sistema. Com este propósito, foi incorporado um novo recurso para se controlar o acesso ao contador local de 42 bits, que passa a apresentar dois estados: o estado livre, que é identificado quando o contador está disponível para ser carregado diretamente, com um valor de PCR extraído de um pacote de entrada, e o estado ocupado, que indica que o registro do contador já foi carregado com um valor de PCR. Neste último, no lugar do contador dedicado para se compensar o atraso, o sistema cria uma nova referência de tempo, que incorpora o estado atual do contador ao valor original de PCR e serve como marcador para a correção final.

4.3 A estrutura de semáforos

O recurso utilizado para indicar o estado atual do contador e fornecer o acesso compartilhado é baseado em uma estrutura de semáforos, de variáveis de valor inteiro, como mostrado na Figura 4.1. Dois semáforos são utilizados: um para a entrada (Sem_in) e outro para a saída (Sem_out), ambos inicializados em 0. Sempre que um pacote contendo informação de PCR é enviado para o *buffer* de entrada, Sem_in é incrementado; do mesmo modo, quando um pacote contendo PCR deixa o módulo de adaptação de taxa, Sem_out é incrementado. Se Sem_in e Sem_out apresentarem o mesmo valor, o que é sempre avaliado quando uma operação sobre Sem_out é finalizada, ambos são reinicializados com o valor 0. Isto significa que todos os pacotes com dados de PCR, que chegaram ao *buffer* de entrada, já foram enviados, e o contador se encontra no estado livre.

Como exposto no último parágrafo, o contador de 27 MHz se encontra no estado livre quando Sem_in é igual a 0, o que significa que, quando um pacote contendo PCR for recebido no *buffer* de entrada, seu valor de PCR poderá ser diretamente carregado no registro do contador. Em seguida, Sem_in é incrementado, o que força o contador a entrar no estado ocupado ($Sem\_in > 0$); a partir deste momento, os próximos valores de PCR serão substituídos por uma nova referência, dada pelo estado atual do contador, até que Sem_in seja novamente 0. O novo valor de PCR é então dado por

$$PCR\_I(i) = PCR(i) - Count\_in(i), \tag{4.1}$$

onde $PCR(i)$ é o valor original do PCR e $Count\_in(i)$ é o estado atual do contador, no momento da chegada do pacote.

Quando o primeiro pacote deixa o sistema, Sem_out apresenta o valor 0, o que indica que o valor corrigido deve ser recuperado diretamente do registro do contador; isto é seguido de um incremento no valor de Sem_out. Quando os pacotes restantes, que chegaram enquanto o primeiro ainda ocupava o contador (Sem_in maior que 0), estão para ser enviados, seus valores de PCR são novamente alterados, de acordo com

$$PCR_{corr}(i) = PCR\_I(i) + Count\_out(i) - \Delta_{PCR}, \tag{4.2}$$

onde $PCR\_I(i)$ é o valor de PCR modificado quando o pacote chegou, $Count\_out(i)$ é o estado atual do contador, quando o pacote é enviado, $PCR_{corr}(i)$ é o valor de PCR corrigido e Δ_{PCR} é uma constante baseada na taxa de dados, usada por todos os valores de PCR. Tal fator está também incorporado ao bloco *inserção de PCR*, na Figura 4.1, para Sem_out 0.

Se os semáforos não alcançarem o mesmo valor, quando os pacotes restantes forem

enviados, mesmo após o primeiro pacote já ter sido processado, o contador local permanecerá ocupado; os próximos pacotes de PCR serão então corrigidos com referências intermediárias, já que não podem haver interferências no registro do contador enquanto ainda houver, no adaptador de taxa, pacotes com referências baseadas em seu registo.

A principal característica desta nova abordagem consiste no fato de que, se houver poucos pacotes com PCR próximos ou se estes estiverem muito afastados um do outro, então as operações aritméticas serão escassas e o sistema poderá realizar todas as tarefas necessárias de maneira muito rápida. Mesmo em casos extremos, ao menos uma das correções, em uma sequência de pacotes muito próximos, não exigirá operações aritméticas.

O módulo de correção pode ser visto como duas máquinas independentes, com uma sendo executa na entrada e outra na saída do sistema. O algoritmo 1 é utilizado pela interface de entrada do sistema, como descrito nos parágrafos anteriores.

```
1 início
2 |  Verificar Sem_in;
3 |  se Sem_in > 0 então
4 |  |  PCR(i) ← PCR(i) - Count_in(i);
5 |  senão
6 |  |  Registro do Contador ← PCR(i);
7 |  fim
8 |  Incrementar Sem_in;
9 fim
```

Algoritmo 1: Semáforo de entrada.

O Algoritmo 2 é utilizado pela interface de saída, sendo avaliado sempre que um pacote deve ser enviado pelo sistema.

4.4 A compensação final da correção com Δ_{PCR}

Dado que outras referências do TS são dependentes do PCR, como a Referência de Tempo de Apresentação (PTS) e a Referência de Tempo de Decodificação (DTS), Δ_{PCR} é importante para evitar a quebra da cadeia de sincronização, causadas por grandes deslocamentos no valor de PCR. Como já foi mencionado, Δ_{PCR} é dependente da taxa da dados e pode ser considerado proporcional ao intervalo de tempo que os pacotes com PCR experimentam, durante o processo de correção. No entanto, o seu valor também é dependente da estratégia de *buffer*, isto é, a maneira como os pacotes são organizados para serem transmitidos. Por exemplo, uma estratégia bastante simples é assegurar que existem ao menos N pacotes armazenados antes de se realizar a transmissão. Num caso extremo, se apenas um pacote com PCR chegar pela interface de entrada, pacotes

```
1  início
2  |  Verificar Sem_out;
3  |  se Sem_in > 0 então
4  |  |  PCR(i) ← PCR(i) + Count_out(i) - Δ_PCR;
5  |  senão
6  |  |  PCR(i) ← Registro do Contador - Δ_PCR;
7  |  fim
8  |  Incrementar Sem_out;
9  |  se Sem_out = Sem_in então
10 |  |  Sem_out ← 0 ;
11 |  |  Sem_in ← 0;
12 |  fim
13 fim
```

Algoritmo 2: Semáforo de saída.

nulos serão enviados, até que outros $N-1$ pacotes sejam armazenados no *buffer*. O tempo decorrido, neste caso, é dado por

$$IPT = \frac{188 * (N-1)}{IN\_RATE}, \tag{4.3}$$

onde IPT é o tempo de entrada de pacotes (*Input Packets Time*), em segundos, e IN_RATE é a taxa de transporte da entrada, em bytes/segundo. Assim que todos os pacotes necessários tiverem chegado, o primeiro pacote é então enviado. No entanto, se os novos pacotes terminaram de chegar justamente no início da transmissão de um novo pacote de saída, pelo menos este processo terá que ser finalizado, antes que outros pacotes sejam enviados. O atraso adicional é dado por

$$OPT = \frac{188}{OUT\_RATE}, \tag{4.4}$$

onde OPT é o tempo de saída de pacote (*Output Packet Time*), em segundos, e OUT_RATE é a taxa de transporte da saída, em bytes/segundo. O intervalo de tempo de correção total, em média, pode ser dado pelas Equações (4.3) e (4.4), como

$$CTI_{exp} = IPT + OPT, \tag{4.5}$$

onde CTI_{exp} é o intervalo de tempo de correção total esperado (*Expected Correction Time Interval*).

Baseado no que foi apresentado e no fato de que Δ_{PCR} deve ser expresso em número de gatilhos de 27MHz, é possível obter, da Equação (4.5), a expressão

$$\Delta_{PCR} = \lfloor \alpha * CTI_{exp} * SYS\_CLK\_FREQ \rfloor , \tag{4.6}$$

onde SYS_CLK_FREQ possui valor 27 MHz e α é uma constante de escalonamento. α pode ser visto como um nível de ajuste de Δ_{PCR}, com o qual o seu valor pode ser reduzido ou aumentado, dependendo do TS e do deslocamento médio pretendido para as correções. Entretanto, o seu valor é geralmente assume um valor fixo do intervalo entre $0,5$ e $1,0$.

4.5 Análise da estrutura proposta

O esquema proposto pode ser visto como um esquema completo de correção de PCR em MPTSs, fornecendo uma máquina de ajustes de referências de tempo e uma estrutura de acesso compartilhado. Quando comparada às abordagens tradicionais, esta nova metodologia possibilita uma implementação mais leve e compacta, dado que apenas um contador é usado e o número de operações aritméticas é consideravelmente reduzido.

A quantidade de recursos computacionais necessários para cada esquema é variável e depende da implementação. Mesmo na literatura disponível, onde se encontram estas informações para os métodos abordados no Capítulo 3 [9, 3, 11], os números apresentados são bastante variáveis e estão relacionados ao tipo de tecnologia utilizada para a realização do sistema e execução dos testes. No entanto, em princípio, o número de unidade lógicas de FPGA da estrutura de semáforos comprometeria menos do que o necessário para um somador de 42 bits ou um contador de 27 MHz. Assim, é possível predizer que o método proposto, na grande maioria de suas aplicações, exigirá menos células lógicas que o método da compensação [9], com dois ou mais somadores, e que o método dos contadores dedicados [3], quando forem empregados mais de dois contadores.

Com relação ao número de operações aritméticas exigidas para a correção de PCR, é possível estimar um valor esperado. Por exemplo, o método da compensação realiza duas operações para cada valor de PCR, o que leva a

$$NAP_{comp} = 2 \sum_{i=0}^{N-1} \left(\frac{1}{PCR\_RATE(i)} \right) , \tag{4.7}$$

onde NAP_{comp} é o numero de operações aritméticas, por segundo, no método da compensação, N é o número de bases de tempo e $PCR\_RATE(i)$ é a taxa de chegada de valores de PCR de uma dada base de tempo i, em segundos. Para o contador

controlado por semáforos, uma formulação matemática não é tão simples, dado que o número de operações aritméticas depende das taxas de PCRs, da proximidade no TS, das diferentes bases de tempo e do atraso nos pacotes de PCR.

A taxa de chegada de pacotes de PCR, para cada base de tempo, pode ser vista como um evento probabilístico independente, que é bem sucedido se uma referência chega durante o intervalo de correção, com uma probabilidade dependente da taxa de PCR. Se mais de uma referência chegar ao módulo de correção de PCR durante este intervalo, operações aritméticas serão realizadas. Além disso, se um pacote com PCR chegar ao final do intervalo de correção, a próxima referência, no próximo intervalo, resultará novamente em operações aritméticas. Sendo assim, o intervalo de tempo, durante o qual a chegada de novas referências resultam em operações, será considerado $2CTI_{exp}$. Supondo que há N diferentes bases de tempo e $2(n-1)$ operações ocorrem quando $n > 1$ pacotes chegam durante o intervalo de tempo considerado, o número esperado de operações é dado por

$$NAP_{sem} = \sum_{n=2}^{N} \frac{(n-1)}{CTI_{exp}} \sum_{A \in F_n} \prod_{l \in A} p_l \prod_{k \in A^c} (1 - p_k), \tag{4.8}$$

onde NAP_{sem} é o número de operações aritméticas, por segundo, no método do contador controlado por semáforos, N é o número de diferentes bases de tempo, F_n é o conjunto de todos os subconjuntos de n bases de tempo, que podem chegar durante o intervalo $2CTI_{exp}$, p_i é a probabilidade de uma referência pertencente à base de tempo i chegar durante o mesmo intervalo, que pode ser aproximado por $2CTI_{exp}/PCR\_RATE(i)$, e A^c é o complemento de A.

Como exemplo, um TS com 8 bases de tempo diferentes, com taxas de 24 ms em duas bases e 33 ms nas outras, adaptando a taxa de 24,1 Mbps para 43 Mbps, resultaria em um $NAP_{comp} = 530,3030$ e um $NAP_{sem} = 11,7913$ operações por segundo. Isto mostra que apenas $2,22\%$ das operações, necessárias no método da compensação, seriam exigidas no método dos semáforos.

Comparado com a Tabela 3.2, os resultados obtidos são descritos na Tabela 4.1.

Tabela 4.1: Características obtidas pelo método do semáforos.

Característica	*Descrição*
Redução de recursos necessários	Atingido, apenas um contador é necessário.
Complexidade computacional	Atingido, redução do número de somas em até 97,8%.
Jitter inserido	Não houve ganhos.

Capítulo 5

A estrutura de correção de PCR integrada à adaptação de taxa

Os métodos tradicionais apresentados no Capítulo 3, assim como a abordagem do Capítulo 4, quando corretamente implementados em um processador de TS, como um remultiplexador [32] ou um demultiplexador, são capazes de corrigir os valores de PCR adequadamente, evitando a quebra da cadeia de sincronização. Entretanto, as soluções propostas apenas modificam os valores de PCR, para que estes reflitam as suas novas posições no TS. Sendo assim, nenhuma delas aborda o descasamento entre a adaptação de taxa e a granularidade da correção de PCR, o que é dado principalmente pela relação entre a interface de saída, a de entrada e a frequência do relógio de 27 MHz. Devido a isso, mesmo quando a correção de PCR é realizada corretamente, ainda há um acréscimo inerente de *jitter*, na saída.

Com base nisso, é possível supor que, se o algoritmo de adaptação de taxa operar de forma cooperativa com o módulo de correção de PCR, o sistema será capaz de reduzir o *jitter* inserido no TS adaptado. Esta é a inspiração para o novo método proposto neste capítulo, que é baseado em uma nova estrutura com correção de PCR integrada à adaptação de taxa, de modo que pacotes com PCR sejam transmitidos apenas quando o *jitter* previsto for suficientemente pequeno.

5.1 O *jitter* inserido na correção de PCR

De modo a se entender como ocorrem erros durante a correção de PCR, analisa-se um exemplo de adaptação da taxa de um TS, com pacotes de 188 bytes, que tem a sua taxa convertida de 12 Mbps para 48 Mbps. Como a correção de PCR é dada por um relógio local de 27 MHz, um pacote passaria, dentro do adaptador, um número TP_{saida}

de períodos de *clock*, dado por

$$TP_{saida} = \frac{8 \cdot STC\_freq}{taxa\_saida} \cdot L\_pacote, \tag{5.1}$$

onde STC_freq é a frequência do relógio local, ou seja, 27 MHz, $taxa\_saida$ é a taxa de saída, em Mbps, definida pelo processador de TS, e L_pacote é o tamanho do pacote de TS MPEG-2, ou seja, 188 ou 204 bytes, o que depende da presença dos bytes de paridade Reed-Solomon.

Assim, no exemplo analisado, pela Equação (5.1), o número de períodos de *clock* durante o envio de um pacote seria

$$TP_{saida} = \frac{8 \cdot 27MHz}{48Mbps} \cdot 188 = 846. \tag{5.2}$$

Como a distância entre dois valores de PCR em um TS é sempre um número inteiro de pacotes [4], pelo valor de TP_{saida} de 846, obtido na Equação (5.2), o processo de correção de PCR compreenderia um número inteiro de contagens do relógio de 27 MHz. Logo, se o *jitter* na entrada for muito pequeno ou 0, dependendo da implementação, é bastante plausível que este permaneça pequeno.

Em outro cenário, se a conversão fosse de 12 Mbps para 20 Mbps, o número de períodos de *clock* por pacote ficaria

$$TP_{saida} = \frac{8 \cdot 27MHz}{20Mbps} \cdot 188 = 2030,4. \tag{5.3}$$

Dado que apenas a parte inteira 2030 é incrementada no PCR, isto levaria a um erro de acúmulo de contagem de $0,4$ períodos de *clock*. Caso NP_{saida} pacotes de saída sejam enviados, o erro de correção $erro_{correcao}$ é dado por

$$erro_{correcao} = \frac{(\delta clock\_saida \cdot NP_{saida}) \bmod 1}{STC\_freq}, \tag{5.4}$$

onde $\delta clock\_saida$ é o descasamento devido à parte não inteira de TP_{saida}, ou seja,

$$\delta clock\_saida = TP_{saida} \bmod 1. \tag{5.5}$$

Por exemplo, supondo-se que 27 pacotes sejam enviados, isso resultaria em um erro de correção de PCR, dado pela Equação (5.4), de

$$erro_{correcao} = \frac{(0,4 \cdot 27) \bmod 1}{27MHz} \approx 30ns. \tag{5.6}$$

Além disso, se também houver erro na próxima correção de PCR, isso poderá causar

uma diferença de dois ou até mesmo três períodos de *clock* de 27 MHz, no intervalo entre dois valores de PCR. Desta forma, se vários dispositivos com tal comportamento processarem um dado TS, o *jitter* resultante poderá prejudicar o processo de sincronização.

No entanto, se 35 pacotes forem enviados entre esses dois pacotes de PCR consecutivos, o erro resultante tenderá a permanecer próximo de 0, dado que

$$erro_{correcao} = \frac{(0,4 \cdot 35) \bmod 1}{27MHz} = 0ns. \tag{5.7}$$

Mesmo que o relógio do transmissor não seja exatamente 27 MHz, a diferença é normalmente bastante pequena, podendo-se assumir um *jitter*, na saída, bastante reduzido.

Outra observação interessante, que pode ser facilmente percebida, consiste no fato de que o momento em que o valor de PCR é carregado para correção pode não estar sincronizado com o relógio de 27 MHz. Sendo assim, tal comportamento pode resultar em mais uma fonte de *jitter*, com potencial para comprometer ainda mais a correção de PCR.

Da análise apresentada, depreende-se que, se o módulo de adaptação de taxa estiver ciente do mecanismo de correção de PCR e operar de forma conjunta, ajustando o número de pacotes enviados antes do pacote contendo a informação de PCR, o *jitter* no TS de saída pode ser mantido próximo do *jitter* presente no TS de entrada.

5.2 A correção integrada utilizando contadores

Esta seção descreve a estratégia de correção de PCR integrada à adaptação de taxa, quando são empregados contadores locais de 27 MHz para se compensar os valores de PCR. Tal abordagem envolve um estimador de erros nas interfaces de entrada e saída e mostra como o conhecimento do erro, que é inerente ao processo de adaptação, pode ser utilizado na redução do *jitter* inserido, quando a correção de PCR é feita de modo integrado à adaptação de taxa.

5.2.1 O atraso da entrada com relação ao relógio de 27 MHz

Dado um TS com taxa de bits constante, o intervalo de tempo $TP_{entrada}$ entre dois pacotes consecutivos, em períodos de *clock* de 27 MHz, pode ser dado por

$$TP_{entrada} = \frac{8 \cdot STC\_freq}{taxa\_entrada} \cdot L\_pacote, \tag{5.8}$$

sendo $taxa\_entrada$ a taxa de bits estimada do TS.

Assim como mostrado para TP_{saida}, o valor de $TP_{entrada}$ também pode indicar um número não inteiro, $\delta clock\_entrada$, dado por

$$\delta clock\_entrada = TP_{entrada} \bmod 1. \tag{5.9}$$

Isto implica que, cada vez que um pacote contendo PCR for carregado no *buffer* de entrada, o início da contagem do relógio local (relacionado ao termo ΔPCR_{comp} da Equação (3.2)) não estará necessariamente sincronizado com a chegada do pacote, consistindo em uma fonte de erro na correção do valor de PCR.

A Figura 5.1 mostra a representação do atraso de entrada. Nesta, o Pacote 1 é recebido entre dois gatilhos do relógio local, de modo que, quando for dado o primeiro gatilho após a sua chegada, indicando que um período de *clock* se passou desde que o valor de PCR foi carregado, será contada também a parte sombreada da figura. No entanto, para o Pacote 2, este erro é nulo, visto que tal pacote chega em sincronismo com o contador. Este erro é inerente ao processo de correção e está presente em todos os métodos citados anteriormente.

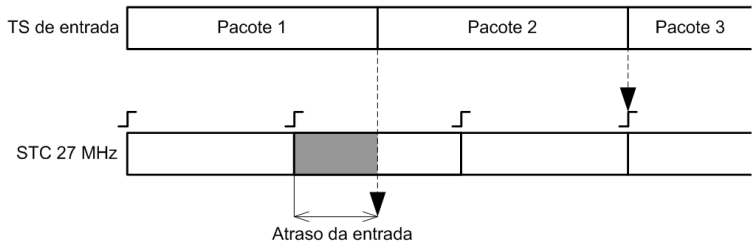


Figura 5.1: Atraso de entrada pela diferença de fase entre entrada de pacotes e o gatilho do contador.

Conhecendo-se a taxa de entrada do TS, é possível estimar a diferença de fase entre a chegada do pacote e o gatilho do contador. Por exemplo, com uma taxa de entrada de 20 Mbps, o tempo entre a chegada de um pacote e outro consecutivo, $TP_{entrada}$, como dado na Equação (5.8), é de

$$TP_{entrada} = \frac{8 \cdot 27MHz}{20Mbps} \cdot 188 = 2030,40, \tag{5.10}$$

isto é, $2030,4$ períodos de 27 MHz. Sendo assim, cada vez que um pacote é carregado no *buffer* de entrada, a diferença $\delta clock\_entrada$, entre o gatilho de *clock* e o instante da chegada do pacote, é aumentada em $0,4$ períodos.

O adaptador de taxa, então, opera contando o número de pacotes recebidos, pela interface de entrada, entre PCRs consecutivos, através de um contador de pacotes controlado pela taxa de entrada. Logo, cada vez que um pacote com PCR é carregado

no $buffer$, o adaptador de taxa toma o valor desse contador, $NP_{entrada}$, para calcular o atraso inserido no sincronismo. Em seguida, o contador é zerado para que a contagem seja reiniciada. O atraso de entrada é então dado por

$$\Delta entrada = (NP_{entrada} \cdot TP_{entrada}) \bmod 1. \tag{5.11}$$

Por exemplo, para 326 pacotes, $\Delta entrada$ seria de

$$\Delta entrada = (2030,4 \cdot 326) \bmod 1 = 661910,4 \bmod 1 = 0,4, \tag{5.12}$$

ou seja, $0,4$ períodos em relação ao gatilho do relógio de 27 MHz.

5.2.2 O atraso da saída com relação ao relógio de 27 MHz

A interface de saída do processador de TS também contribui para o *jitter*, através da diferença entre o gatilho do contador de 27 MHz e o instante em que o pacote é enviado. Com uma taxa de dados constante, o atraso por pacote pode ser calculado pela parte não inteira de TP_{saida} da Equação (5.1), o mesmo $\delta clock\_saida$ da Equação (5.5), aqui repetidas por conveniência:

$$TP_{saida} = \frac{8 \cdot STC\_freq}{taxa\_saida} \cdot L\_pacote \tag{5.1}$$

e

$$\delta clock\_saida = TP_{saida} \bmod 1. \tag{5.5}$$

Assim, por exemplo, para uma taxa de saída de 25 Mbps, o valor de $\delta clock\_saida$ seria de $0,32$, dado que um pacote na saída lavaria um tempo de

$$TP_{saida} = \frac{8 \cdot 27MHz}{25Mbps} \cdot 188 = 1624,32, \tag{5.13}$$

isto é, $1624,32$ períodos de *clock* de 27 MHz.

Vale ressaltar que é desejável que o tempo de permanência de um pacote no adaptador de taxa, desde o instante em que o mesmo é carregado no *buffer* de entrada até o momento em que é enviado pela interface de saída (representado por ΔPCR_{comp} da Equação (3.2)), seja representado por um número inteiro de períodos de *clock* de 27 MHz. Com isso, é possível inserir o menor *jitter* possível, conforme mostrado na Seção 5.1.

Assim, um processo similar ao cálculo da diferença entre o gatilho do contador de 27 MHz e o instante da chegada de um pacote, na interface de entrada, pode ser utilizado

para a interface de saída. Desta forma, contam-se os pacotes enviados entre PCRs, representado por NP_{saida}, sendo que o contador é zerado toda vez que um pacote é enviado. Com isso, é possível estimar a diferença inserida entre o gatilho do contador de 27 MHz e o instante em que um pacote é enviado, fazendo-se

$$\Delta saida = (NP_{saida} \cdot TP_{saida}) \bmod 1. \tag{5.14}$$

Por exemplo, para 408 pacotes, o atraso seria de

$$\Delta saida = (1624,32 \cdot 408) \bmod 1 = 662722,56 \bmod 1 = 0,56, \tag{5.15}$$

ou seja, $0,56$ períodos em relação ao gatilho do relógio de 27 MHz.

5.2.3 A estratégia de correção integrada

No início deste capítulo, foi mostrado que, se o envio de pacotes resultar em um número não inteiro de períodos de *clock*, tal fato pode dar origem a uma fonte adicional de *jitter*. Como a correção de PCR está relacionada à quantidade de gatilhos do contador de 27 MHz, que um pacote passa no adaptador de taxa, é interessante fazer com que essa quantidade de tempo seja o mais próximo possível de um inteiro. A meta fundamental da correção de PCR integrada à adaptação de taxa é estimar o *jitter* introduzido quando pacotes com PCR são enviados, de modo a agendar a sua transmissão para o momento em que o *jitter* resultante for mínimo, ou seja, o tempo no qual o número de períodos de *clock* de 27 MHz seja próximo de um número inteiro.

Com base nisso, quando um pacote contendo informação de PCR no seu campo de adaptação chega ao *buffer* de entrada, toma-se o valor do contador de pacotes de entrada $NP_{entrada}$, que indica a quantidade de pacotes que chegaram desde o último PCR (ver seção 5.2.1), de modo a se calcular a diferença de fase entre o gatilho do contador de 27 MHz e a chegada de pacotes na interface de entrada. Este valor, chamado de $\Delta entrada$, é dado pela Equação (5.11). Assim, quando for necessário enviar este pacote, o módulo de adaptação de taxa tomará o valor do contador de saída (número de pacotes que foram enviados desde o último PCR, ou seja, NP_{saida}) para agendar o seu envio, fazendo com que o erro inserido entre o gatilho do contador de 27 MHz e o instante da saída do pacote ($\Delta saida$) seja o mais próximo possível do que foi medido na entrada, isto é, $\Delta saida \approx \Delta entrada$.

Conforme o cenário apresentado na seção 5.2.1, o atraso inserido na entrada seria de $\Delta entrada = 0,4$ períodos. Além disso, no caso de o adaptador de taxa enviar o pacote conforme a seção 5.2.2, ele compensaria com um atraso inserido na saída igual a $\Delta saida = 0,56$ períodos. Desse modo, poder-se-ia dizer que o pacote foi enviado com

um erro de contagem dado por

$$\Delta SE = \Delta saida - Esaida \tag{5.16}$$

sendo ΔSE a diferença entre $\Delta saida$ e o valor esperado na saída $Esaida$, que neste caso é o mesmo valor de $\Delta entrada$. No exemplo dado, $\Delta SE = 0,56 - 0,40 = 0,16$ períodos de *clock*.

Após transmitir o pacote contendo informação de PCR, o adaptador de taxa reinicia o contador de pacotes enviados e guarda a diferença ΔSE entre o atraso inserido na saída e o da entrada, em um registro na memória, que no exemplo anterior seria de $0,16$. Assim, quando o próximo pacote com PCR for escalonado para ser enviado, o módulo de adaptação de taxa procurará o melhor momento para isso, com o objetivo de igualar o atraso de saída ao de entrada, subtraído pelo valor de ΔSE. Isto pode ser expresso como

$$Esaida = (\Delta entrada - \Delta SE) \bmod 1, \tag{5.17}$$

onde $Esaida$ é o valor de atraso desejado, na saída, e $\Delta entrada$ é o atraso identificado pela contagem dos pacotes de entrada, em módulo de 1, de modo que $0 \leq Esaida < 1$.

É interessante notar que ΔSE representa a diferença de tempo entre os instantes de chegada e os de envio dos pacotes, com relação ao *clock* de 27 MHz. Além disso, este valor é iniciado com zero, dado que o sistema começa a enviar pacotes assim que o primeiro é detectado no $buffer$ de entrada.

O adaptador de taxa opera segundo o diagrama de fluxo mostrado na Figura 5.2, enviando dados sempre que o $buffer$ de entrada indicar dois ou mais pacotes armazenados; caso contrário, ele envia pacotes nulos. Quando um pacote com PCR é agendado para ser enviado, o módulo de adaptação de taxa calcula quantos pacotes nulos devem ser enviados antes deste, de modo que o atraso inserido na saída seja o mais próximo possível de $Esaida$. Assim, gera-se um vetor chamado $Atrasos$, que indica o $jitter$ de saída para n pacotes, de modo que seja possível escolher o n mais adequado, onde n varia de 0 (pacote enviado imediatamente) até um número máximo dado por

$$max\_hold = \frac{(B_{size} - Pack\_n)}{PCR\_n} \cdot \frac{Taxa\_de\_saida}{Taxa\_de\_entrada}, \tag{5.18}$$

onde B_{size}, $Pack\_n$, e PCR_n são o tamanho do $buffer$ do módulo de adaptação de taxa, o número de pacotes já armazenados e o número de pacotes contendo PCR, respectivamente. Logo, max_hold é o número de pacotes que podem ser enviados sem que o $buffer$ fique sem espaço. Além disso, max_hold não pode ser maior que um número predefinido, de modo a garantir que tais cálculos sejam executados em tempo

hábil.

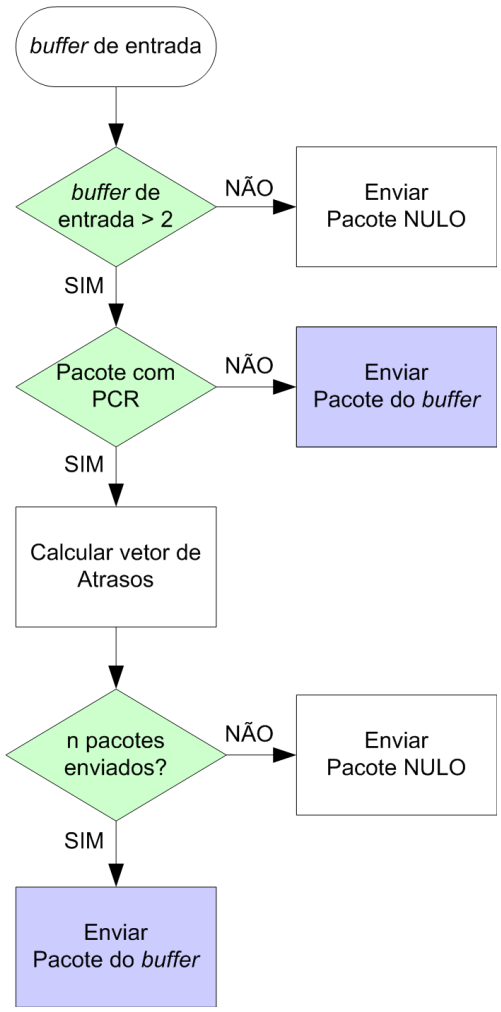


Figura 5.2: Diagrama do módulo de adaptação de taxa integrado à correção de PCR.

A Tabela 5.1 mostra um exemplo de processo de adaptação de taxa de 20 Mbps para 25 Mbps, mostrando contadores, atraso de entrada, valor desejado de saída, atraso na saída e a diferença ΔSE, onde cada iteração (iter.) representa um pacote com PCR recebido pelo processador de TS.

5.2.4 Compartilhamento do contador: a estrutura de semáforo

A estrutura de semáforos também pode ser incorporada a esta nova metodologia, com o objetivo de compartilhar o acesso ao contador de 27 MHz. Como visto na seção 4.3, uma vantagem imediata dessa abordagem é a diminuição da quantidade de recursos necessários. A estrutura resultante está ilustrada na Figura 5.3 e utiliza os mesmos pseudo-algoritmos para a entrada e a saída (Algoritmos 1 e 2), mostrados na página 28.

Tabela 5.1: Exemplo de adaptação de taxa.

iter.	$NP_{entrada}$	$\Delta entrada$	ΔSE	$Esaida$	NP_{saida}	$\Delta saida$	$Novo\ \Delta SE$
0	0	0	0	0	0	0	0
1	312	0.8	0	0.8	403	0.96	0.16
2	465	0	0.16	0.84	584	0.88	0.04
3	466	0.4	0.04	0.36	580	0.6	0.24
4	466	0.4	0.24	0.16	585	0.2	0.04
5	468	0.2	0.04	0.16	582	0.24	0.08

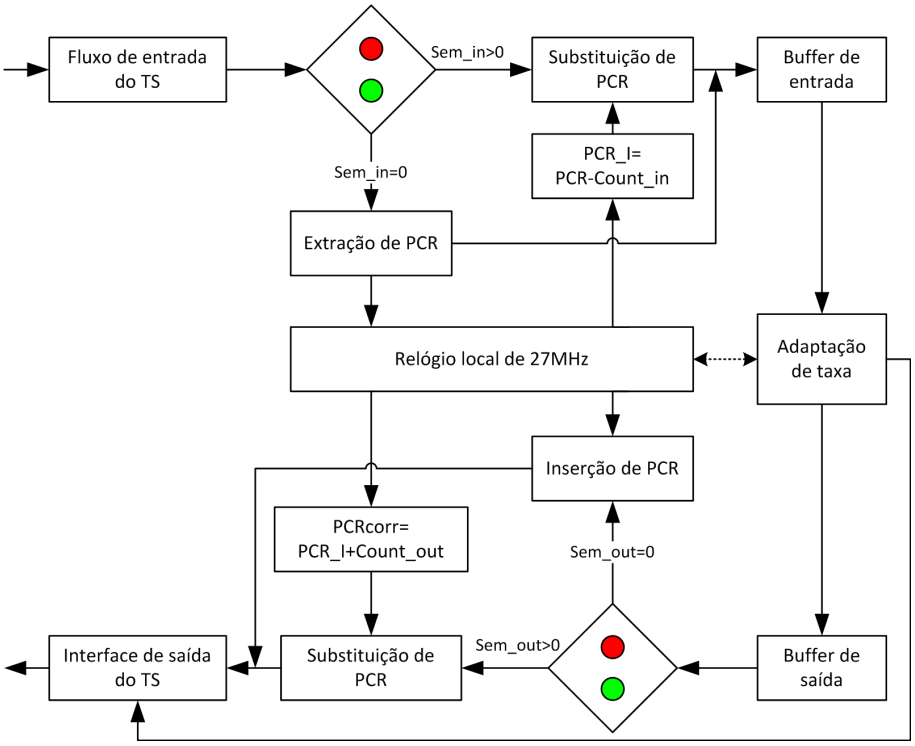


Figura 5.3: O diagrama de blocos do método de Correção de PCR integrada à Adaptação de Taxa, com a utilização de contadores.

5.3 A correção integrada utilizando acumuladores

Assim como no método proposto por Machmerth e Stoerte [12], descrito na Seção 3.2.4, também é possível se realizar a correção de PCR integrada à adaptação de taxa com acumuladores. Isto pode ser feito através da substituição do contador local de $27MHz$ por um acumulador de saída, gatilhado pela taxa de bytes da interface de saída, que é responsável pela correção dos valores de PCR, e do emprego de um acumulador de entrada, gatilhado pela taxa de bytes da interface de entrada, cujo objetivo é compensar a diferença de fase entre o momento da chegada de pacotes com PCR e o gatilho do acumulador de saída. A estrutura de semáforos também pode ser

aplicada aos acumuladores, de modo a processar um MPTS com um único módulo de correção.

Obviamente, esta abordagem aumenta a complexidade computacional, dado que utiliza aritmética em ponto flutuante e exige maior carga computacional durante o processamento. No entanto, mostra-se que o método fica mais robusto, devido ao fato dos gatilhos de correção de PCR estarem diretamente relacionados à taxa de bytes da interface de saída, o que evita erros inseridos pelo descasamento entre o relógio local e o instante de saída dos pacotes.

5.3.1 O acumulador de saída para a correção de PCR

Embora as abordagens mais tradicionais utilizem apenas o contador local como relógio, é possível o emprego de outra fonte de pulsos, já disponível no sistema, para o cálculo dos gatilhos de $27MHz$. O presente método utiliza um acumulador gatilhado pela taxa de bytes da interface de saída [12], carregado com os atuais valores de PCR, de modo a corrigir as referências de tempo. Este acumulador, chamado de acumulador de saída, tem a sua taxa de acumulação dada por

$$taxa\_acum\_saida = \frac{STC\_freq}{taxa\_saida}, \tag{5.19}$$

onde $taxa\_saida$ é a taxa de saída, em bytes por segundo, e STC_freq é a frequência do relógio do sistema, ou seja, $27MHz$. Sendo assim, este acumulador contém o valor de PCR corrigido de um dado pacote, com base no período de tempo em que ele passou no módulo de adaptação de taxa.

5.3.2 O acumulador de entrada para o compensação de erros

O valor de PCR, lido pela interface de entrada, é carregado no acumulador de saída, no instante em que é detectado, e é incrementado pelo mesmo valor ($taxa\_acum\_saida$), a cada gatilho, não importando o instante em que foi carregado. Sendo assim, há um erro inerente, causado pela diferença de fase entre o byte de entrada e o gatilho do acumulador de saída (taxa de bytes na saída). Logo, de modo a se estimar este erro, utiliza-se um acumulador, chamado de acumulador de entrada, que é gatilhado pela taxa de bytes da interface de entrada, com uma taxa de acumulação dada por

$$taxa\_acum\_entrada = \frac{taxa\_saida}{taxa\_entrada} \bmod 1, \tag{5.20}$$

onde $taxa\_acum\_entrada$ contém a diferença de fase a cada byte da interface de entrada, em quantidade de períodos de bytes de saída, que estão diretamente relacionados

aos gatilhos de correção de PCR, e *taxa_entrada* é a taxa de entrada, em bytes por segundo.

O acumulador de entrada é incrementado considerando-se apenas sua parte não inteira, com

$$acum\_entrada = (acum\_entrada + taxa\_acum\_entrada) \bmod 1, \tag{5.21}$$

onde *acum_entrada* contém a diferença de fase acumulada, relacionada ao tempo de chegada do pacote com PCR no adaptador de taxa.

No momento em que um valor de PCR é carregado no acumulador de saída, o valor de *acum_entrada* é utilizado para se estimar o erro de contagem de correção de PCR δF, dado por

$$\delta F = acum\_entrada \cdot taxa\_acum\_saida. \tag{5.22}$$

O termo δF é análogo ao atraso de entrada calculado para a correção integrada com contadores, ou seja, o valor $\Delta entrada$ introduzido na seção 5.2.1, e representa a parte escura da Figura 5.1, que deve ser deduzida da contagem de correção de PCR. Isto ocorre devido à ausência de sincronismo entre as interfaces de entrada e saída.

5.3.3 O procedimento de correção de PCR

No presente método, a correção do valor de PCR é dada pela quantidade de tempo que um determinado pacote permanece no módulo de adaptação de taxa, incrementando seu valor a cada gatilho do acumulador de saída. Além disso, o valor δF indica a parte que deve ser subtraída do total acumulado, durante o processo de correção. Desta forma, quando um pacote com PCR é detectado, o acumulador de saída é carregado com o respectivo valor de PCR, subtraído do termo δF, que é obtido através do valor *acum_entrada* atual do acumulador de entrada. Este termo é de caráter essencial à abordagem de correção, devido à maior granularidade da taxa de correção do acumulador de saída, quando comparado ao relógio de $27MHz$. A forma completa da estratégia de correção é dada por

$$PCR_{corr} = PCR + \Delta PCR_{comp} - \delta F \tag{5.23}$$

onde PCR_{corr} é o valor de PCR corrigido, PCR é o valor de PCR original, ΔPCR_{comp} é a quantidade adicionada ao valor de PCR, pelo acumulador de saída, e δF o erro de diferença de fase a ser descartado.

5.3.4 A estrutura de semáforos para o compartilhamento do acumulador de saída

Esta arquitetura é baseada na forma original apresentada no Capítulo 4, com alterações no módulo de correção, mas mantendo as mesmas funcionalidades dos semáforos de entrada e de saída. O seu diagrama em blocos está ilustrado na Figura 5.4.

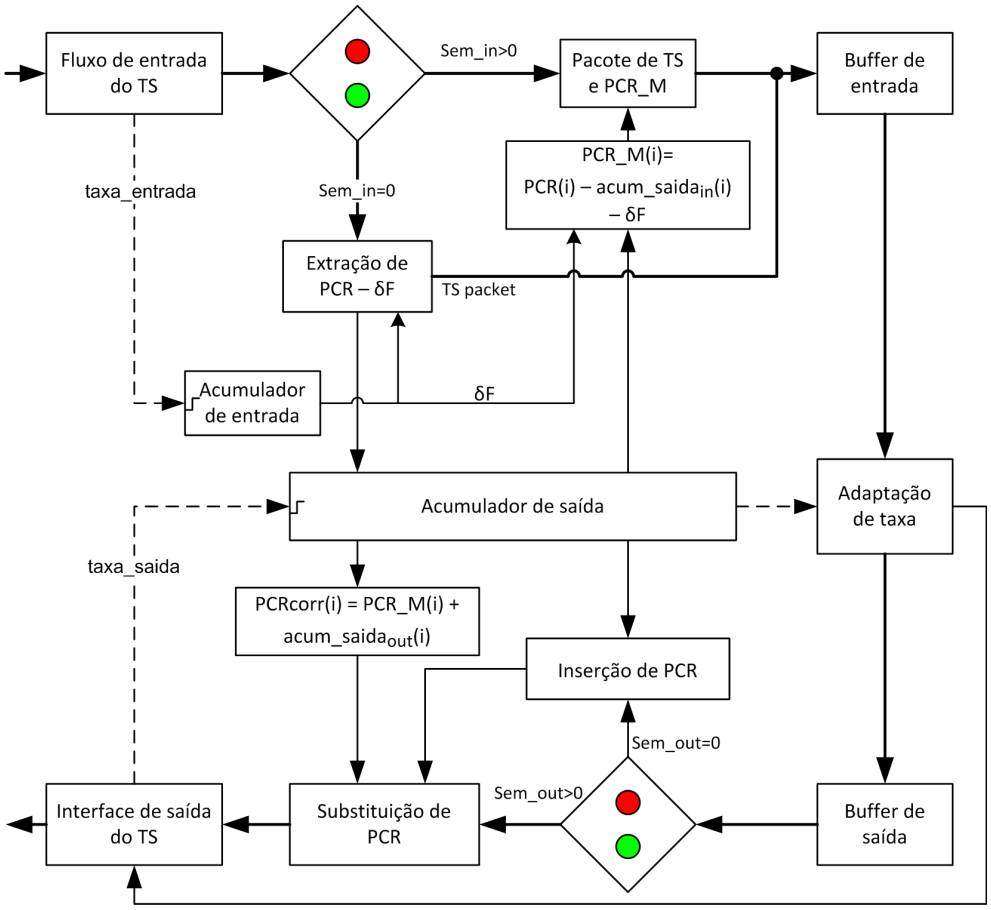


Figura 5.4: O diagrama de blocos do método de Correção de PCR integrada à Adaptação de Taxa, com a utilização de acumuladores.

Assim como descrito anteriormente, utiliza-se uma estrutura de semáforos para o compartilhamento do módulo de correção de PCR, entre todas as bases de tempo do TS. Um semáforo de entrada indica se a correção será feita diretamente pelo carregamento do valor de PCR no acumulador de saída ou, se o acumulado já estiver ocupado, por correções intermediárias através da compensação do valor de entrada pela saída do acumulador de saída. Ainda, um semáforo de saída determina a forma complementar para a finalização da correção de PCR, indicando se o valor deve ser lido diretamente do registro do acumulador de saída ou de deve ser novamente compensado pelo acumulador de saída. Estes semáforos (entrada e saída) são de valores inteiros e inicializados com 0, com incrementos a cada contagem de pacote com PCR. Eles retornam ao seu estado

inicial, ou seja, 0, sempre que os seus valores forem iguais, o que indica que todos os pacotes com PCR, detectados na interface de entrada, já foram processados e enviados pela interface de saída.

Quando o primeiro pacote contendo uma referência de tempo chega à interface de entrada, seu valor de PCR é extraído e carregado no registro do acumulador, com o valor

$$Reg\_Acumulador = PCR(i) - \delta F. \tag{5.24}$$

Isto ocorre porque o valor inicial do semáforo de entrada é 0, que em seguida é incrementado. Logo, os próximos pacotes encontrarão este semáforo com valor maior que 0, indicando que o mesmo se encontra no estado ocupado. Nestes casos, o sistema gera uma referência temporária, PCR_M, baseada no seu valor de PCR e no acumulador de saída, que é dada por

$$PCR\_M(i) = PCR(i) - acum\_saida_{in}(i) - \delta F, \tag{5.25}$$

onde $acum\_saida_{in}(i)$ é o valor do acumulador de saída, no momento em que o pacote é detectado na interface de entrada. Desta forma, o Algoritmo 3 é relacionado ao semáforo de entrada do módulo de correção. Neste, quando o primeiro pacote com PCR for enviado pela interface de saída, com o semáforo de saída ainda em 0, o valor de PCR será substituído diretamente com o valor do acumulador de saída. No entanto, quando o semáforo for maior que 0, o que ocorre para os pacotes subsequentes, o procedimento de correção será feito através de

$$PCR_{corr}(i) = PCR\_M + acum\_saida_{out}(i), \tag{5.26}$$

onde $acum\_saida_{out}(i)$ é o valor do registro do acumulador, quando o pacote é agendado para envio. Substituindo-se a Equação (5.25) na Equação (5.26), é possível comparar o termo ΔPCR_{comp} com a Equação (5.23):

$$\Delta PCR_{comp}(i) = acum\_saida_{out}(i) - acum\_saida_{out}(i), \tag{5.27}$$

que consiste na diferença entre os valores do acumulador de saída, quando o pacote é recebido e quando este é enviado pelo sistema.

Finalmente, quando os semáforos de entrada e de saída possuírem o mesmo valor, ambos são ajustados com o valor 0. Isto indica que todos os pacotes com PCR carregados no *buffer* já foram enviados e o acumulador de saída está novamente livre para ser diretamente carregado. O Algoritmo 4 é relacionado ao semáforo de saída do módulo de correção e é executado sempre que os pacotes são agendados para serem enviados

pelo adaptador de taxa.

```
1  início
2  |  δF ← acum_saida × taxa_acum_saida;
3  |  Verificar Sem_in;
4  |  se Sem_ in > 0 então
5  |  |  PCR_M(i) ← PCR(i) - acum_saida_in(i) - δF;
6  |  |  PCR_M(i) acompanha o pacote de TS
7  |  senão
8  |  |  Registro do Acumulador ← PCR(i) - δF;
9  |  fim
10 |  Incrementar Sem_in;
11 fim
```

Algoritmo 3: Semáforo de entrada para correção integrada com acumuladores.

```
1  início
2  |  Verificar Sem_out;
3  |  se Sem_ in > 0 então
4  |  |  PCR_corr(i) ← PCR_M(i) + acum_saida_out(i) - Δ_PCR;
5  |  senão
6  |  |  PCR_corr(i) ← Registro do Acumulador - Δ_PCR;
7  |  fim
8  |  Incrementar Sem_out;
9  |  se Sem_ out = Sem_ in então
10 |  |  Sem_out ← 0 ;
11 |  |  Sem_in ← 0;
12 |  fim
13 fim
```

Algoritmo 4: Semáforo de saída para correção integrada com acumuladores.

Vale ressaltar que, ao se substituir o valor final de $PCR_{corr}(i)$, deve haver uma conversão para um inteiro de 42 bits, separando-se a base e a extensão de PCR [4].

5.3.5 A estratégia de correção integrada com acumuladores

No início deste capítulo, mostrou-se que enviar pacotes que tenham passado um número não inteiro de períodos de 27 MHz no adaptador de taxa, mesmo com um mecanismo de correção de PCR bem executado, pode constituir uma fonte adicional de *jitter*. Desta forma, é de interesse que o módulo de adaptação de taxa tenha conhecimento do *jitter* inserido pelo módulo de correção, de modo que eles operem de forma integrada, aproximando o tempo que os mesmos passam no adaptador de taxa a um número inteiro de períodos de 27 MHz.

Quando um pacote com PCR é agendado para envio, o adaptador de taxa calcula o valor PCR_{corr} que seria escrito neste mesmo pacote, se este fosse enviado instantaneamente ou em alguma próxima oportunidade futura, de modo a manter PCR_{corr} o mais próximo possível de um número inteiro. Desta forma, gera-se o vetor *Atrasos*, que consiste nos possíveis valores de PCR para n pacotes futuros, com n variando de 0 a max_hold, como dado na Equação (5.18) da seção 5.2.3.

Após a criação do vetor *Atrasos*, o adaptador de taxa procura a posição de n do vetor que possui a menor parte fracionária, enviando n pacotes nulos antes do pacote com PCR.

5.4 Análise do método proposto de correção integrada

A principal característica do método proposto é a utilização de uma estrutura de processamento inteligente, que envia pacotes na janela de tempo mais adequada. Em resumo, o módulo de correção de PCR apresenta realimentação direta com o adaptador de taxa e influencia o seu comportamento, o que é diferente dos métodos tradicionais, em que cada módulo efetua todas as tarefas de forma independente.

Como se pode notar, a complexidade associada é maior quando comparada à de métodos tradicionais. Contudo, esta nova abordagem é capaz de fornecer um *jitter* reduzido, com o potencial de aumento de velocidade de sincronização do sistema, e produzir TSs com bases de tempo mais robustas, que podem ser posteriormente reprocessados.

Assim como feito para a correção por um contador controlado por semáforos, também é possível uma estimativa para o número de operações aritméticas envolvidas no método. A formulação matemática para se chegar a este número envolve as taxas de entrada e de saída de bits, as taxas de PCR, a proximidade entre diferentes bases de tempo e o valor esperado do intervalo de tempo de correção (CTI_{exp}). Apenas as operações realizadas pelo módulo de correção e módulos de adaptação serão abordadas.

A análise é similar à apresentada na seção 4.5, onde a taxa de chegada de pacotes de PCR, para cada base de tempo, é vista como um evento probabilístico independente. Devido à estrutura do semáforo, se o semáforo de entrada e, consequentemente, o de saída forem 0, como pode ser visto na Tabela 5.2, a correção integrada com contadores realizará quatro multiplicações em ponto flutuante (Equações (5.11), (5.14) e (5.18)), uma adição em ponto flutuante (Equação (5.16)) e duas adições de inteiros (Equação (5.18) e a adição do termo de compensação Δ_{PCR}). Entretanto, nas mesmas condições, a correção integrada com acumuladores realizará duas multiplicações em ponto flutu-

ante (Equação (5.18)), uma adição em ponto flutuante (Equação (5.22)) e as mesmas duas adições de inteiros. Caso contrário, se o semáforo for maior que 0, serão realizadas duas adições extras, em inteiros, para o caso dos contadores, e em ponto flutuante para os acumuladores. Além disso, se um novo pacote com PCR chegar no instante final de um intervalo de correção, as próximas correções também serão feitas com três adições. Sendo assim, o intervalo de tempo durante o qual pode ocorrer um valor de semáforo zero será considerado $2CTI_{exp}$, que foi deduzido na Equação (4.5).

Tabela 5.2: Número de operações fixas no método de correção integrada.

Tipo	Correção com Contadores	Correção com Acumuladores
Adição em ponto flutuante	1	1
Adição em inteiro	2	2
Multiplicação em ponto flutuante	4	2

Supondo que existam N bases de tempo diferentes, o número esperado de operações de adição extras também é dado pela Equação (4.7), repetida aqui por conveniência:

$$NADD_{extra} = \sum_{n=2}^{N} \frac{(n-1)}{CTI_{exp}} \sum_{A \in F_n} \prod_{l \in A} p_l \prod_{k \in A^c} (1 - p_k). \tag{5.28}$$

$NADD$ é o número de operações de adição extras, em inteiro, para o caso dos contadores, ou em ponto flutuante, para os acumuladores, e o restante dos componentes tem o mesmo significado apresentado na seção 4.5.

O conhecimento do número de valores de PCR em um segundo, que demandam operações fixas quando o semáforo for 0, definido como NP, também é importante para as estimativas de complexidade, sendo dado por

$$NP = \sum_{i=0}^{N-1} \left(\frac{1}{PCR\_RATE(i)} \right), \tag{5.29}$$

onde $PCR\_RATE(i)$ é a taxa de chegada de valores de PCR, de uma determinada base de tempo i, em segundos, o que é similar à Equação (4.7)

O número de adições, em ponto flutuante, ainda envolve o fato de que cada correção de PCR pode levar no máximo max_hold adições para predizer o $jitter$ de n ($n = 0, 1, 2...max\_hold$) pacotes a frente.

Assim, para a correção de PCR integrada à adaptação de taxa, com contadores, o número de operações aritméticas, por segundo, é dado por

$$NADD\_FLOAT_{cont} = (1 + max\_hold) \cdot NP \tag{5.30}$$

$$NADD\_INT_{cont} = 2 \cdot NP + NADD_{extra} \tag{5.31}$$

$$NMUL\_FLOAT_{cont} = 4 \cdot NP, \tag{5.32}$$

onde $NADD\_FLOAT_{cont}$ e $NADD\_INT_{cont}$ são o número de adições em ponto flutuante e o número de adições em inteiro, por segundo, respectivamente, e $NMUL\_FLOAT_{cont}$ é o número de multiplicações em ponto flutuante, por segundo, para o caso dos contadores.

Para a correção de PCR integrada à adaptação de taxa, com acumuladores, o número de operações aritméticas, por segundo, é dado por

$$NADD\_FLOAT_{acum} = (1 + max\_hold) \cdot NP + NADD_{extra} \tag{5.33}$$

$$NADD\_INT_{acum} = 2 \cdot NP \tag{5.34}$$

$$NMUL\_FLOAT_{acum} = 2 \cdot NP, \tag{5.35}$$

onde $NADD\_FLOAT_{acum}$ e $NADD\_INT_{acum}$ são o número de adições em ponto flutuante e o número de adições em inteiro, por segundo, respectivamente, e $NMUL\_FLOAT_{acum}$ é o número de multiplicações em ponto flutuante, por segundo, para o caso dos acumuladores.

A análise é a mesma para o método de correção integrada, seja esta com contadores ou acumuladores. No entanto, deve-se notar que o uso de acumuladores implica em um maior número de operações aritméticas em ponto flutuante, o que requer maior carga computacional, ao passo que os contadores utilizam, predominantemente, operações com números inteiros.

Pelas características apontadas na Tabela 3.2, os resultados obtidos são descritos na Tabela 5.3

Tabela 5.3: Características obtidas pelo método de correção integrada.

Característica	*Descrição*
Redução de recursos necessários	Atingido, apenas um contador é necessário.
Complexidade computacional	Aumentou, grande número de operações aritméticas.
Jitter inserido	Atingido, o *jitter* inserido foi bastante reduzido.

Capítulo 6

Resultados

Neste capítulo, os resultados de simulação dos métodos propostos são expostos, com o objetivo de avaliar o seu desempenho e compará-los aos métodos existentes na literatura. As simulações foram realizadas em ambiente Simulink/Matlab [33], com as medições dos resultados executadas por códigos no Matlab. Para cada TS, são analisados o *jitter* de PCR, o intervalo de chegada de PCR, o *jitter* global e o desvio do relógio de $27MHz$. Também são realizadas medições estatísticas a respeito do desvio-padrão [34] para o *jitter* de PCR e o intervalo de PCR, de modo a fornecer outra ferramenta de comparação dos resultados entre as diferentes metodologias. As simulações são divididas em vários cenários. Iniciam com um Fluxo de Transporte com *jitter* de entrada zero, que possibilita analisar o erro de PCR inserido unicamente pelo processador do TS. Em seguida, vários outros cenários são simulados, com números de bases de tempo variados e diferentes taxas de entrada e saída. As taxas de saída influenciam fortemente o processo de correção, de acordo com a parte decimal apresentada pelo TP_{saida}, como demonstrado na Seção 5.1.

6.1 Métodos de medição de PCR

Em um Fluxo de Transporte MPEG-2, o mecanismo desenvolvido para sincronizar o receptor com um determinado programa consiste na leitura de amostras do relógio de 27 MHz do codificador, ou seja, o PCR. Cada valor de PCR está relacionado a uma base de tempo de um programa específico, que pode, também, ser compartilhada entre outros programas. Assim, determinam-se padrões de medição para se avaliarem os efeitos do PCR e assegurar a sincronização do receptor.

As medições adotadas neste trabalho são relacionadas unicamente à formação do TS, com uma avaliação das informações contidas nos valores de PCR e sem se considerarem efeitos de transmissão. Sendo assim, as medições são feitas em modo *offline*,

segundo a recomendação TR 101 290 [35].

6.1.1 Medição do *jitter* de PCR

O *jitter* de PCR, definido como a diferença entre o valor de PCR, obtido de um determinado pacote com referência de tempo, e o valor que o mesmo deveria indicar, de acordo com sua posição no TS, é medido para avaliar a acurácia das referências de tempo. É uma medição realizada apenas em Fluxos de Transporte com taxa de bits constante durante a transmissão.

Como descrito na seção 2.4, o valor de PCR esperado, de acordo com a posição no TS, é dado pela Equação (2.5) e depende unicamente da taxa de bits, da quantidade de bits desde o último valor de PCR, do valor de PCR lido anteriormente e da frequência nominal do relógio de sistema, ou seja, de 27 MHz. Assim, o *jitter* pode ser obtido diretamente pela Equação (2.6).

O diagrama em blocos de todo o sistema de medição para *jitter*, *jitter* global e intervalo de PCR está ilustrado na Figura 6.1. São utilizados os valores de PCR atuais e anterios, assim como os números de bytes. De acordo com a figura, o *jitter* de PCR é obtido através do terminal indicado com o rótulo A.

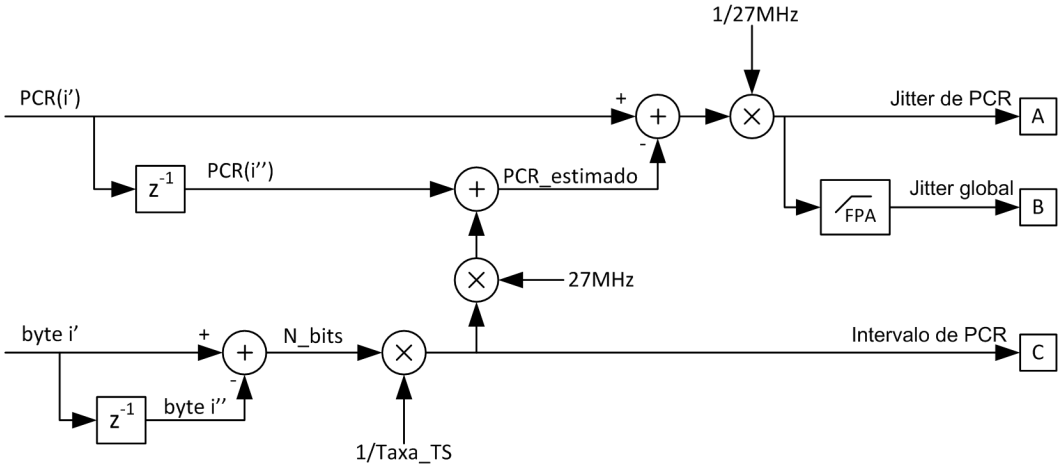


Figura 6.1: Diagrama de blocos da medição de *jitter* de PCR, *jitter* global e intervalo de PCR.

6.1.2 Medição do intervalo de PCR

O intervalo de PCR é a medida do tempo de chegada do valor de PCR atual $PCR(i')$, com base no valor de PCR anterior $PCR(i'')$. Ele é calculado através da diferença de

bits entre valores de PCR adjacentes e a taxa de bits do TS:

$$Int\_PCR(i) = \frac{N\_bits}{Taxa\_TS}, \tag{6.1}$$

onde N_bits é medido em bits e $Taxa\_TS$ em bits por segundo. Na Figura 6.1, este valor é obtido através do terminal com rótulo C.

O limite máximo sugerido pela norma MPEG-2 [4], para o intervalo de PCR, é de $100ms$. No entanto, o guia de implementação para DVB, TR 101 154 [30], recomenda que este valor seja inferior a $40ms$.

6.1.3 Medição do *jitter* global

O *jitter* global é definido como a medição das altas frequências do *jitter* de PCR [35], de modo a analisar os erros que mais causam desvios no sincronismo do receptor. Deste modo, ele pode ser obtido através da utilização de um filtro passa alta do sinal de *jitter* de PCR.

Como as amostras de PCR não chegam sempre a intervalos constantes, antes de se utilizar o filtro é realizada uma interpolação linear [36] do sinal de *jitter* de PCR, de modo a se obter um sinal com taxa de amostragem fixa. O sinal é interpolado nos pontos definidos pela taxa de amostragem, sendo o período desta igual à mediana do intervalo de PCR calculado.

O filtro passa-alta utilizado possui uma janela de Taylor [37], de tamanho 28, com frequência de corte em um terço da banda do sinal. Na Figura 6.1, o *jitter* global pode ser lido no terminal indicado com rótulo B.

6.1.4 Medição do desvio de frequência do relógio de $27MHz$

O desvio de frequência do relógio de 27 MHz é definido como a diferença entre a frequência obtida através da leitura dos valores de PCR, de um determinado programa, e a frequência nominal do relógio de sistema do transmissor [35].

Neste modo de avaliação *offline*, a frequência obtida reflete o comportamento do relógio, no momento da codificação. Assim, é possível prever os desvios que serão gerados no processo de sincronização do receptor.

A frequência é obtida através dos valores de PCR, da contagem de bits e da taxa do TS, de acordo com

$$freq\_PCR(i) = \frac{(PCR(i') - PCR(i'')) \cdot Taxa\_TS}{N\_bits}, \tag{6.2}$$

Em seguida, utiliza-se um filtro passa baixa, de modo a se estimar a oscilação da sincronização do relógio local. O filtro utilizado possui uma janela de Taylor, de tamanho 28, com frequência de corte iniciando em um terço da banda do sinal. Finalmente, o desvio de frequência, em partes por milhão (ppm), é obtido como

$$Desvio\_freq(i) = \frac{freq\_PCR(i) - freq\_nom}{freq\_nom} \cdot 10^6, \tag{6.3}$$

onde $freq\_nom$ é a frequência nominal de 27 MHz. O diagrama de blocos da Figura 6.2 representa o esquema de medição do desvio de frequência do relógio. O desvio de frequência máximo aceito pela norma MPEG-2 é de ±30 ppm [4].

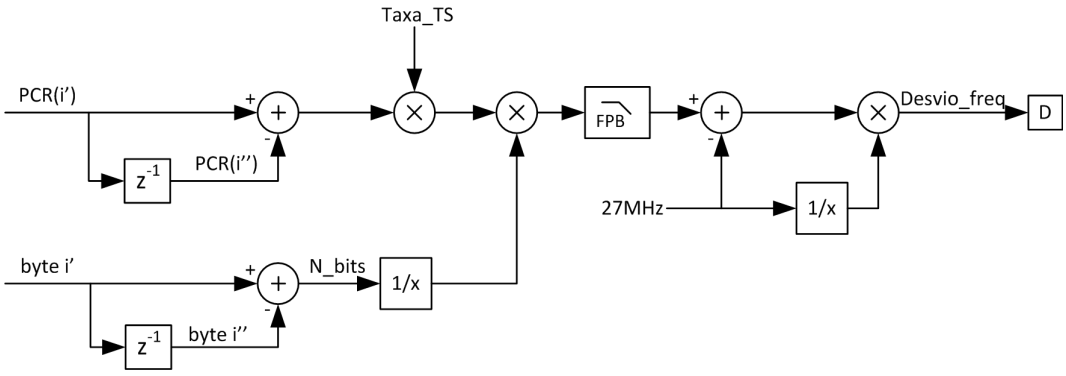


Figura 6.2: Diagrama de blocos do desvio de frequência.

6.2 Simulações

Para se avaliar o desempenho dos métodos propostos, ou seja, estrutura de semáforos e correção de PCR integrada com semáforos e com acumuladores, simulações para cinco diferentes cenários foram realizadas. O ambiente de simulação foi desenvolvido em Simulink/Matlab e corresponda a um conversor de interface SPI-SPI, utiliza fluxos de transporte MPEG-2 reais e, embora não haja uma base de dados definida para avaliação, os cenários escolhidos buscam abranger uma grande parte dos TSs disponíveis comercialmente.

Em cada cenário foram simulados 20 segundos, com os métodos propostos de correção com estrutura de semáforos (Capítulo 4) e correção de PCR integrada à adaptação de taxa (Capítulo 5) utilizando contadores (Seção 5.2) e utilizando acumuladores (Seção 5.3). Para todos os casos, o adaptador de taxa envia pacotes do $buffer$ de entrada sempre que houver 2 ou mais pacotes, caso contrário são enviados pacotes nulos, de acordo com a lógica definida pelo método. O tamanho máximo do $buffer$ do adaptador de taxa foi fixado em 40 pacotes, em nenhum dos cenários este limite foi atingido.

A Tabela 6.1 indica as taxas de entrada dos fluxos de transporte simulados, a taxa de saída fixada pelo adaptador de taxa, assim como o número de bases de tempo contidas no TS.

Tabela 6.1: Taxas de entrada e saída das simulações.

Simulação	*Taxa de entrada*	*Taxa de saída*	*Bases de tempo*
1	$4Mbps$	$5,2Mbps$	1
2	$24,1Mbps$	$34Mbps$	8
3	$10,27Mbps$	$15Mbps$	1
4	$22,1Mbps$	$27Mbps$	1
5	$29,96Mbps$	$35Mbps$	4

6.2.1 Simulação de um TS com *jitter* de entrada zero

No primeiro cenário, 20 segundos de um TS com uma taxa de bits de 4 Mbps e uma única base de tempo, com *jitter* de PCR de entrada zero, são processados por um conversor de taxa, utilizando, separadamente, o método do contador controlado por semáforos e o método de correção de PCR integrada à adaptação de taxa com contadores e também com acumuladores. O tamanho dos pacotes é de 188 bytes e a taxa de saída do conversor foi fixada em $5,2$ Mbps. Sendo assim, TP_{saida} é dado pela Equação 5.1 como

$$TP_{saida} = \frac{8 \cdot 27MHz}{5,2Mbps} \cdot 188bytes = 7809,2308. \tag{6.4}$$

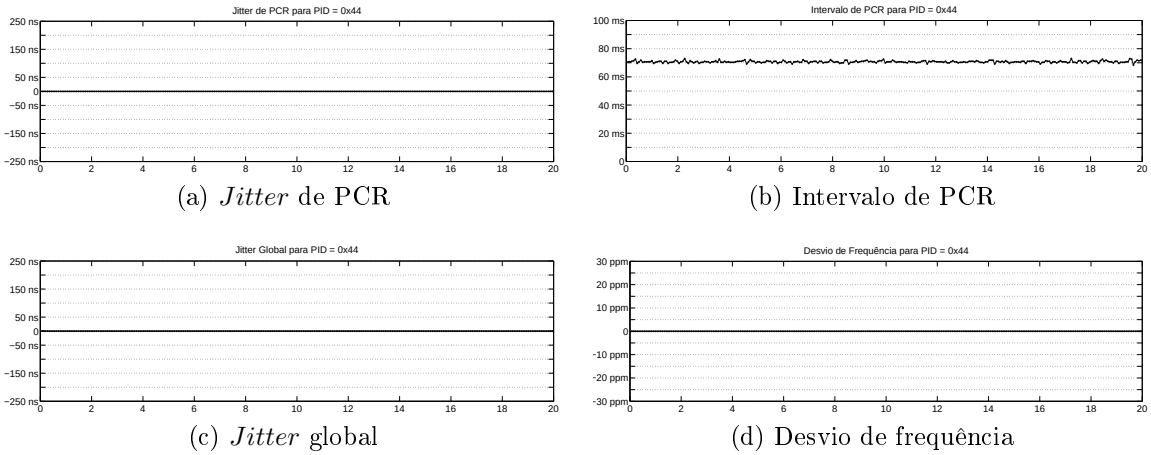
A parte decimal faz com que o erro inserido pelos métodos tradicionais, vistos no Capítulo 3, seja inerente, a fim de mostrar o desempenho das abordagens propostas. As medições referentes ao TS na entrada são mostradas na Figura 6.3.

Observe que o *jitter* de PCR antes do processamento é zero, de modo que os resultados mostrados para cada método, a seguir, indicam o erro inserido unicamente pelo processador de TS na adaptação de taxa.

Os resultados referentes ao método de contadores controlados por semáforos são mostrados na Figura 6.4, de modo a se avaliar os desvios causados por este.

Como apenas um único programa foi processado, a estrutura de semáforos não foi aproveitada. Assim, não há ganho de processamento para o método dos contadores controlados por semáforo e os resultados são basicamente os mesmos apresentados pelas abordagens tradicionais.

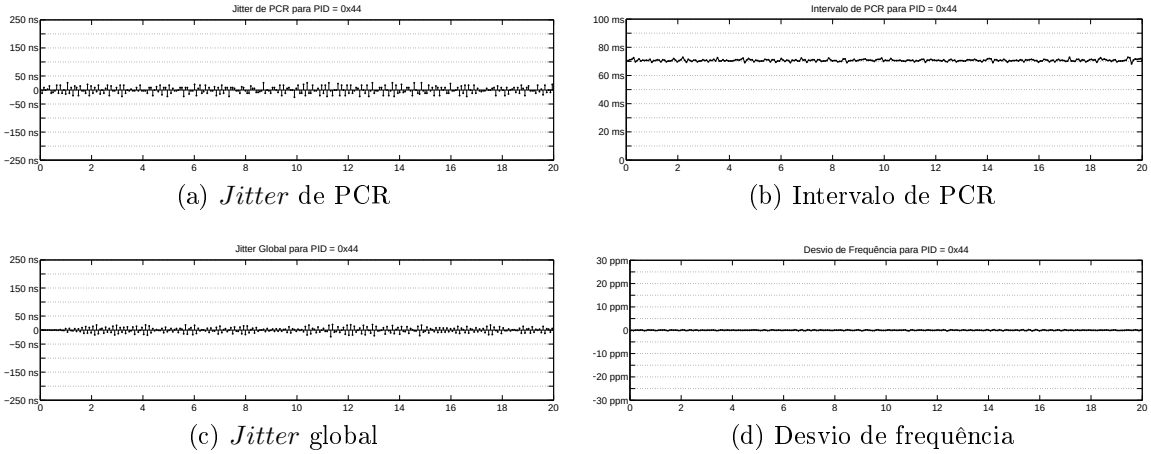
Os resultados referentes ao método de correção de PCR integrada à adaptação de taxa, com contadores, são mostrados na Figura 6.5. Novamente, como o *jitter*



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.3: Medição do TS de *jitter* zero na entrada.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

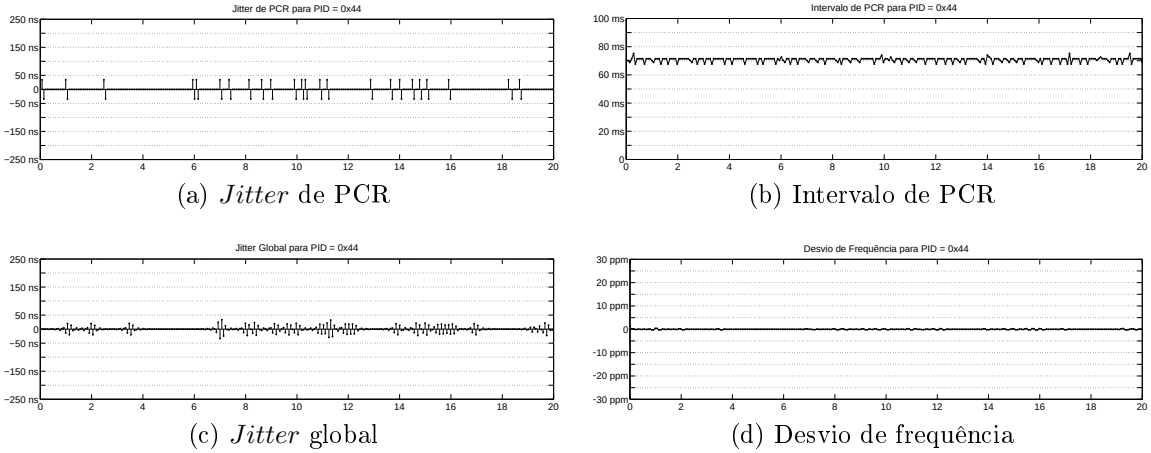
Figura 6.4: Resultados da simulação 1 referentes ao método dos contadores controlados por semáforos.

na entrada é zero, os resultados mostram os erros inseridos unicamente pelo método proposto.

Por último, a Figura 6.6 apresenta os resultados para a correção de PCR integrada à adaptação de taxa, com acumuladores.

De modo geral, o *jitter* inserido por cada método se encontra em níveis aceitáveis pela norma, porém, percebe-se claramente que o método do contador controlado por semáforos muda bastante a característica de *jitter*, o método integrado com contadores apresenta picos de variação menos frequentes mas, algumas vezes, mais pronunciados, e o método integrado com acumuladores realiza um processamento mais transparente, mantendo o TS de saída bastante similar ao de entrada.

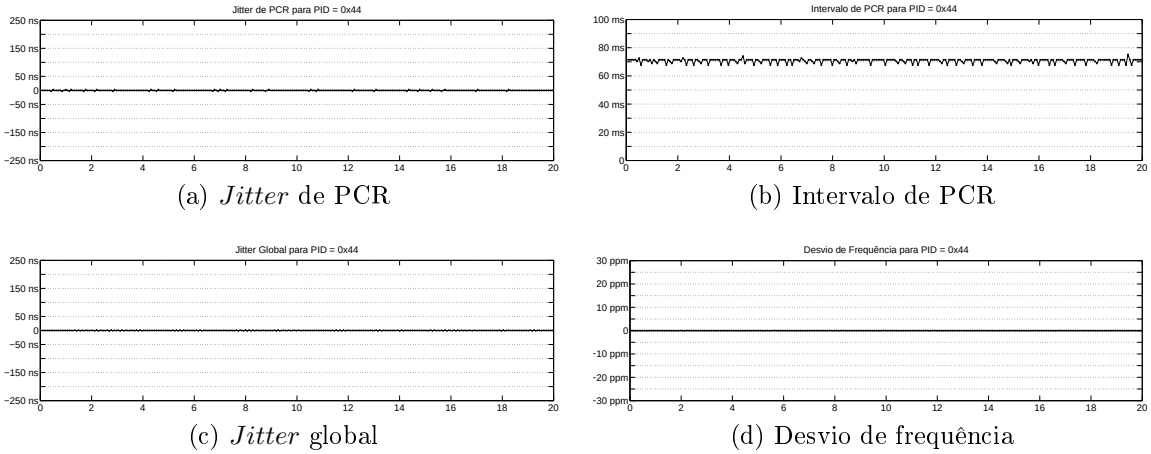
Como o método de correção integrada à adaptação de taxa opera armazenando os pacotes com PCR, para enviá-los na ocasião mais adequada, a tendência do mesmo



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.5: Resultados da simulação 1 referentes ao método de correção integrada, com contadores.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.6: Resultados da simulação 1 referentes ao método de correção integrada, com acumuladores.

é modificar a taxa de chegada da informação de PCR. No entanto, a espera nunca excede o número de pacotes dado pelo valor de max_buffer, fazendo com que esse deslocamento seja bastante pequeno, o que evita grandes alterações no perfil do tempo de chegada de PCR. Sendo assim, esta ação não prejudica o processo de sincronização.

Os resultados apresentados por outros métodos, embora não tenham sido obtidos sob as mesmas condições e com os mesmos TSs, possibilitam uma comparação simplificada. A Tabela 6.2 mostra o *jitter* de saída dos métodos aqui propostos e também dos métodos apresentados na literatura, mostrando que o desempenho destes é superior aos apresentados pelos demais esquemas. Conforme esperado, devido à granularidade do contador de 27 MHz, os resultados da abordagem integrada, com acumuladores, são superiores aos da abordagem com contadores, o que resulta em um mínimo de *jitter*

no TS de saída. Apesar dos resultados apresentados pelos métodos propostos serem superiores aos da literatura, é possível afirmar que, para uma mesma implementação e sob as mesmas condições de operação, o *jitter* inserido pelo contador controlado por semáforos tende a ser semelhante ao apresentado pelos métodos tradicionais. Isto ocorre porque a granularidade de correção é a mesma (27 MHz) e os circuitos de correção também são similares.

Tabela 6.2: Comparação do *jitter* com outros métodos - Simulação 1.

Método	*Jitter de saída*
Semáforos (proposto)	$< 40ns$
Correção integrada com contadores (proposto)	$< 40ns$
Correção integrada com acumuladores (proposto)	$< 8ns$
Xingdong et al. [8]	$108ns$
He et al. [10]	$> 100ns$
Chen et al. [11]	$74ns$

Embora o máximo inserido pelo método com semáforo seja igual ao da correção integrada utilizando contadores, esta possui picos mais dispersos, afetando menos o processo de sincronização do relógio.

A Tabela 6.3 indica o número de operações de soma realizadas por cada um dos métodos. Dado que o TS possui apenas uma (1) base de tempo, a estrutura de semáforos não pode ser amplamente aproveitada, pois esta indica que o módulo de correção (contador ou acumulador) está sempre livre, sem recorrer a correções intermediárias.

Tabela 6.3: Comparação de operações e contadores nos diferentes métodos - Simulação 1.

Método	*Somas*	*Contadores*	*Acumuladores*	*Lógica adicional*
Contadores dedicados [7]	0	1	0	*Não*
Contadores dedicados com janelas deslizantes [3]	0	1	0	*Sim*
Compensação [8, 9]	592	1	0	*Não*
Semáforos (proposto)	0	1	0	*Sim*
Correção integrada com contadores e semáforos (proposto)	592 (int.) 296 (p.f.)	1	0	*Sim*
Correção integrada com acumuladores e semáforos (proposto)	592 (int.) 296 (p.f.)	0	1	*Sim*

A Tabela 6.4 disponibiliza as medidas estatísticas de desvio-padrão para o *jitter* de PCR e o intervalo de PCR, assim como o valor máximo do *jitter* de PCR. Mostra-se como o método de Correção Integrada com acumuladores aproxima o *jitter* de PCR àquele medido na entrada, com pouca alteração no intervalo de PCR.

Tabela 6.4: Desvio-padrão e valor máximo - Simulação 1.

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 0044			
TS de entrada	0	$7,0299.10^{-4}$	$0ns$
Semáforos (proposto)	$1,2749.10^{-8}$	$7,1424.10^{-4}$	$25,6ns$
Correção integrada com contadores (proposto)	$1,4155.10^{-8}$	$1,4906.10^{-3}$	$34,2ns$
Correção integrada com acumuladores (proposto)	$1,1795.10^{-9}$	$1,3853.10^{-3}$	$2,8ns$

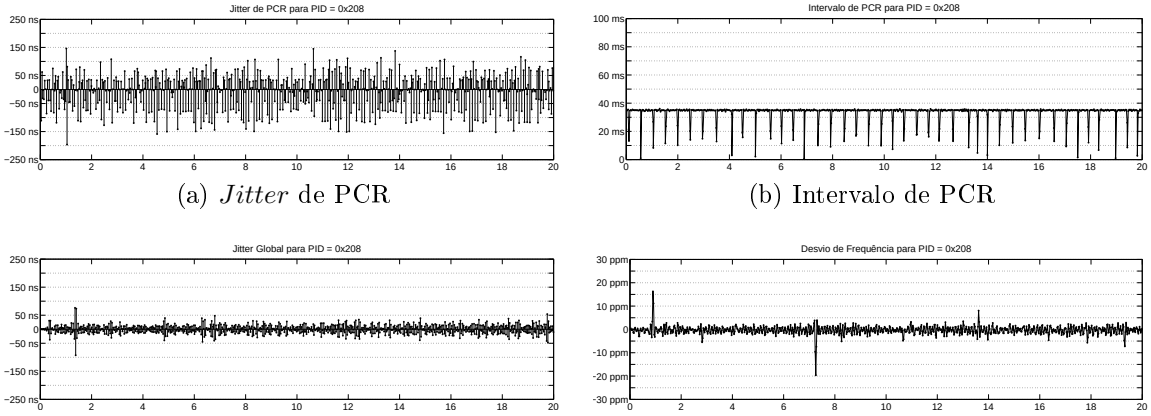
6.2.2 Simulação de um TS com 8 bases de tempo

Neste cenário, 20 segundos de um TS com 8 bases de tempo diferentes são processados por um conversor de taxa, de modo a testar o comportamento com um número maior de programas, utilizando as três abordagens propostas para a correção das informações de PCR. A taxa de bits multiplexada é de, aproximadamente, 24, 1 Mbps, e os resultados das medições, na entrada do processador de PCR, referentes a cada base de tempo, são mostradas nas Figuras 6.7 a 6.14.

O tamanho dos pacotes é de 188 bytes e a taxa de saída do conversor foi fixada em 34 Mbps, de tal modo que TP_{saida} assume o valor

$$TP_{saida} = \frac{8 \cdot 27MHz}{34Mbps} \cdot 188bytes = 1194,353. \tag{6.5}$$

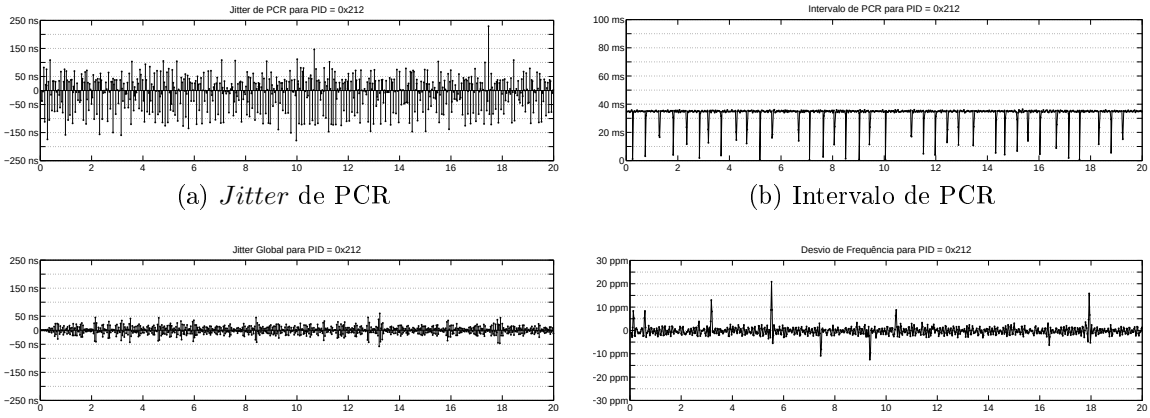
O comportamento das bases de tempo é bastante diferente, com picos de variação no intervalo de PCR para os PIDs 0208, 0212, 021C e 0226, alto *jitter* global para os PIDs 026D e 0277 e picos de desvio de frequência para os PIDS 0208, 0212, 021C e 0226. Este conjunto tem o potencial de testar diversas características das metodologias propostas e revelar os principais pontos que podem influenciar o processo de sincronização. Cada uma destas características das medições tende a ser incrementada quando o TS é processado, *i.e.*, o processo de sincronização é degradado. Assim, o objetivo neste cenário é manter o características das bases de tempo dentro dos níveis aceitáveis pela norma.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

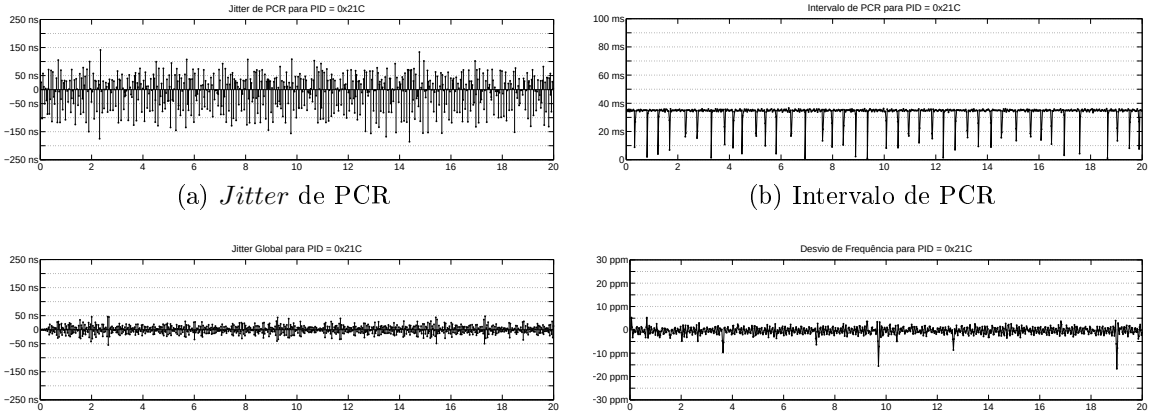
Figura 6.7: Medição de PCR de entrada na simulação 2 - PID 0208.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

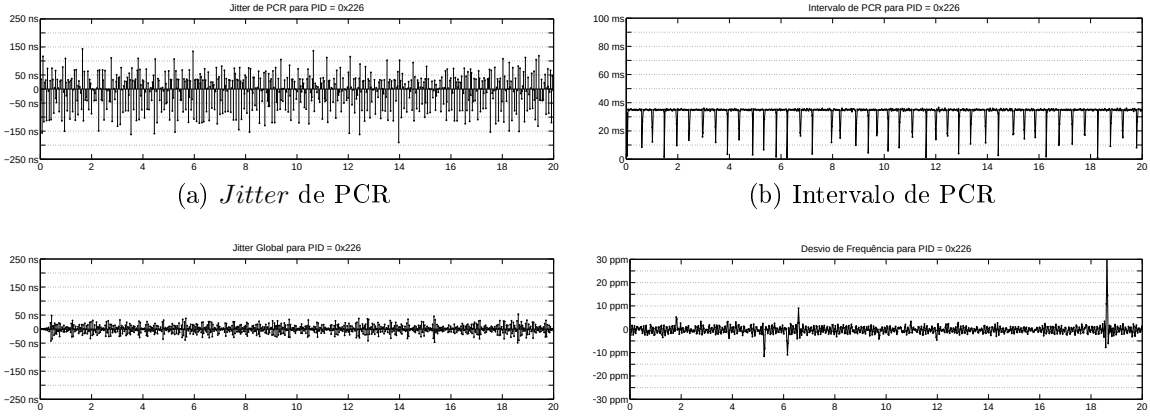
Figura 6.8: Medição de PCR de entrada na simulação 2 - PID 0212.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

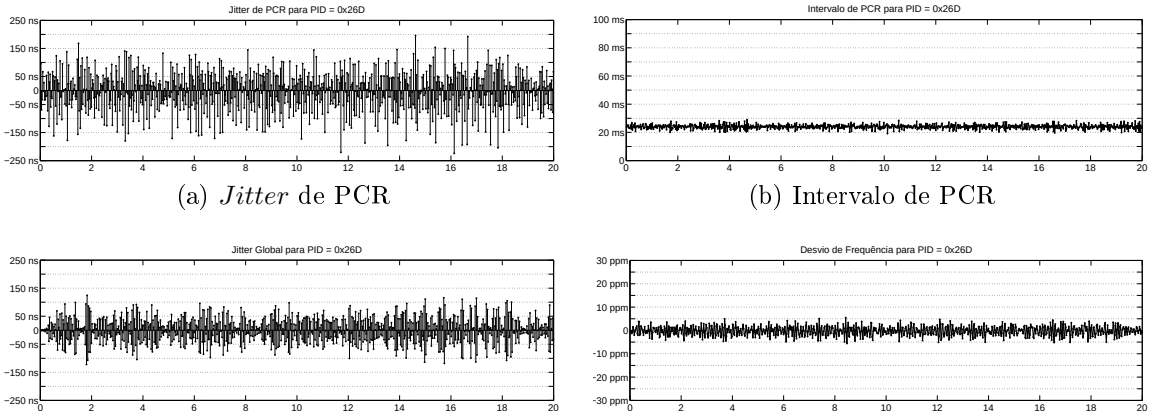
Figura 6.9: Medição de PCR de entrada na simulação 2 - PID 021C.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

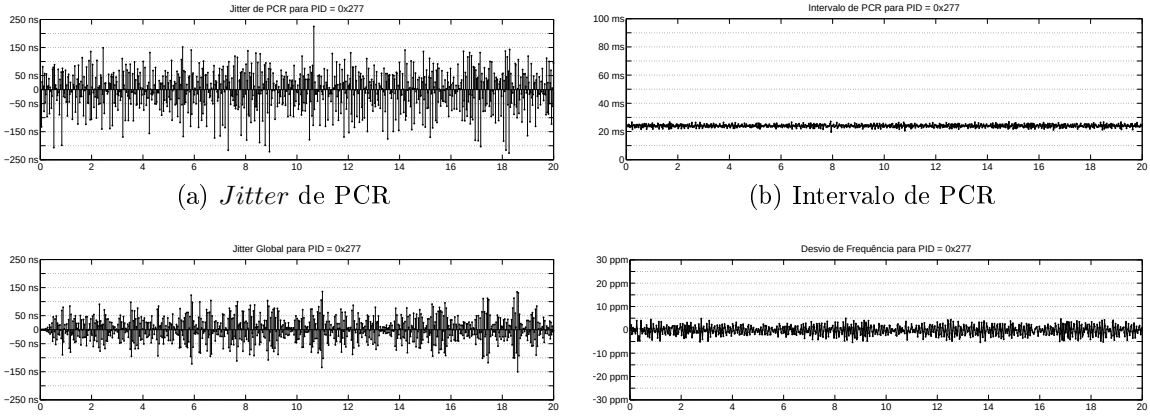
Figura 6.10: Medição de PCR de entrada na simulação 2 - PID 0226.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

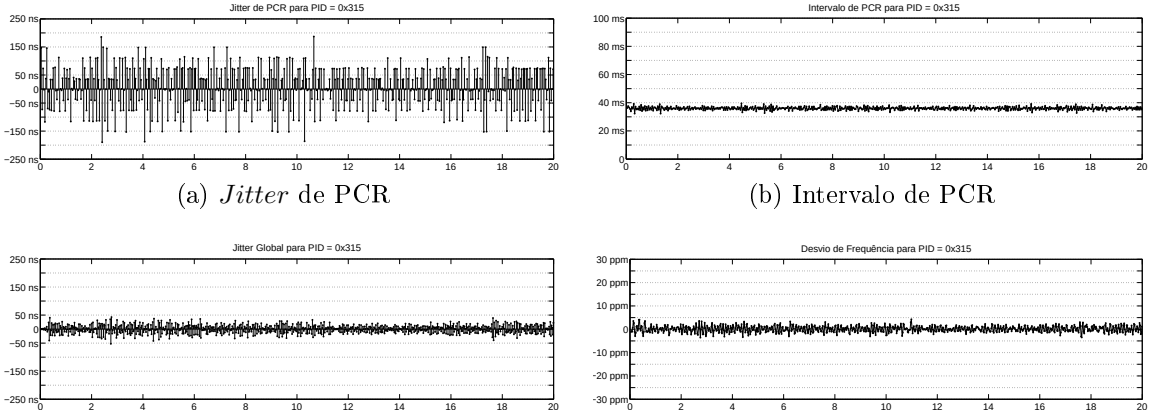
Figura 6.11: Medição de PCR de entrada na simulação 2 - PID 026D.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

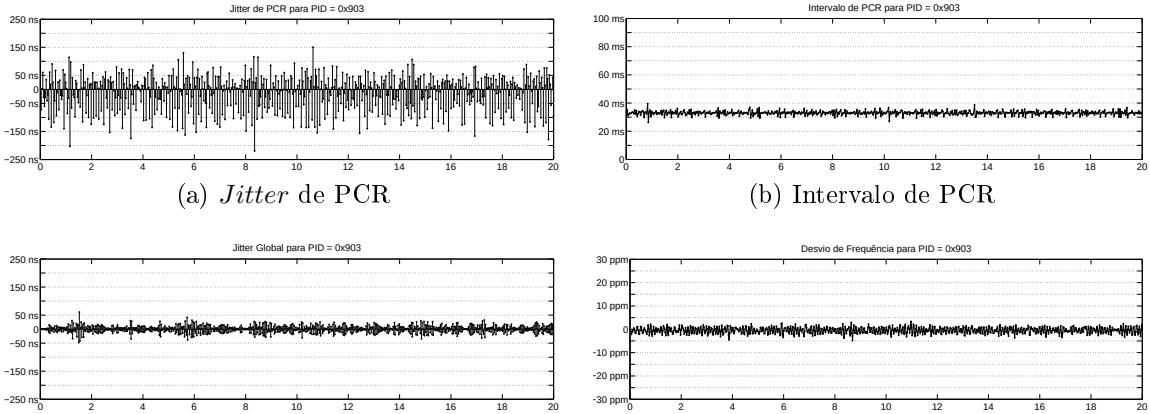
Figura 6.12: Medição de PCR de entrada na simulação 2 - PID 0277.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.13: Medição de PCR de entrada na simulação 2 - PID 0315.



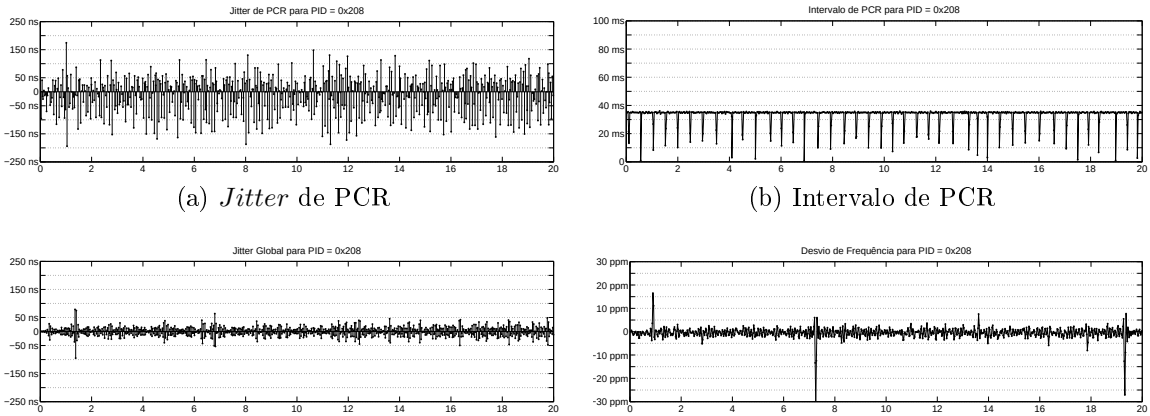
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.14: Medição de PCR de entrada na simulação 2 - PID 0903.

6.2.2.1 Resultados com o contador controlado por semáforos

Os resultados referentes à utilização do método de contador controlado por semáforos são mostrados nas Figuras 6.15 a 6.22. De um modo geral, os resultados indicam que as características analisadas não foram significativamente alteradas, com um maior peso o *jitter* de PCR e desvio de frequência. O primeiro teve um acréscimo visível, cuja causa principal reside na granularidade do processo de correção, o que pode ser visto para os PIDs 0212, $026D$, 0277 e 0315.

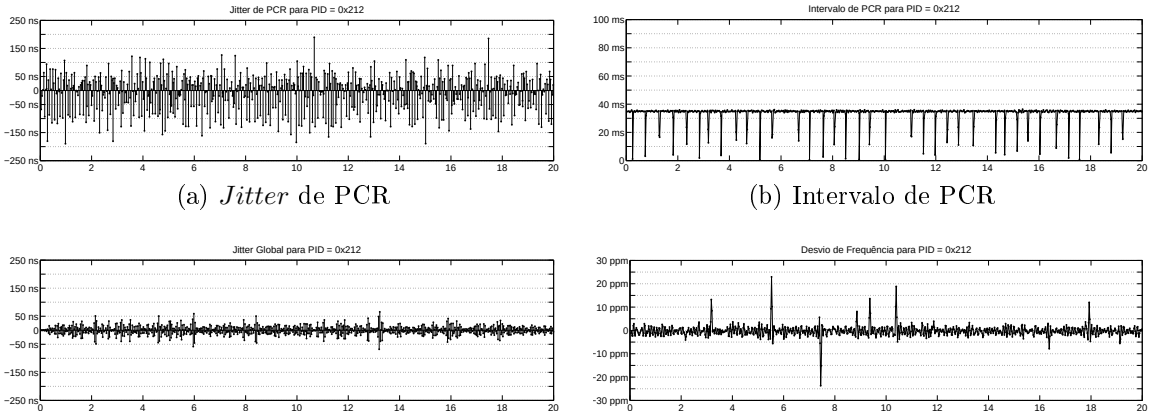


(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.15: Contador controlado por semáforo - simulação 2 - PID 0208.

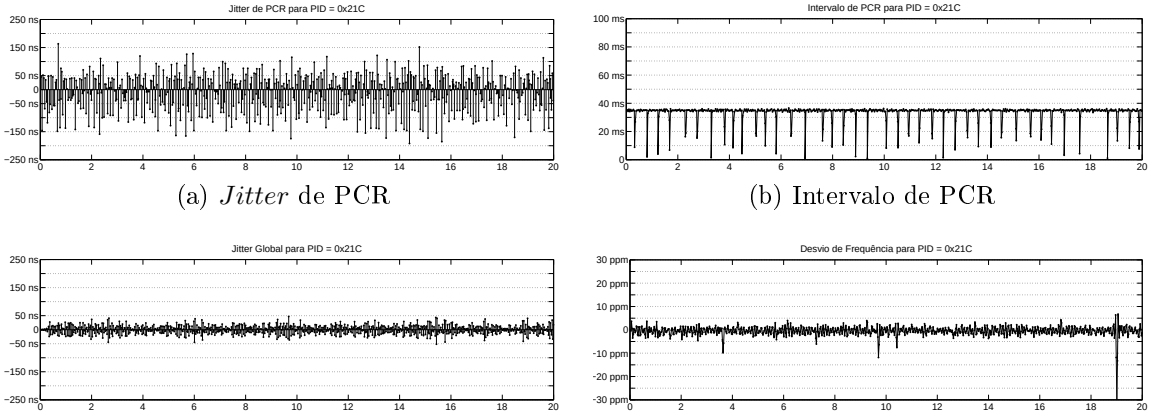
Os desvios de frequência são notavelmente afetados, alcançados valores que chegam ao limite de $-30ppm$ da norma, no caso dos programa com PID 0208, $021C$ e 0226. Embora o *jitter* de PCR tenha sido bastante alterado, as componentes de alta frequência que geram maior impacto na sincronização, não foram consideravelmente alteradas, como pode ser visto nos gráficos de *jitter* global.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

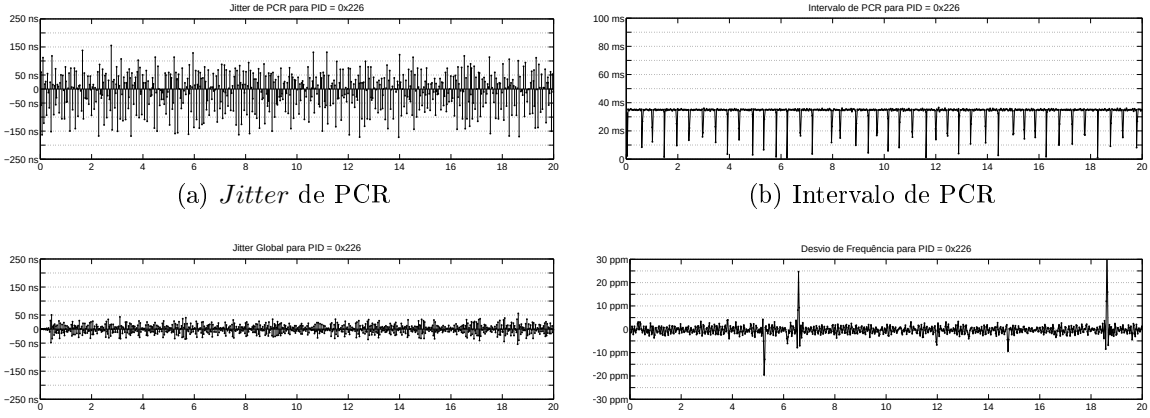
Figura 6.16: Contador controlado por semáforo - simulação 2 - PID 0212.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

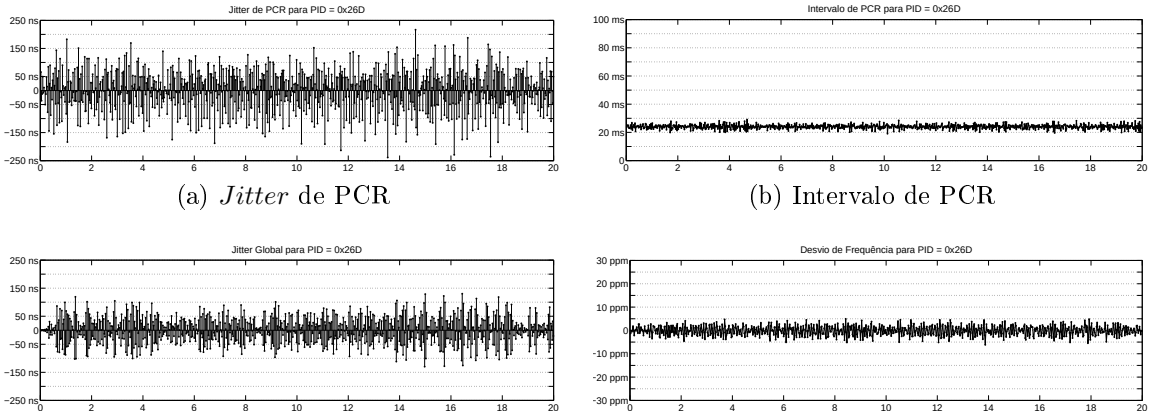
Figura 6.17: Contador controlado por semáforo - simulação 2 - PID 021C.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

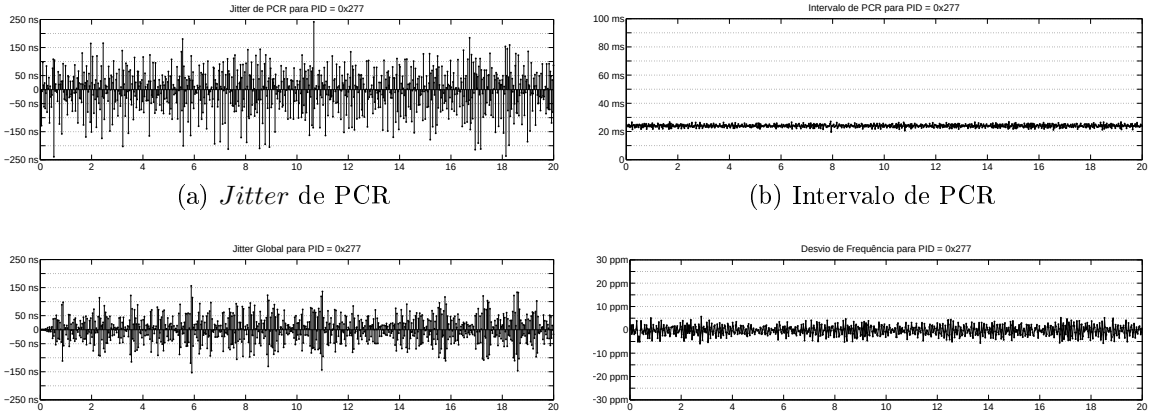
Figura 6.18: Contador controlado por semáforo - simulação 2 - PID 0226.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

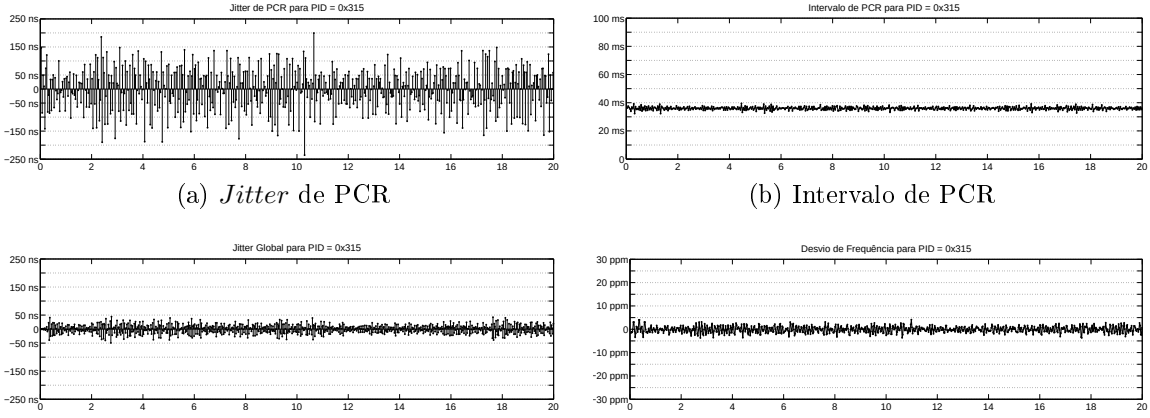
Figura 6.19: Contador controlado por semáforo - simulação 2 - PID 026D.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

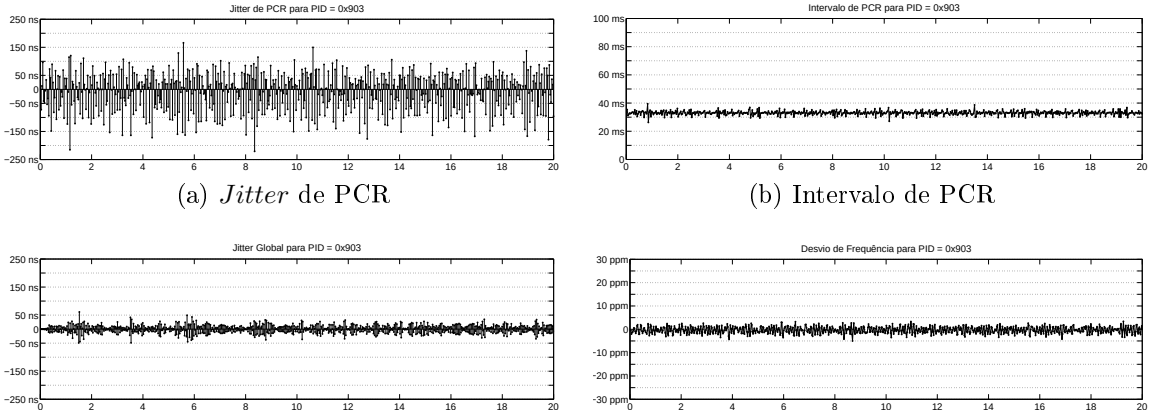
Figura 6.20: Contador controlado por semáforo - simulação 2 - PID 0277.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.21: Contador controlado por semáforo - simulação 2 - PID 0315.



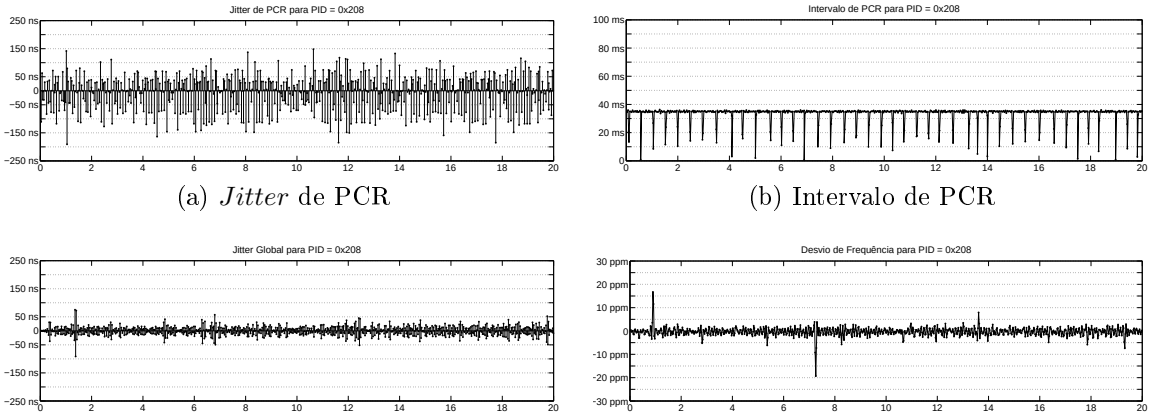
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.22: Contador controlado por semáforo - simulação 2 - PID 0903.

6.2.2.2 Resultados para a correção integrada com contadores

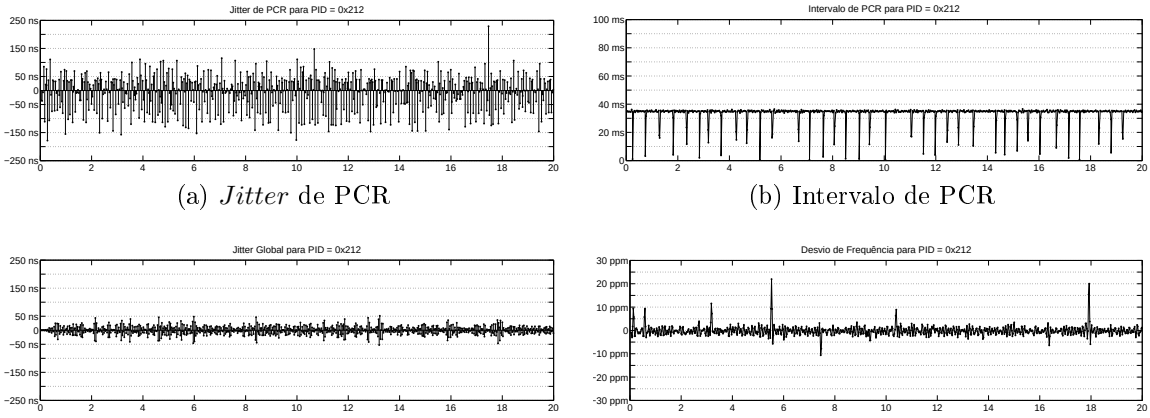
Para o método de correção de PCR integrada à adaptação de taxa com contadores, os resultados são mostrados nas Figuras 6.23 a 6.30. Com relação ao que foi visto na seção anterior, os resultados evidenciam que o impacto no *jitter* de PCR e no desvio de frequência foi bastante reduzido, principalmente neste primeiro. Isto já era esperado, devido ao cálculo de predição de *jitter* realizado para cada pacote com PCR.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

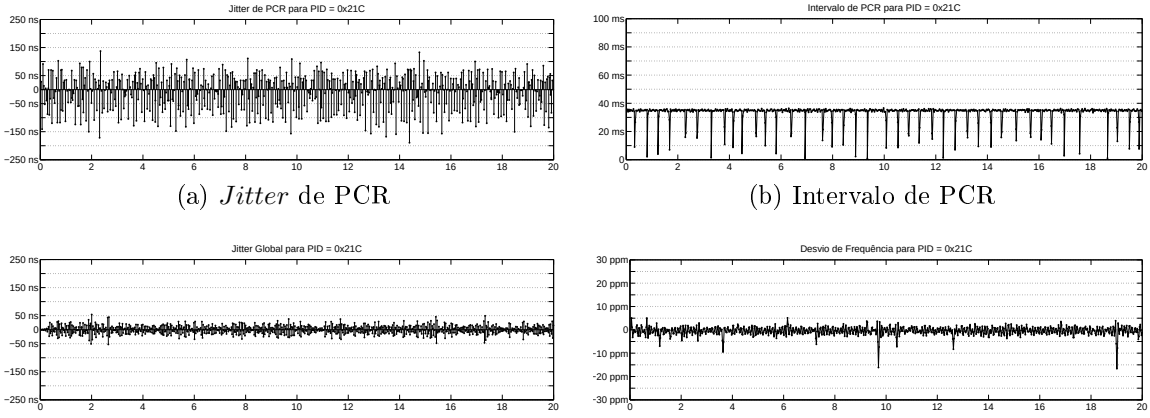
Figura 6.23: Correção integrada com contadores - simulação 2 - PID 0208.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

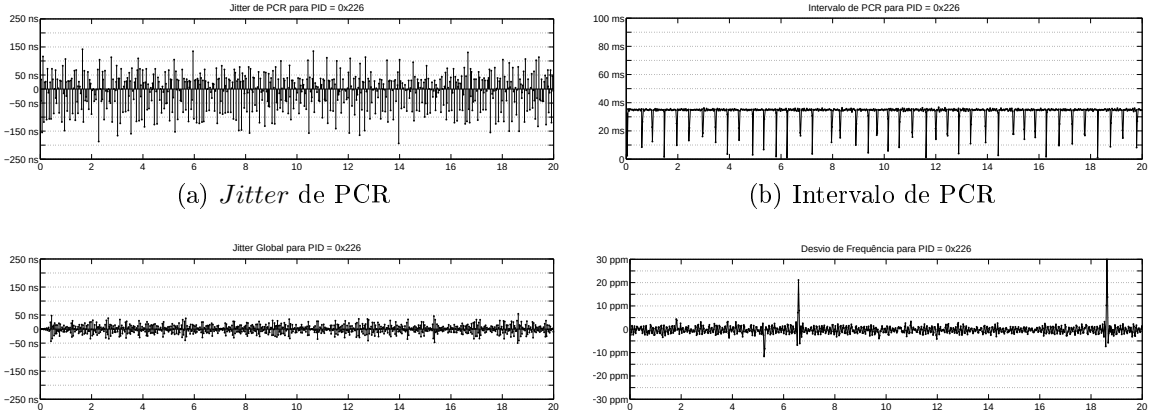
Figura 6.24: Correção integrada com contadores - simulação 2 - PID 0212.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

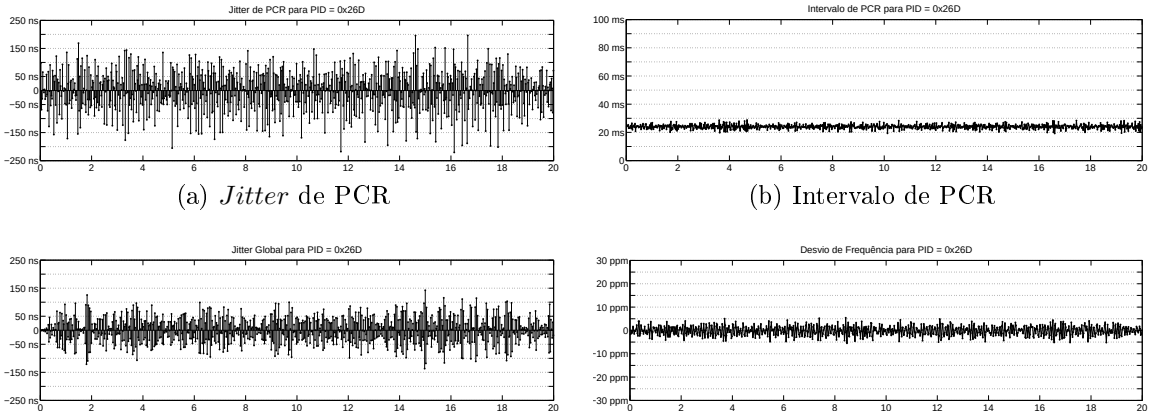
Figura 6.25: Correção integrada com contadores - simulação 2 - PID 021C.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

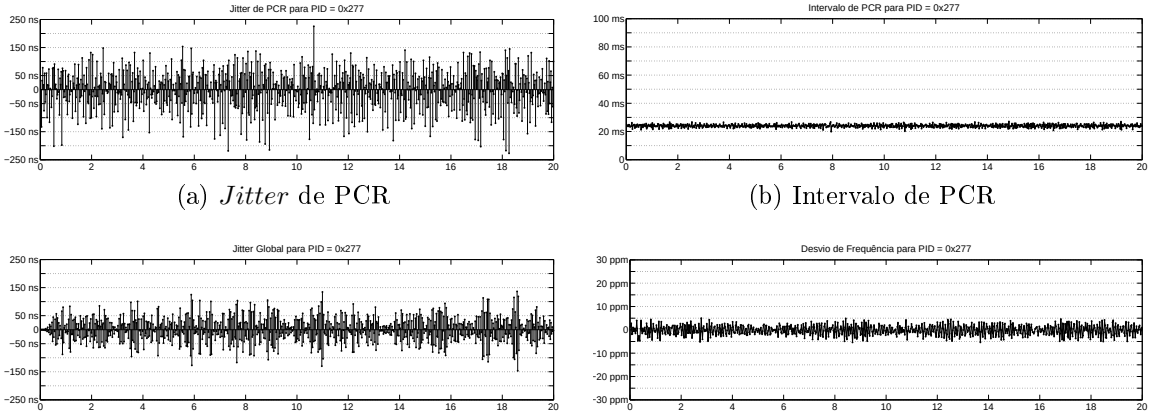
Figura 6.26: Correção integrada com contadores - simulação 2 - PID 0226.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

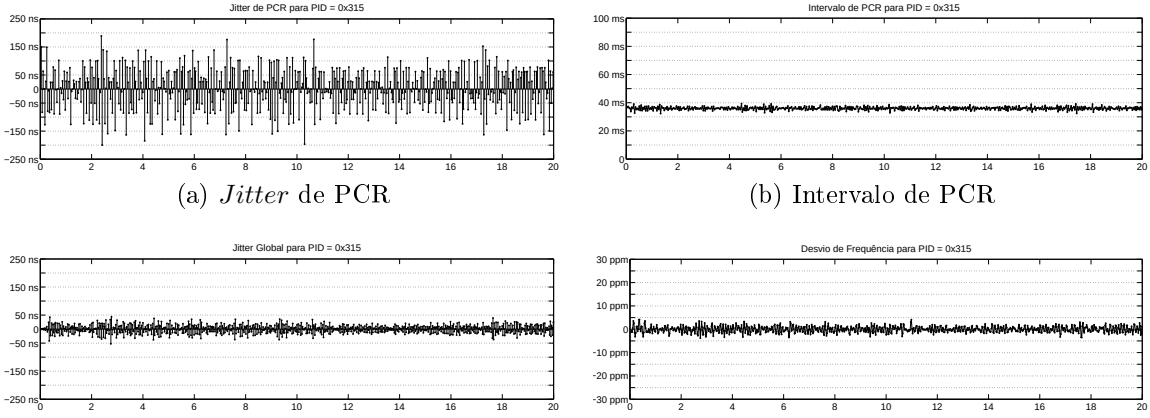
Figura 6.27: Correção integrada com contadores - simulação 2 - PID 026D.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

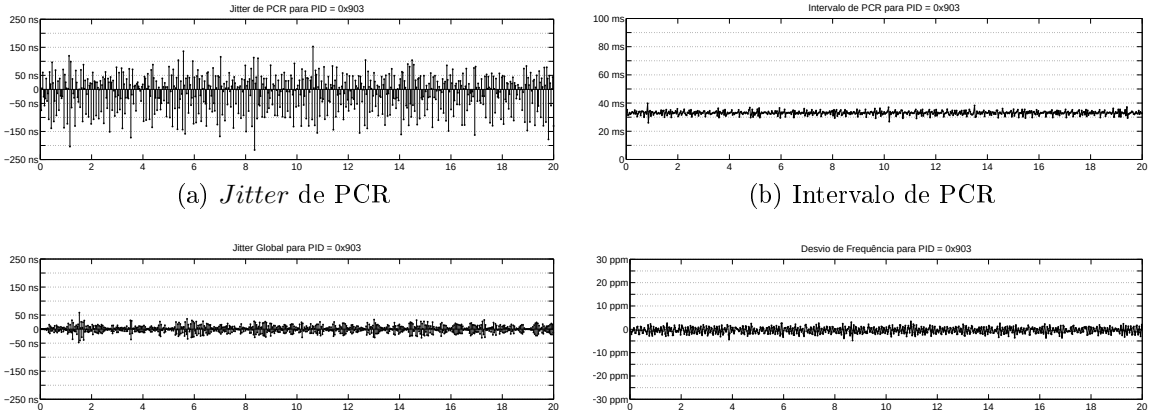
Figura 6.28: Correção integrada com contadores - simulação 2 - PID 0277.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.29: Correção integrada com contadores - simulação 2 - PID 0315.



(a) *Jitter* de PCR (b) Intervalo de PCR

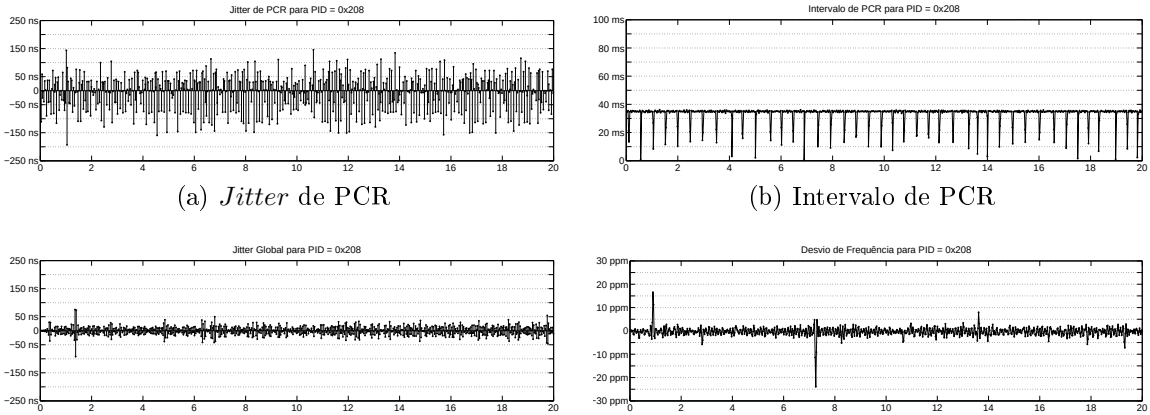
(c) *Jitter* global (d) Desvio de frequência

Figura 6.30: Correção integrada com contadores - simulação 2 - PID 0903.

6.2.2.3 Resultados para a correção integrada com acumuladores

Por último, sob as mesmas condições, o TS é processado com o método de correção de PCR integrada à adaptação de taxa com acumuladores. Os resultados das medições do TS na saída são mostrados nas Figuras 6.31 a 6.38.

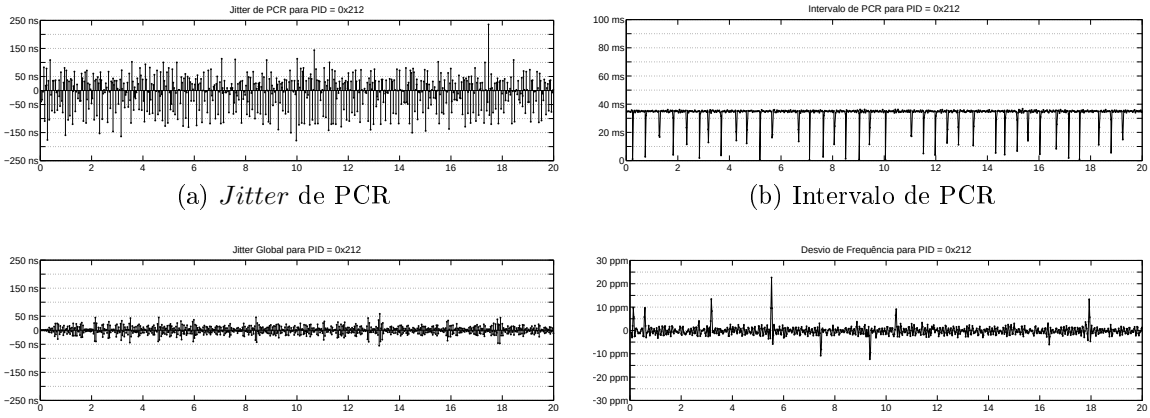
Em princípio, realizando-se uma comparação com os outros métodos testados, é possível afirmar que a operação da correção integrada com acumuladores ocorreu de forma quase transparente, sem qualquer impacto perceptível nas características analisadas. Os envelopes das figuras de mérito são preservados e até mesmo uma comparação numérica revela resultados praticamente iguais.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

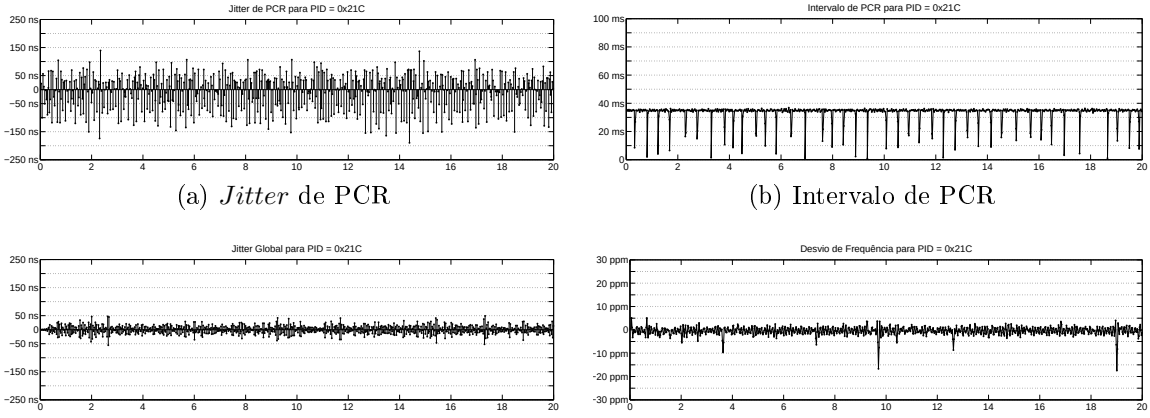
Figura 6.31: Correção integrada com acumuladores - simulação 2 - PID 0208.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

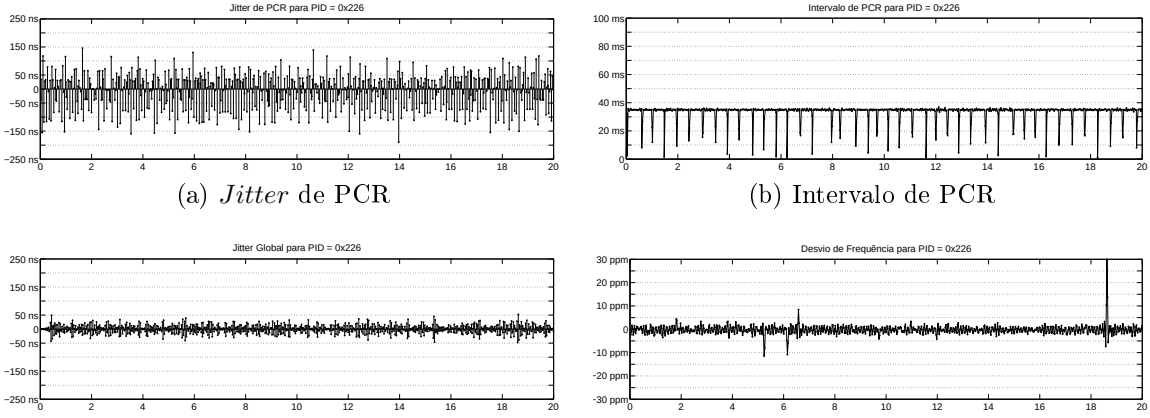
Figura 6.32: Correção integrada com acumuladores - simulação 2 - PID 0212.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

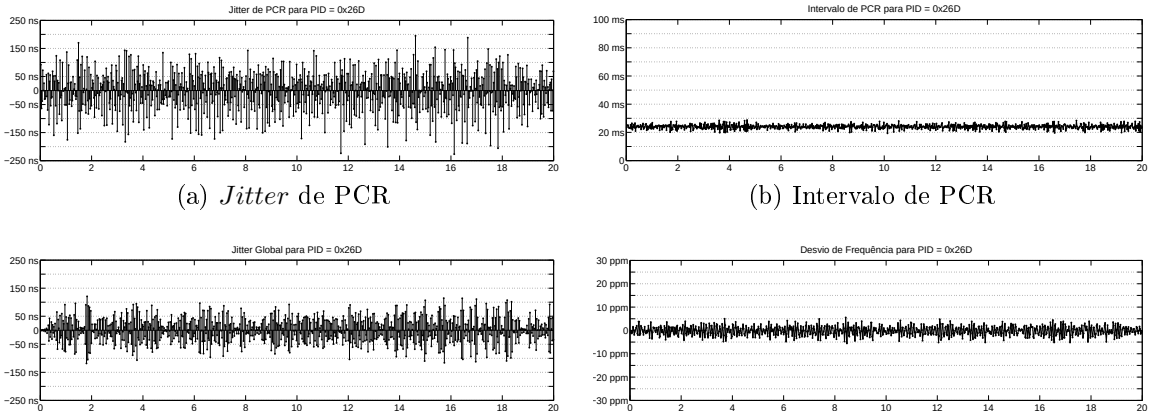
Figura 6.33: Correção integrada com acumuladores - simulação 2 - PID 021C.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

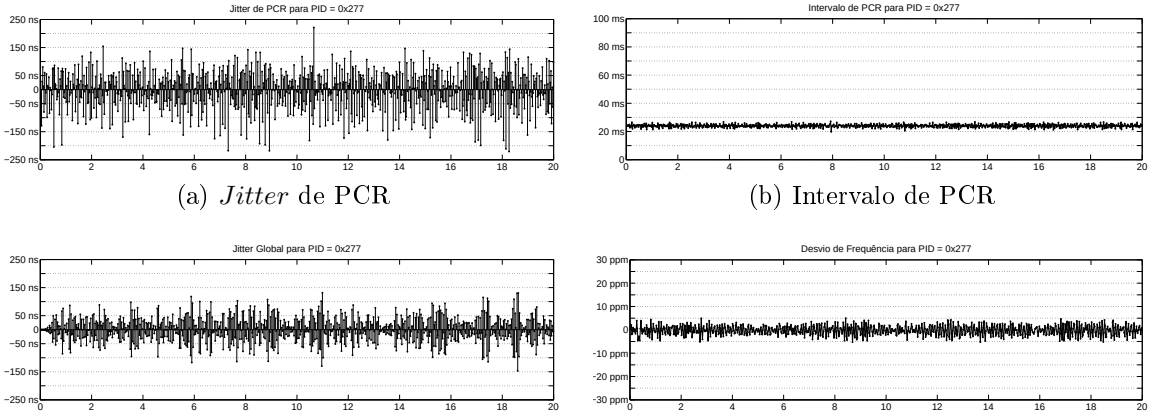
Figura 6.34: Correção integrada com acumuladores - simulação 2 - PID 0226.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

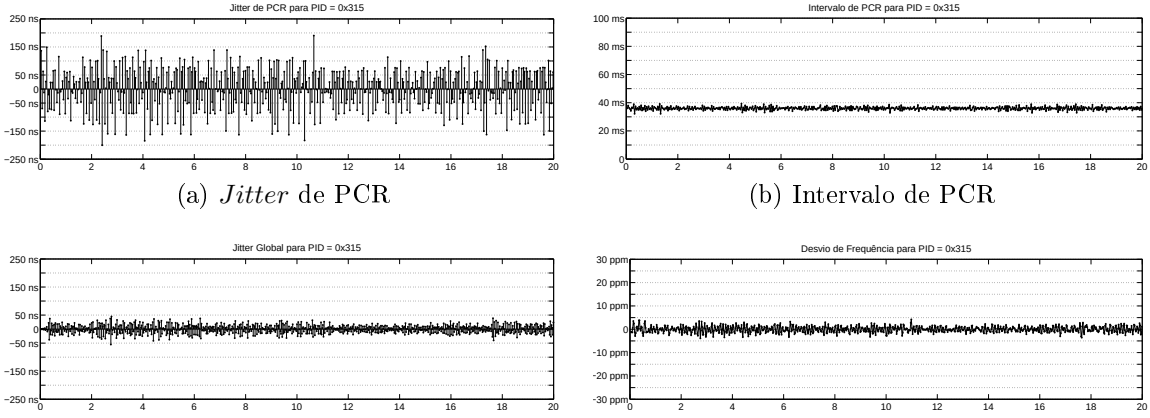
Figura 6.35: Correção integrada com acumuladores - simulação 2 - PID 026D.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

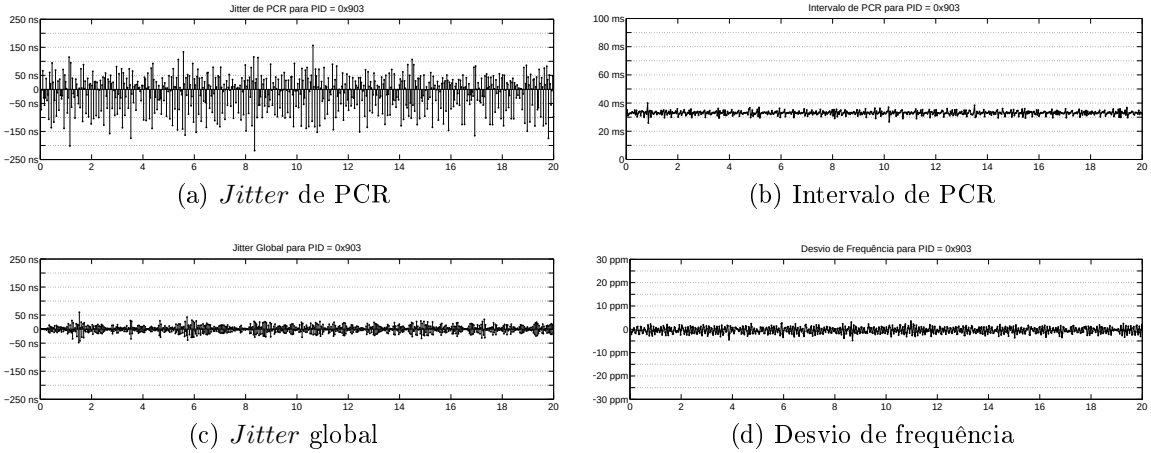
Figura 6.36: Correção integrada com acumuladores - simulação 2 - PID 0277.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.37: Correção integrada com acumuladores - simulação 2 - PID 0315.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.38: Correção integrada com acumuladores - simulação 2 - PID 0903.

6.2.2.4 Considerações sobre os resultados das simulações

Os resultados para o método do contador controlado por semáforo mostram como o perfil do *jitter* de PCR é alterado, em relação ao perfil de entrada, e apresenta maiores picos. No entanto, a estratégia de adaptação de taxa é simples, causando pouca alteração no gráfico de intervalo de PCR.

Para a correção de PCR integrada à adaptação de taxa, o perfil de PCR foi mantido, com maiores picos inseridos pelo método utilizando os contadores, apresentando os melhores resultados com a utilização dos acumuladores.

Dada a alteração no tempo de envio dos pacotes com PCR, esperar-se-ia grande alteração no gráfico do intervalo de PCR. No entanto, como há um grande número de bases de tempo a serem processadas, o valor de max_hold é menor. Assim, o tempo máximo que os pacotes são armazenados no $buffer$ também é reduzido, causando menor alteração no perfil do intervalo de PCR. Deve-se notar, no entanto, que se max_hold for menor, a eficiência da correção integrada é diminuída, já que haverá menos possibilidades de envio do pacote. Isto poderia forçar o módulo de adaptação de taxa a enviar um pacote com PCR mesmo que o *jitter* inserido não fosse o mais adequado.

A Tabela 6.5 mostra o número de adições realizadas em cada abordagem, comparando os métodos propostos com os métodos da literatura. Como descrito anteriormente, o número de operações aritméticas envolvidas com a correção integrada é significativamente maior, o que requer um *hardware* com mais recursos, que execute as operações em tempo hábil.

A Tabela 6.6 contém as medições estatísticas de desvio-padrão para o *jitter* de PCR e o intervalo de PCR e o módulo do valor máximo *jitter* de PCR, para cada programa

Tabela 6.5: Comparação de operações e contadores nos diferentes métodos - Simulação 2.

Método	*Somas*	*Contadores*	*Acumuladores*	*Lógica adicional*
Contadores dedicados [7]	0	8	0	*Não*
Contadores dedicados com janelas deslizantes [3]	0	4	0	*Sim*
Compensação [8, 9]	11080	1	0	*Não*
Semáforos (proposto)	226	1	0	*Sim*
Correção integrada com contadores e semáforos (proposto)	11946 (int.) 5540 (p.f.)	1	0	*Sim*
Correção integrada com acumuladores e semáforos (proposto)	11080 (int.) 6406 (p.f.)	0	1	*Sim*

do TS. Nota-se como o desvio-padrão do intervalo de PCR é mantido pelo método utilizando apenas os semáforos, enquanto o do *jitter* de PCR se mantém próximo ao TS de entrada nos casos de correção integrada, assim como o valor máximo.

Tabela 6.6: Desvio-padrão e valor máximo - Simulação 2.

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 0208			
TS de entrada	$6,3419.10^{-8}$	$6,8821.10^{-3}$	$196,6ns$
Semáforos (proposto)	$6,6962.10^{-8}$	$6,8820.10^{-3}$	$194,1ns$
Correção integrada com contadores (proposto)	$6,3635.10^{-8}$	$6,8845.10^{-3}$	$190,9ns$
Correção integrada com acumuladores (proposto)	$6,3421.10^{-8}$	$6,8951.10^{-3}$	$193,1ns$
PID 0212			
TS de entrada	$6,2176.10^{-8}$	$6,9786.10^{-3}$	$228,6ns$
Semáforos (proposto)	$6,6257.10^{-8}$	$6,9781.10^{-3}$	$189,7ns$
Correção integrada com contadores (proposto)	$6,3403.10^{-8}$	$6,9776.10^{-3}$	$228,9ns$

Continua na página seguinte

Tabela 6.6 – continuação da página anterior

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
Correção integrada com acumuladores (proposto)	$6,2241.10^{-8}$	$6,9884.10^{-3}$	$235,3ns$
PID 021C			
TS de entrada	$6,2972.10^{-8}$	$6,9723.10^{-3}$	$185,2ns$
Semáforos (proposto)	$6,6709.10^{-8}$	$6,9734.10^{-3}$	$192,7ns$
Correção integrada com contadores (proposto)	$6,4099.10^{-8}$	$6,9749.10^{-3}$	$189,3ns$
Correção integrada com acumuladores (proposto)	$6,2956.10^{-8}$	$6,9791.10^{-3}$	$189,3ns$
PID 0226			
TS de entrada	$6,3376.10^{-8}$	$7,0311.10^{-3}$	$191,0ns$
Semáforos (proposto)	$6,7630.10^{-8}$	$7,0321.10^{-3}$	$172,1ns$
Correção integrada com contadores (proposto)	$6,4458.10^{-8}$	$7,0348.10^{-3}$	$193,8ns$
Correção integrada com acumuladores (proposto)	$6,3433.10^{-8}$	$7,0412.10^{-3}$	$191,1ns$
PID 026D			
TS de entrada	$7,0821.10^{-8}$	$1,5518.10^{-3}$	$224,9ns$
Semáforos (proposto)	$7,3986.10^{-8}$	$1,5521.10^{-3}$	$239,9ns$
Correção integrada com contadores (proposto)	$7,2036.10^{-8}$	$1,5640.10^{-3}$	$221,4ns$
Correção integrada com acumuladores (proposto)	$7,0865.10^{-8}$	$1,5724.10^{-3}$	$226,8ns$
PID 0277			
TS de entrada	$7,0086.10^{-8}$	$1,0670.10^{-3}$	$226,5ns$
Semáforos (proposto)	$7,4412.10^{-8}$	$1,0668.10^{-3}$	$240,9ns$
Correção integrada com contadores (proposto)	$7,0660.10^{-8}$	$1,0877.10^{-3}$	$226,9ns$
Correção integrada com acumuladores (proposto)	$7,0079.10^{-8}$	$1,0945.10^{-3}$	$221,1ns$
PID 0315			
TS de entrada	$7,2037.10^{-8}$	$1,1223.10^{-3}$	$189,7ns$
Semáforos (proposto)	$7,4539.10^{-8}$	$1,1217.10^{-3}$	$235,5ns$

Continua na página seguinte

Tabela 6.6 – continuação da página anterior

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
Correção integrada com contadores (proposto)	$7,2244.10^{-8}$	$1,1256.10^{-3}$	$199,9ns$
Correção integrada com acumuladores (proposto)	$7,2236.10^{-8}$	$1,1408.10^{-3}$	$200,7ns$
PID 0903			
TS de entrada	$6,3688.10^{-8}$	$1,4922.10^{-3}$	$219,4ns$
Semáforos (proposto)	$6,8631.10^{-8}$	$1,4908.10^{-3}$	$220,7ns$
Correção integrada com contadores (proposto)	$6,4809.10^{-8}$	$1,5164.10^{-3}$	$217,3ns$
Correção integrada com acumuladores (proposto)	$6,3813.10^{-8}$	$1,5225.10^{-3}$	$219,3ns$

6.2.3 Simulação de um TS com taxa de saída de 15 Mbps

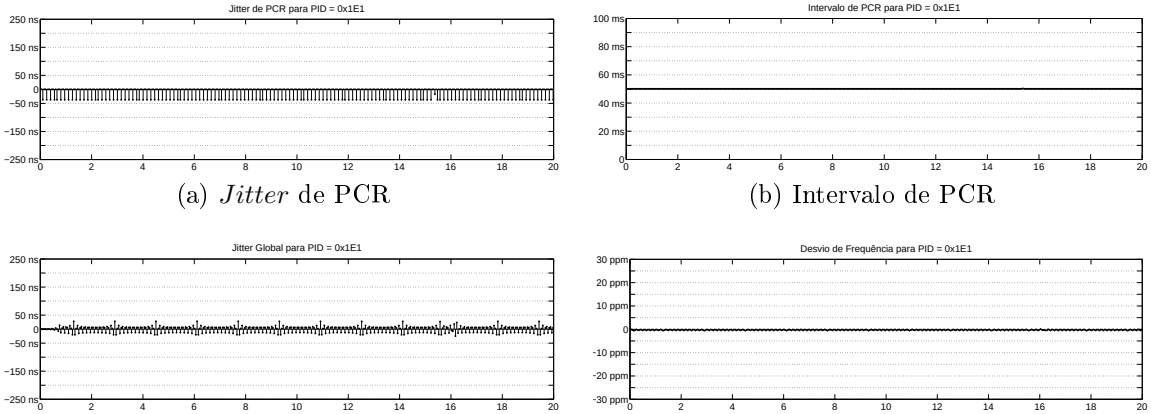
Neste cenário, 20 segundos de um TS, com uma única base de tempo, são processados, em um conversor de taxa, com as três abordagens propostas para se corrigir os valores de PCR. A taxa de bits original é de 10270800 bps e os resultados das medições de PCR, o tamanho dos pacotes é de 188 bytes e a taxa de saída do conversor foi fixada em 15 Mbps, o que faz TP_{saida}

$$TP_{saida} = \frac{8 \cdot 27MHz}{15Mbps} \cdot 188bytes = 2707,2. \tag{6.6}$$

Esta simulação foi escolhida com o objetivo de se avaliar os resultados para TP_{saida} com parte fracionária, de apenas uma casa decimal, que pode se tornar um inteiro a cada 5 pacotes. Na Figura 6.39, que representa as medição da entrada do TS, o *jitter* de PCR apresenta pequenos deslizes de PCR, diretamente ligado à granularidade do relógio de $27MHz$. Seu intervalo de PCR é constante e apresenta pouco desvio de frequência.

6.2.3.1 Resultados com o contador controlado por semáforo

Sob as condições definidas, os resultados referentes à utilização do método de contador controlado por semáforo são mostrados na Figura 6.40. É possível perceber um claro aumento no *jitter* presente no TS, devido principalmente a característica já mencionada

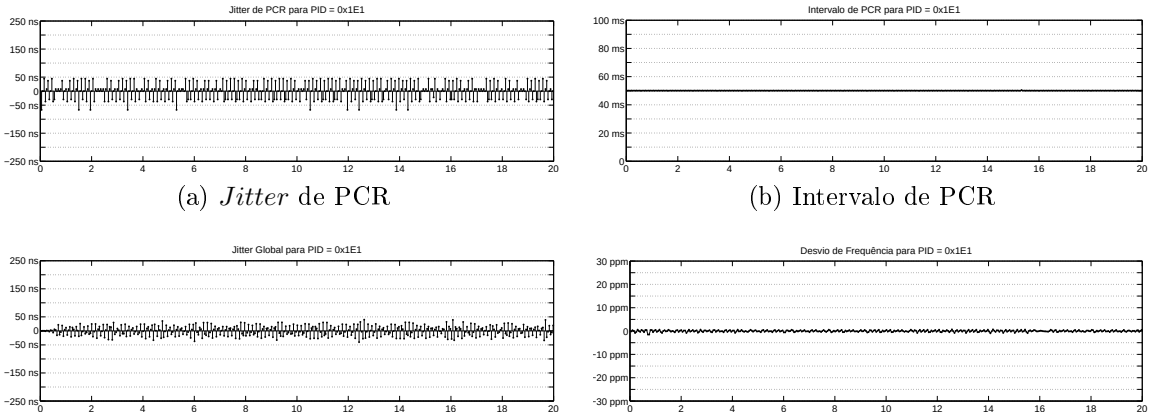


(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.39: Medição de PCR de entrada na simulação 3.

sobre o número de gatilhos de 27 MHz envolvido em cada pacote. Isto faz com que, algumas vezes, o número de pacotes entre PCRs seja um múltiplo de 5 e, em outros momentos, erros se acumulem entre estes. Nota-se grande aumento no *jitter* de PCR, ocasionando maior oscilação no desvio de frequência, que ainda se mantém em níveis bastante estáveis. O intervalo de PCR se mantém constante, como era de se esperar para o método dos semáforos.



(a) *Jitter* de PCR (b) Intervalo de PCR

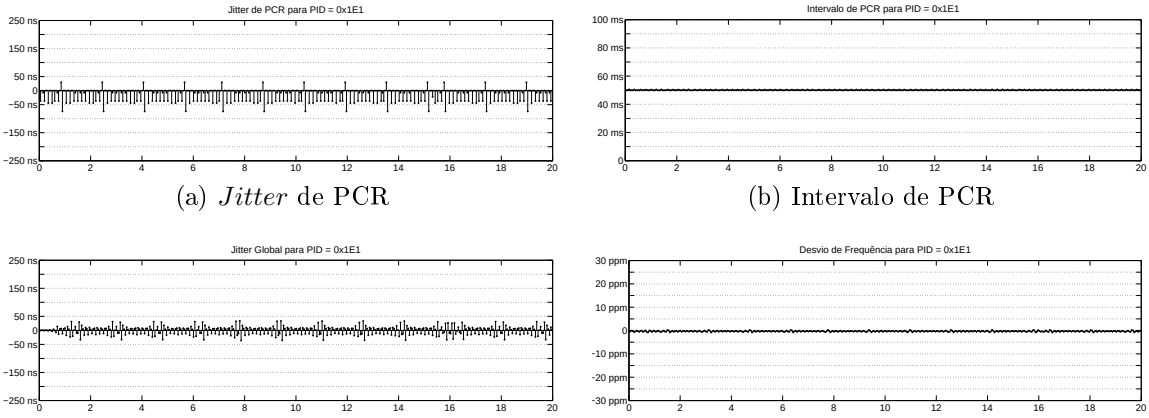
(c) *Jitter* global (d) Desvio de frequência

Figura 6.40: Contador controlado por semáforo - simulação 3.

6.2.3.2 Resultados com a correção integrada utilizando contadores

Para o método de correção de PCR integrada à adaptação de taxa com contadores, os resultados são mostrados na Figura 6.41. Verifica-se, dados da figura, que o *jitter* de PCR resultante é menor quando comparado ao método anterior, mas alguns picos positivos permanecem, o que é devido à granularidade do contador de 27 MHz empregado

na correção. Pode-se notar a alteração do intervalo de PCR, dado que este método opera alterando a cadência de pacotes com PCR.



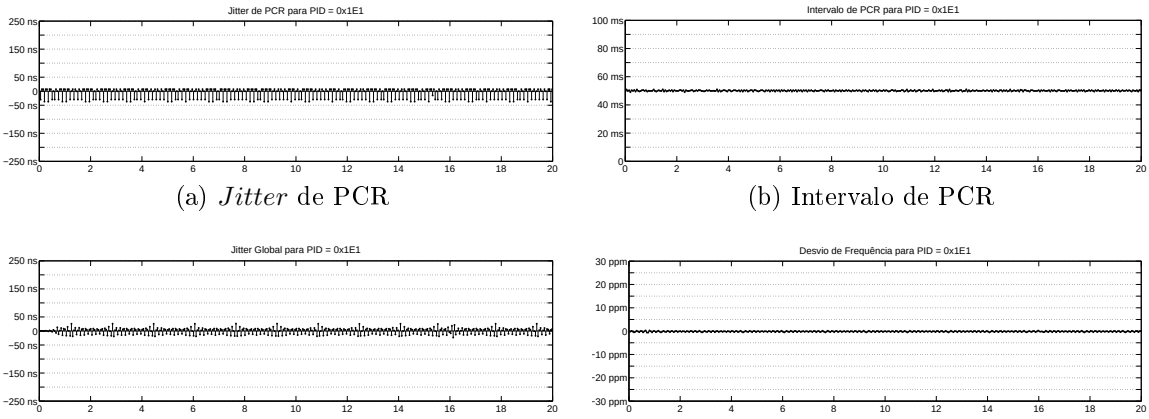
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.41: Correção integrada com contadores - simulação 3.

6.2.3.3 Resultados com a correção integrada utilizando acumuladores

Por último, sob as mesmas condições, o TS é processado com a metodologia de correção de PCR integrada à adaptação de taxa, utilizando acumuladores. Os resultados das medições do TS na saída são mostrados na Figura 6.42. Como esperado, as alterações causadas no *jitter* de PCR são bem pequenas e pouco modificam as características deste. Nota-se, como característica deste método de correção integrada, a alteração do intervalo de PCR. A medição para o desvio de frequência é similar àquela encontrada na entrada.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.42: Correção integrada com acumuladores - simulação 3.

6.2.3.4 Considerações sobre os resultados das simulações

Os resultados mostram como o perfil do *jitter* de PCR é bastante alterado pelo método que utiliza apenas a estrutura de semáforos, sem correção integrada, quando comparado àquele medido na entrada do processador. Os resultados ficaram abaixo do máximo permitido e o módulo de correção não alterou, de forma agressiva, as características do TS. Entretanto, o valor de TP_{saida} (parte fracionária que pode se tornar inteira a cada 5 pacotes) possibilita que os resultados alcançados pelo método de correção integrada sejam bastante satisfatórios, mostrando muito pouca alteração para o caso dos acumuladores. Ainda, há picos inseridos quando se utilizam os contadores, isto se dá pela pequena granularidade do relógio do período de 27 MHz, que em alguns casos causa o deslize de contagem para o gatilho seguinte, incrementando o valor de PCR em 1.

A Tabela 6.7 mostra o número de adições realizadas para cada abordagem, comparando os métodos propostos com os métodos da literatura.

Tabela 6.7: Comparação de operações e contadores nos diferentes métodos - Simulação 3.

Método	*Somas*	*Contadores*	*Acumuladores*	*Lógica adicional*
Contadores dedicados [7]	0	1	0	*Não*
Contadores dedicados com janelas deslizantes [3]	0	1	0	*Sim*
Compensação [8, 9]	1198	1	0	*Não*
Semáforos (proposto)	0	1	0	*Sim*
Correção integrada com contadores e semáforos (proposto)	1198 (int.) 599 (p.f.)	1	0	*Sim*
Correção integrada com acumuladores e semáforos (proposto)	1198 (int.) 599 (p.f.)	0	1	*Sim*

O número de operações aritméticas realizadas pelos métodos de correção integrada mostrou-se, em média, 50% maior que o número de operações aritméticas executadas pelos demais métodos, quando se processam TS com apenas uma base de tempo. Isto é relacionado ao fato do contador, ou acumulador, estar sempre livre para carga, o que não ocorre quando a cadência de pacotes com PCR é maior, como no caso de fluxos de transporte com um maior número de bases de tempo.

A Tabela 6.8 contém as medições estatísticas de desvio-padrão para o *jitter* de PCR e o intervalo de PCR e o módulo do valores máximo do *jitter* de PCR. O desvio-

padrão do intervalo de PCR é mantido na mesma ordem daquele no TS de entrada pelo método utilizando apenas os semáforos, enquanto o do *jitter* de PCR se mantém próximo nos casos de correção integrada com acumuladores, mantendo os níveis do valores máximos bastante próximos aos originais.

Tabela 6.8: Desvio-padrão e valor máximo - Simulação 3.

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 01E1			
TS de entrada	$1,7578.10^{-8}$	$1,4679.10^{-5}$	$37,0ns$
Semáforos (proposto)	$2,9889.10^{-8}$	$5,3098.10^{-5}$	$66,7ns$
Correção integrada com contadores (proposto)	$2,2089.10^{-8}$	$2,0829.10^{-4}$	$74,1ns$
Correção integrada com acumuladores (proposto)	$1,8137.10^{-8}$	$3,8926.10^{-4}$	$37,2ns$

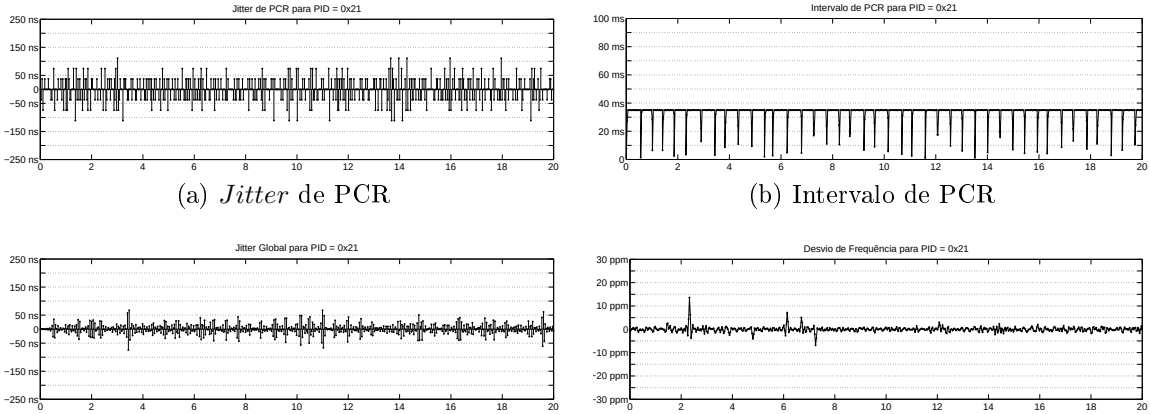
6.2.4 Simulação de um TS com taxa de saída de 27 Mbps

Nesta simulação, 20 segundos de um TS, com uma única base de tempo, são processados em um conversor de taxa, com as três abordagens propostas para a correção dos valores de PCR. A taxa de bits original é de aproximadamente $22,1$ Mbps e o tamanho dos pacotes é de 188 bytes e a taxa de saída do conversor foi fixada em 27 Mbps, o faz TP_{saida} assumir o valor

$$TP_{saida} = \frac{8 \cdot 27MHz}{27Mbps} \cdot 188bytes = 1504. \tag{6.7}$$

Esta simulação foi escolhida para se avaliar os resultados quando o valor de TP_{saida} for um número inteiro. Sendo assim, a espera pelo melhor momento para o envio de pacote não terá efeito e pode-se prever que as medições dos resultados apresentarão o mesmo comportamento, para todos os métodos propostos.

os resultados das medições de PCR, na entrada do processador, podem ser vistas na Figura 6.43. Apresenta picos de *jitter* global e de desvio de frequência. As falhas no intervalo de PCR são causadas por pacotes com PCR muito próximos e se encontram dentro dos padrões aceitáveis.



(a) *Jitter* de PCR (b) Intervalo de PCR

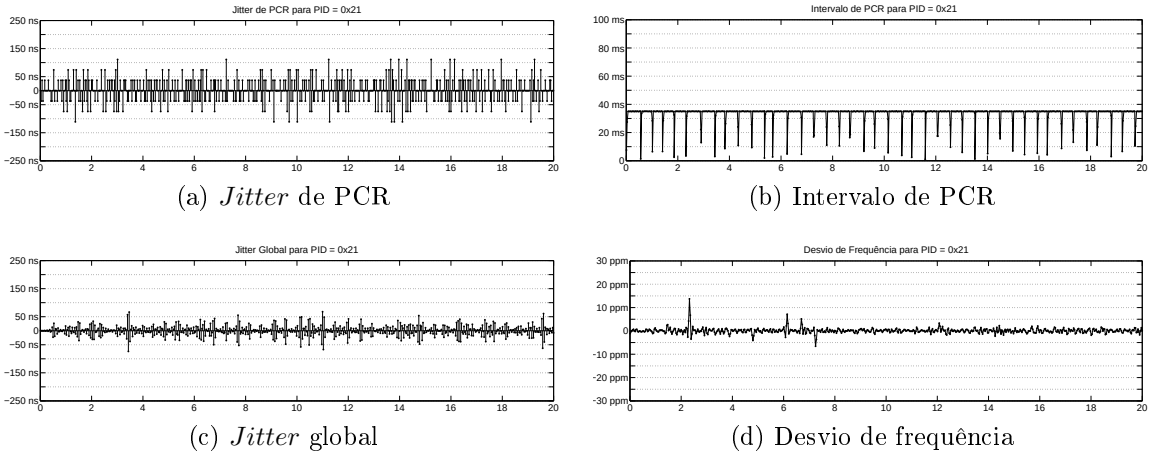
(c) *Jitter* global (d) Desvio de frequência

Figura 6.43: Medição de PCR de entrada na simulação 4.

6.2.4.1 Resultados referentes às três abordagens

Os resultados referentes à utilização do método de contador controlado por semáforo são mostrados na Figura 6.44. Quando se utiliza o método de correção de PCR integrada à adaptação de taxa com contadores, os resultados são mostrados na Figura 6.45. Finalmente, o TS é processado com o método de correção de PCR integrada à adaptação de taxa utilizando acumuladores. Os resultados das medições do TS na saída são mostrados na Figura 6.46.

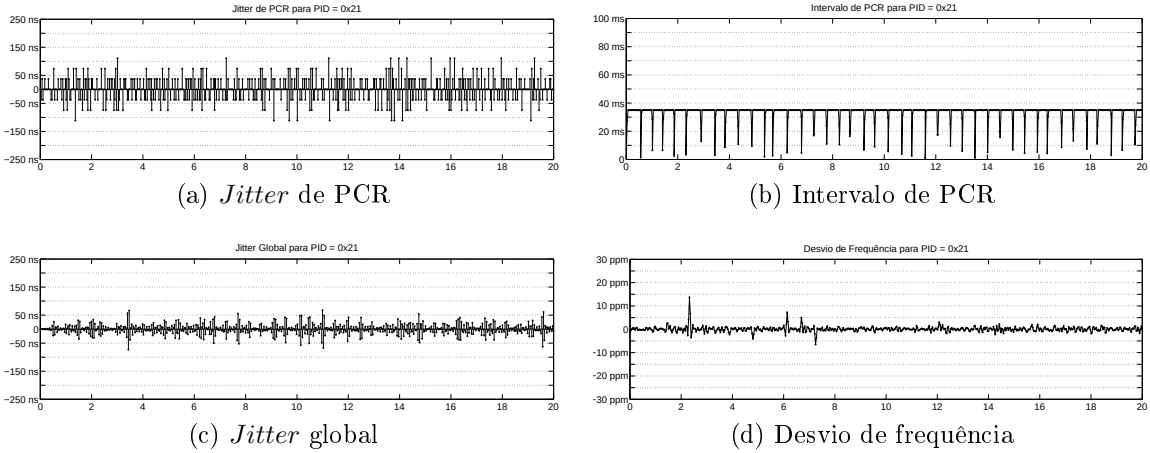
Pode-se notar que o TS não sofreu alterações notáveis nas suas características e que não há diferenças entre as três abordagens.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

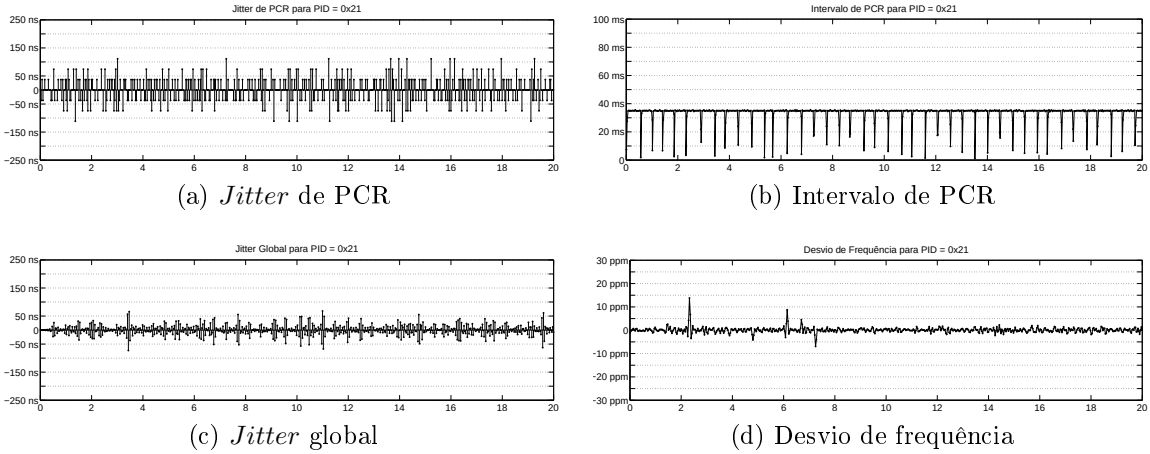
Figura 6.44: Contador controlado por semáforo - simulação 4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.45: Correção integrada com contadores - simulação 4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.46: Correção integrada com acumuladores - simulação 4.

6.2.4.2 Considerações sobre os resultados das simulações

Como já se havia previsto, os resultados são os mesmos e a espera pelo melhor tempo de envio do pacote não tem um efeito significativo sobre o *jitter* de saída. Nestes casos, o único fato que altera os resultados são os tempos de chaveamentos dos circuitos empregados no processamento do TS, que afetam os disparos do relógio e o tempo de execução das operações relacionadas à correção de PCR.

A Tabela 6.9 mostra o número de adições realizadas para cada abordagem, comparando os métodos propostos com os métodos da literatura.

No presente cenário, dentre os métodos propostos, o mais adequado para se lidar com este tipo de situação é aquele que utiliza apenas a estrutura de semáforo, dado que ele demanda menos recursos computacionais e se apresenta com desempenho similar aos demais.

Tabela 6.9: Comparação de operações e contadores nos diferentes métodos - Simulação 4.

Método	*Somas*	*Contadores*	*Acumuladores*	*Lógica adicional*
Contadores dedicados [7]	0	1	0	*Não*
Contadores dedicados com janelas deslizantes [3]	0	1	0	*Sim*
Compensação [8, 9]	1840	1	0	*Não*
Semáforos (proposto)	0	1	0	*Sim*
Correção integrada com contadores e semáforos (proposto)	1840 (int.) 920 (p.f.)	1	0	*Sim*
Correção integrada com acumuladores e semáforos (proposto)	1840 (int.) 920 (p.f.)	0	1	*Sim*

A Tabela 6.10 contém as medições estatísticas de desvio-padrão para o *jitter* de PCR e o intervalo de PCR e o módulo do valores máximo do *jitter* de PCR. Mostra-se, neste cenário de simulação com TP_{saida} inteiro, que não há diferença entre os resultados da adaptação de taxa integrada com a abordagem tradicional, *i.e.*, sem integração entre o adaptador de taxa e o módulo de correção.

Tabela 6.10: Desvio-padrão e valor máximo - Simulação 4.

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 0021			
TS de entrada	$4,1982.10^{-8}$	$7,2962.10^{-5}$	$111,1ns$
Semáforos (proposto)	$4,3047.10^{-8}$	$7,2981.10^{-3}$	$111,1ns$
Correção integrada com contadores (proposto)	$4,3047.10^{-8}$	$7,2981.10^{-3}$	$111,1ns$
Correção integrada com acumuladores (proposto)	$4,3047.10^{-8}$	$7,2981.10^{-3}$	$111,1ns$

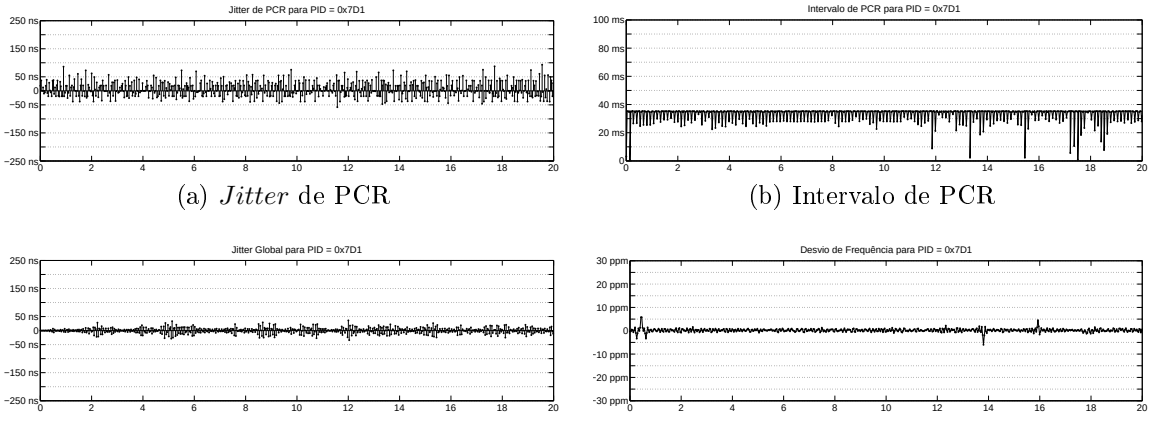
6.2.5 Simulação de um TS 3D com taxa de saída de 35 Mbps e 4 bases de tempo

Nesta simulação, um TS comercial brasileiro, transportando conteúdo 3D, com 4 bases de tempos, é processado. Em resumo, 20 segundos são enviados a um conversor de taxa, que implementa as três abordagens propostas para a correção dos valores de PCR. A taxa de bits original é de aproximadamente 29, 96 Mbps, o tamanho dos pacotes é de 188 bytes e a taxa de saída do conversor foi fixada em 35 Mbps, o que faz TP_{saida} igual a

$$TP_{saida} = \frac{8 \cdot 27MHz}{35Mbps} \cdot 188bytes = 1160,2285, \tag{6.8}$$

de modo que os métodos de correção de PCR integrada à adaptação de taxa podem gerar resultados significativos.

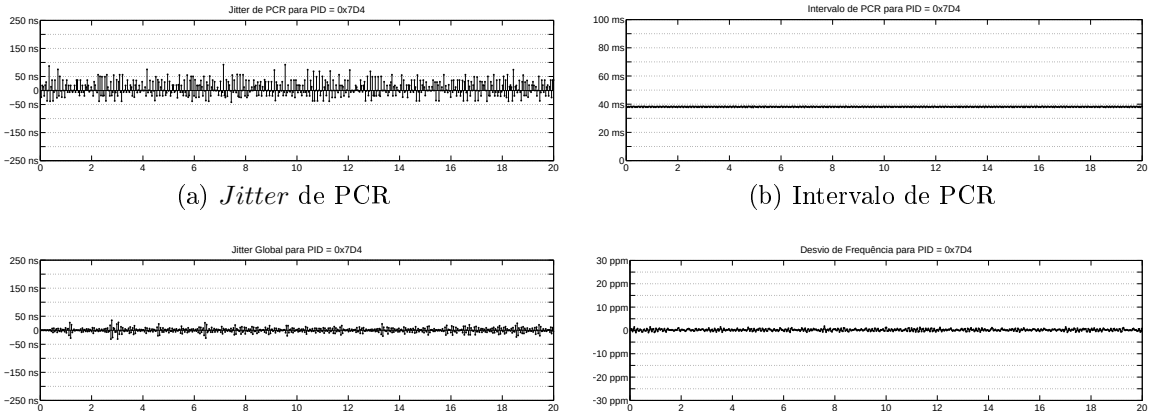
As medições na entrada do processador, podem ser vistos nas Figuras 6.47 a 6.50. Mostram que os níveis de *jitter* de PCR se mantém estáveis em cada base de tempo, apresentando os maiores picos para o PID 0068. Ainda assim, as componentes de alta frequência,*i.e.*, *jitter* global, se mostra mais evidente nos PIDs $07D1$ e 0068, sendo que os desvios de frequência são maiores no primeiro. Os intervalos de PCR são constantes para os PIDs $07D4$ e $1FFE$, apresentando falhas para o PID $07D1$ e maior oscilação no PID 0068.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

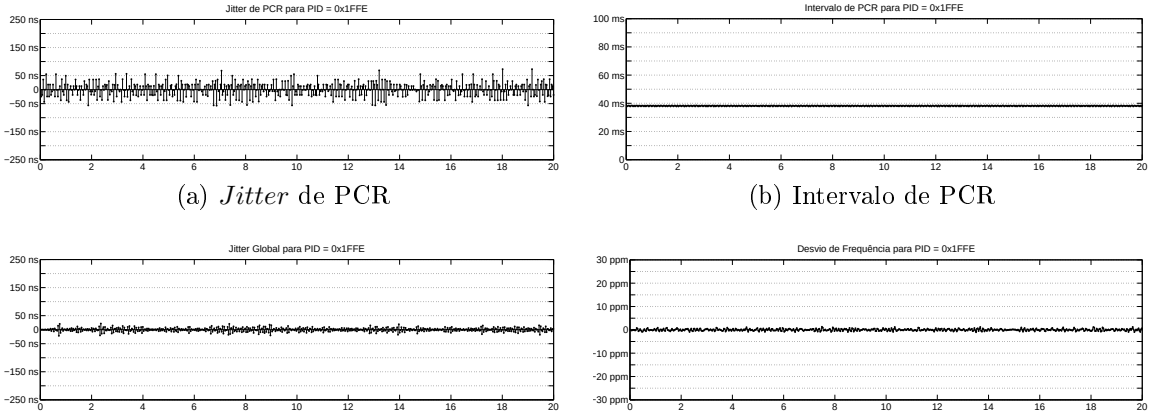
Figura 6.47: Medição de PCR de entrada na simulação 5 - PID 07D1.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

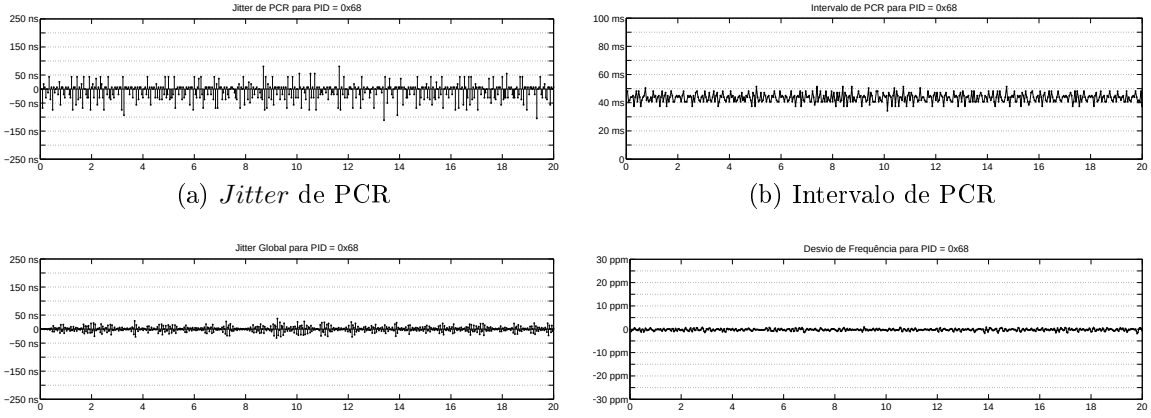
Figura 6.48: Medição de PCR de entrada na simulação 5 - PID 07D4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.49: Medição de PCR de entrada na simulação 5 - PID 1FFE.



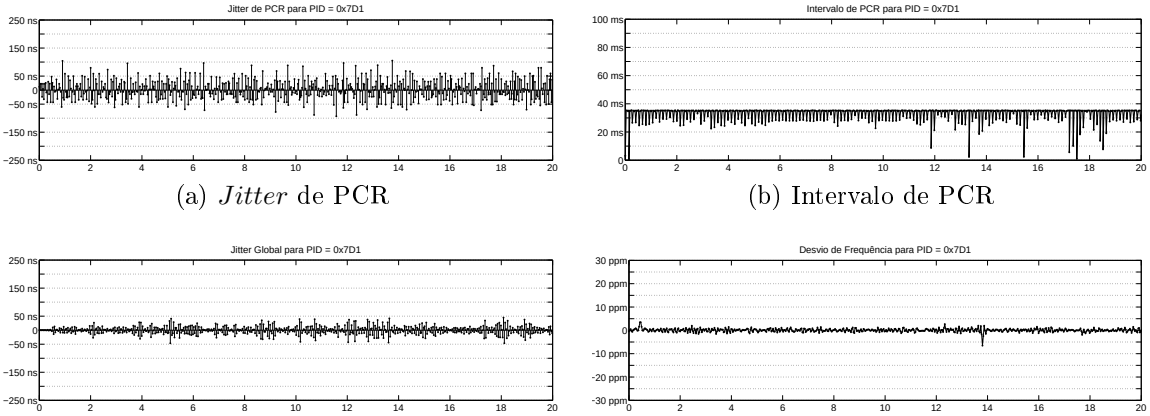
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.50: Medição de PCR de entrada na simulação 5 - PID 0068.

6.2.5.1 Resultados com o contador controlado por semáforo

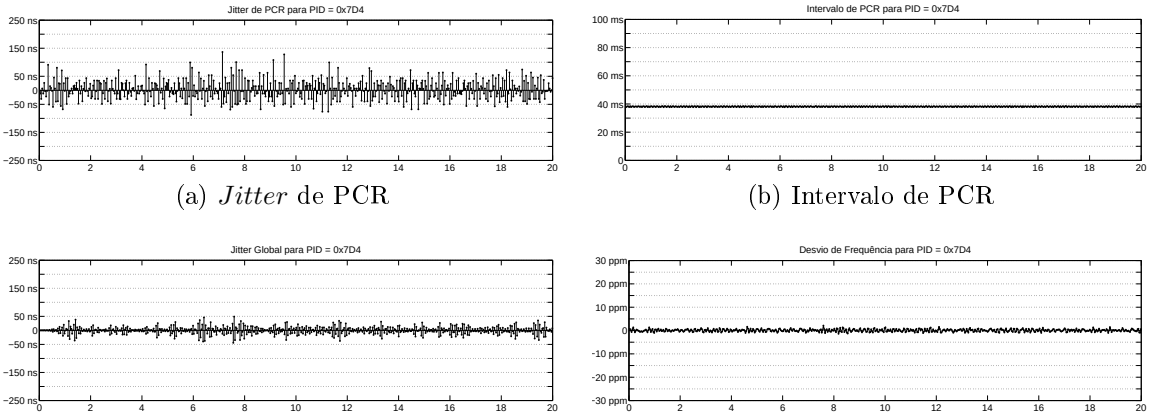
Os resultados referentes à utilização do método de contador controlado por semáforos são mostrados nas Figuras 6.51 a 6.54. Em resumo, como já visto nas simulações anteriores, há um aumento visível no *jitter* do TS em todas as quatro bases de tempo. Ainda assim, este aumento não é visível no desvio de frequência, de modo que a sincronização ainda é estável.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

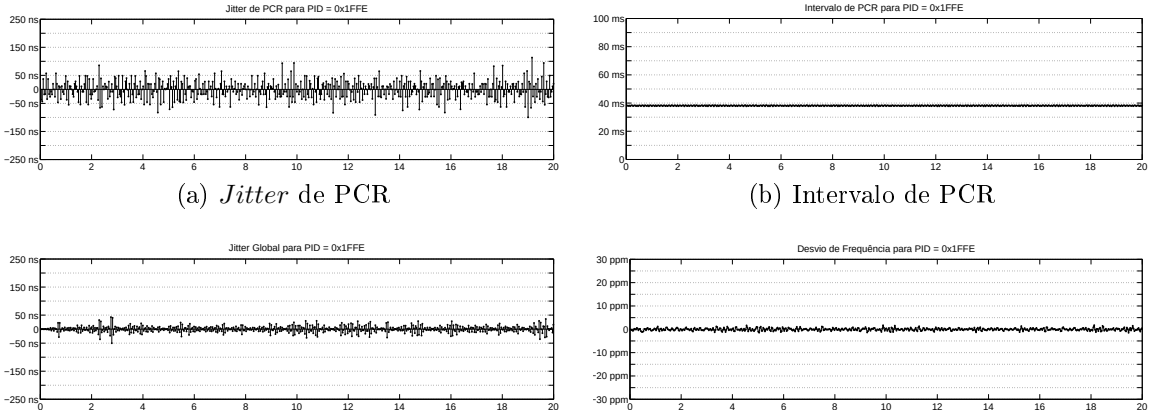
Figura 6.51: Contador controlado por semáforo - simulação 5 - PID 07D1.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

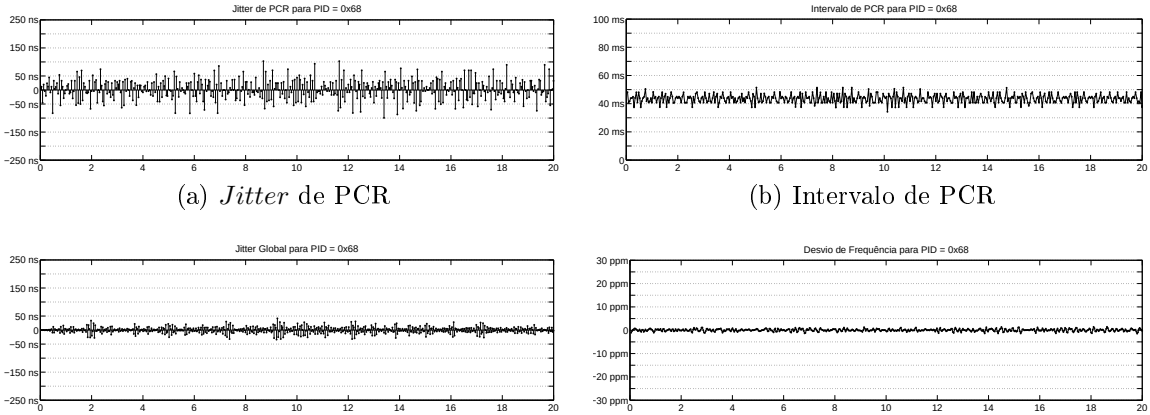
Figura 6.52: Contador controlado por semáforo - simulação 5 - PID 07D4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.53: Contador controlado por semáforo - simulação 5 - PID 1FFE.



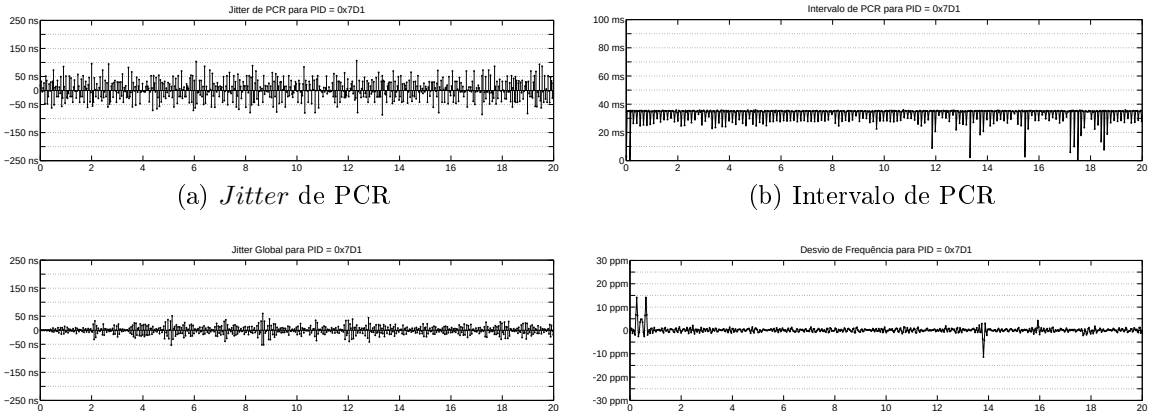
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.54: Contador controlado por semáforo - simulação 5 - PID 0068.

6.2.5.2 Resultados com a correção integrada utilizando contadores

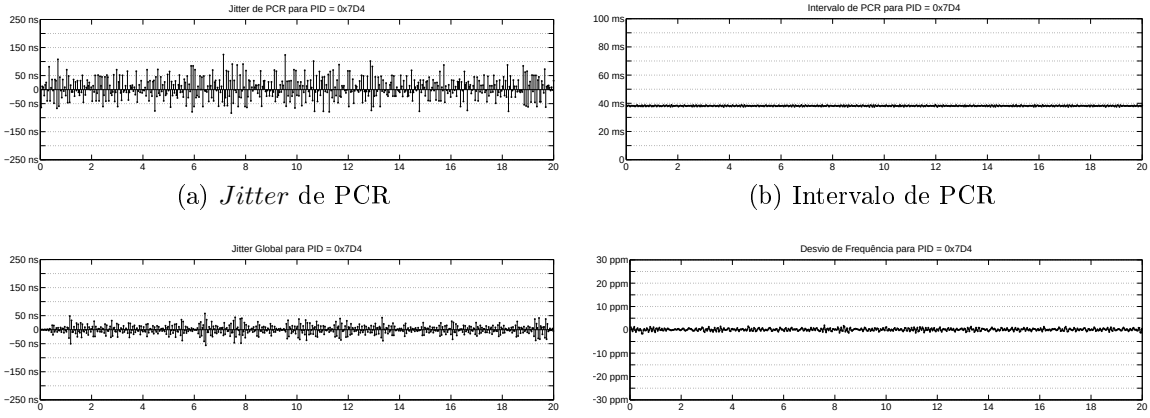
Quando se utiliza o método de correção de PCR integrada à adaptação de taxa com contadores, os resultados são mostrados nas Figuras 6.55 a 6.58. Novamente, nota-se o aumento no *jitter* em todas as bases de tempo e grande alteração no desvio de frequência para o programa de PID $07D1$.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

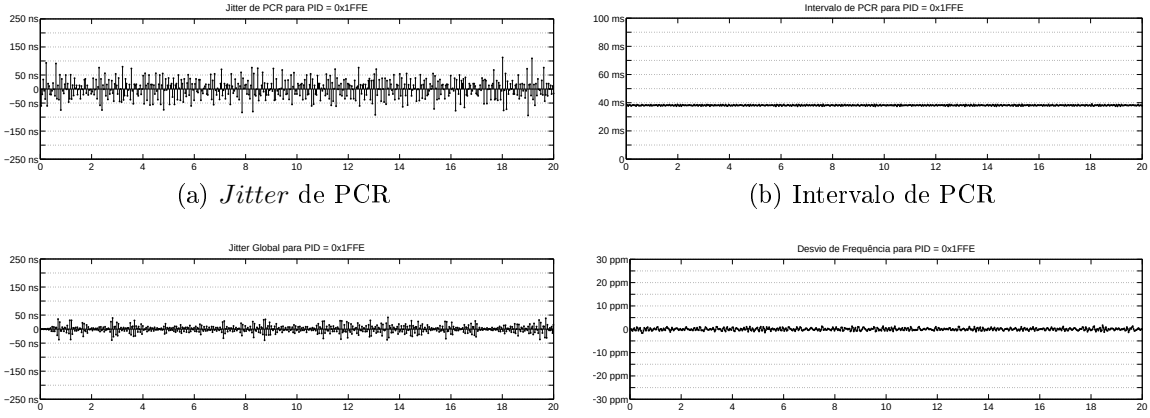
Figura 6.55: Correção integrada com contadores - simulação 5 - PID 07D1.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

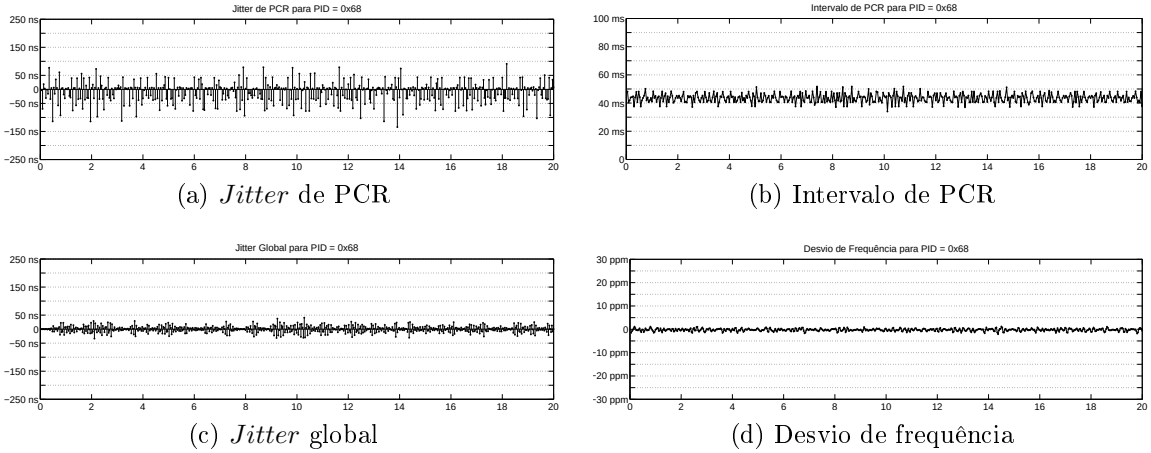
Figura 6.56: Correção integrada com contadores - simulação 5 - PID 07D4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.57: Correção integrada com contadores - simulação 5 - PID 1FFE.



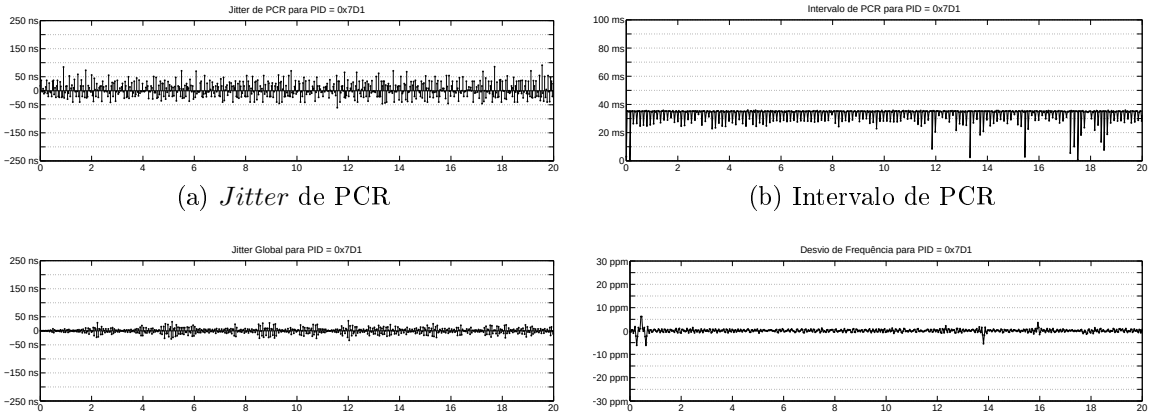
(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.58: Correção integrada com contadores - simulação 5 - PID 0068.

6.2.5.3 Resultados com a correção integrada utilizando acumuladores

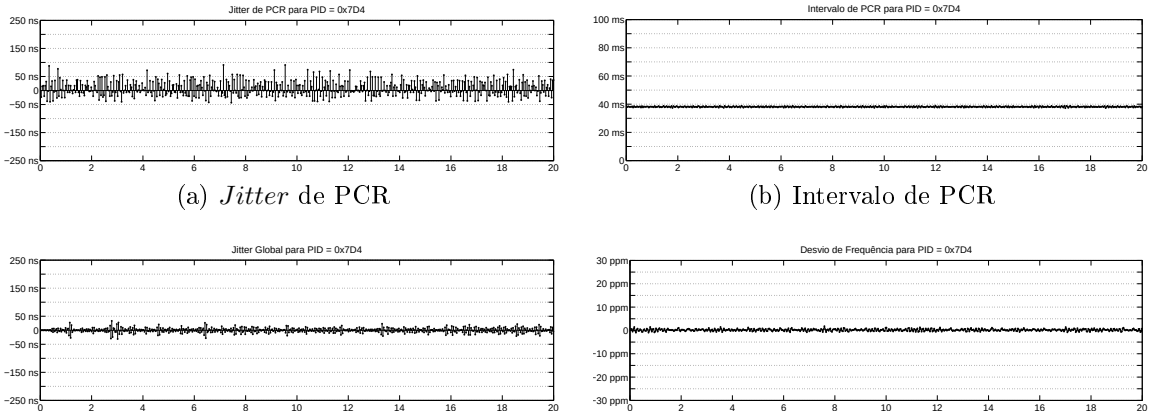
Finalmente, o TS é processado com o método de correção de PCR integrada à adaptação de taxa utilizando acumuladores. Os resultados das medições do TS na saída são mostrados nas Figuras 6.59 a 6.62. Conforme era esperado, a operação da abordagem integrada, com acumuladores, se mostra transparente, causando pouca alteração no *jitter*, alterando apenas, de forma pouco sensível, o intervalo de PCR.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

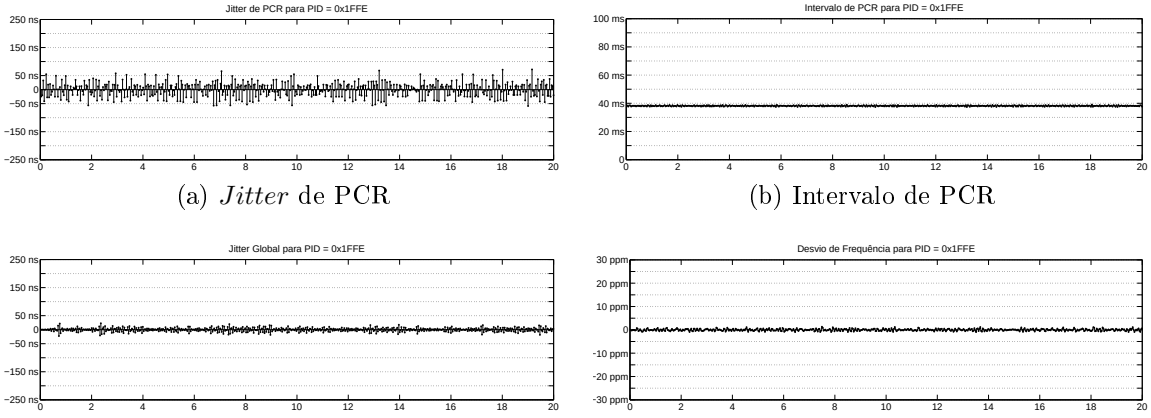
Figura 6.59: Correção integrada com acumuladores - simulação 5 - PID 07D1.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

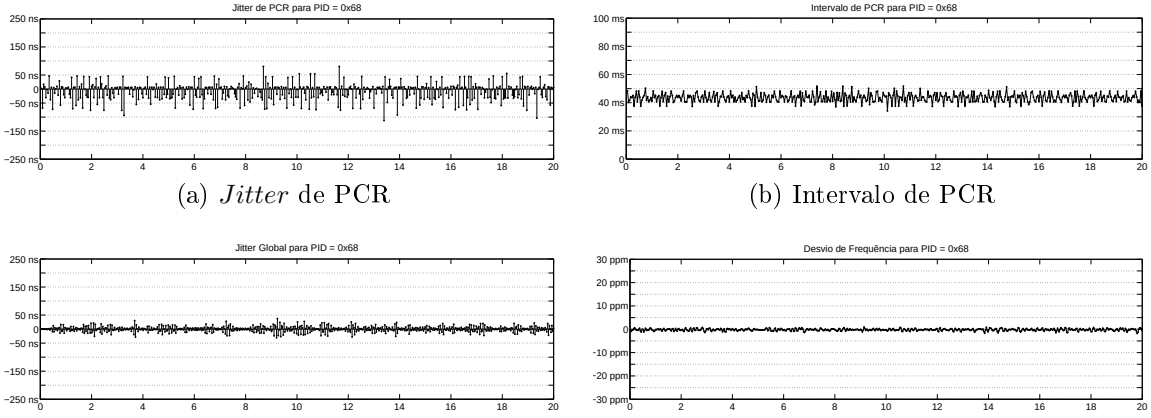
Figura 6.60: Correção integrada com acumuladores - simulação 5 - PID 07D4.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.61: Correção integrada com acumuladores - simulação 5 - PID 1FFE.



(a) *Jitter* de PCR (b) Intervalo de PCR

(c) *Jitter* global (d) Desvio de frequência

Figura 6.62: Correção integrada com acumuladores - simulação 5 - PID 0068.

6.2.5.4 Considerações sobre os resultados das simulações

Os resultados mostram, novamente, como o perfil do *jitter* de PCR é alterado pelos métodos propostos, além do efeito mais agressivo causado pelo método que utiliza apenas a estrutura de semáforos, sem correção integrada. Vale ressaltar que os resultados alcançados pelo método de correção integrada mostram muito pouca alteração para o caso dos acumuladores. Entretanto, há muitos picos inseridos quando se utilizam os contadores. Afirma-se, novamente, que isto é devido à pequena granularidade do relógio do período de 27 MHz, que causa o deslize de contagem para o gatilho seguinte, incrementando o valor PCR em 1.

A Tabela 6.11 mostra o número de adições realizadas para cada abordagem, comparando os métodos propostos com os métodos da literatura.

Tabela 6.11: Comparação de operações e contadores nos diferentes métodos - Simulação 5.

Método	*Somas*	*Contadores*	*Acumuladores*	*Lógica adicional*
Contadores dedicados [7]	0	4	0	*Não*
Contadores dedicados com janelas deslizantes [3]	0	2	0	*Sim*
Compensação [8, 9]	6358	1	0	*Não*
Semáforos (proposto)	46	1	0	*Sim*
Correção integrada com contadores e semáforos (proposto)	6534 (int.) 3179 (p.f.)	1	0	*Sim*
Correção integrada com acumuladores e semáforos (proposto)	6358 (int.) 3355 (p.f.)	0	1	*Sim*

Neste cenário, a estrutura de semáforos foi bastante aproveitada, de modo que as operações de soma realizadas pelos métodos de correção integrada foram bastante reduzidas. Este valor poderia chegar a 15985, dado que foram processados 3179 pacotes com PCR, que sem semáforos exigiriam 5 operações aritméticas em cada correção por compensação, *i.e.*, as 3 operações fixas, segundo a Tabela 5.2, e as duas operações de referências intermediárias. Isto significa que, devido aos semáforos, foi realizado apenas um percentual de operações inicialmente necessárias, dado por

$$reducao = \frac{(6534 + 3179)}{15985} = 0,6076, \tag{6.9}$$

ou seja, $60,76\%$ do que seria exigido sem a estrutura de semáforos.

A Tabela 6.12 mostra os resultados das medições estatísticas de desvio-padrão para o *jitter* de PCR e o intervalo de PCR e o módulo do valores máximo do *jitter* de PCR. Assim como esperado, o desvio-padrão do intervalo de PCR é mantido pela abordagem com semáforos, enquanto o do *jitter* de PCR se mantém próximo ao TS de entrada nos casos de correção integrada, mantendo-se os níveis de valor máximo mais próximos da entrada no caso da correção integrada com acumuladores.

Tabela 6.12: Desvio-padrão e valor máximo - Simulação 5.

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 07D1			
TS de entrada	$2,7246.10^{-8}$	$4,9931.10^{-3}$	$93,2ns$
Semáforos (proposto)	$3,5441.10^{-8}$	$4,9900.10^{-3}$	$114,3ns$
Correção integrada com contadores (proposto)	$3,5321.10^{-8}$	$5,0026.10^{-3}$	$128,0ns$
Correção integrada com acumuladores (proposto)	$2,7391.10^{-8}$	$4,9977.10^{-3}$	$90,9ns$
PID 07D4			
TS de entrada	$2,7161.10^{-8}$	$2,5121.10^{-4}$	$92,0ns$
Semáforos (proposto)	$3,4649.10^{-8}$	$2,4211.10^{-4}$	$109,9ns$
Correção integrada com contadores (proposto)	$3,6467.10^{-8}$	$3,0245.10^{-4}$	$102,6ns$
Correção integrada com acumuladores (proposto)	$2,7211.10^{-8}$	$3,0732.10^{-4}$	$96,2ns$
PID 1FFE			
TS de entrada	$2,5369.10^{-8}$	$2,4834.10^{-4}$	$72,9ns$
Semáforos (proposto)	$3,3046.10^{-8}$	$2,5475.10^{-4}$	$102,7ns$
Correção integrada com contadores (proposto)	$3,5229.10^{-8}$	$3,0715.10^{-4}$	$93,3ns$
Correção integrada com acumuladores (proposto)	$2,5483.10^{-8}$	$3,1530.10^{-4}$	$74,1ns$

Continua na página seguinte

Tabela 6.12 – continuação da página anterior

Método	*Desvio-padrão jitter de PCR*	*Desvio-padrão interv. PCR*	*Máximo*
PID 0068			
TS de entrada	$3,1102.10^{-8}$	$3,1209.10^{-3}$	$111,1ns$
Semáforos (proposto)	$3,4576.10^{-8}$	$3,1201.10^{-3}$	$102,7ns$
Correção integrada com contadores (proposto)	$3,4943.10^{-8}$	$3,1327.10^{-3}$	$116,6ns$
Correção integrada com acumuladores (proposto)	$3,1189.10^{-8}$	$3,1400.10^{-3}$	$112,7ns$

Capítulo 7

Considerações finais

Neste trabalho, propõe-se, inicialmente, uma abordagem de correção de PCR, a ser implementada em equipamentos de TV Digital como remultiplexadores, demultiplexadores e outros, visando à diminuição de recursos computacionais, em número de contadores, e a redução da carga computacional, em número de operações aritméticas. Em seguida, ao se entender o modo como o erro de PCR é inserido no TS, buscou-se um segundo método, capaz de minimizar o *jitter* causado pelo próprio circuito de processamento do Fluxo de Transporte.

Conforme foi exposto, o processamento de fluxos de transporte MPEG-2 adiciona, de forma inerente, *jitter* de PCR, o que é decorrente do reposicionamento dos pacotes de TS. Entretanto, é através da correção de PCR que o *jitter* de saída é mantido dentro de níveis estáveis, que são normatizados. Com base nas abordagens disponíveis na literatura e em sua análise, os novos métodos descritos neste trabalho foram propostos.

O primeiro método apresentado é baseado em um contador controlado por semáforos, com o objetivo de corrigir o *jitter* causado pelo módulo de adaptação de taxa, em um processador de TS MPEG-2. O novo método se mostrou eficaz, no sentido de que mantém o *jitter* de saída dentro dos limites recomendados e próximo ao medido na entrada, segundo o mostrado nos resultados das simulações para o método com semáforos. Além disso, ele exige uma configuração de hardware simples para MPTSs com diferentes bases de tempo e requer menor poder de processamento, pois reduz o número de operações aritméticas exigidas quando comparado aos métodos presentes na literatura. A redução pode chegar até 98%, como visto na Simulação 2 (ver Seção 6.2.2). A lógica adicional para a estrutura de semáforos é simples e de fácil implementação.

Com o objetivo de se minimizar o *jitter* inserido pelo processo de correção, o método baseado em correção de PCR integrada à adaptação de taxa foi apresentado. Tal método introduz menos *jitter*, quando comparado aos esquemas tradicionais, nos quais a adaptação de taxa e a correção de PCR trabalham de forma independente. Isto é

obtido através da troca de informações entre os módulos envolvidos, fazendo com que o tempo utilizado no processo de correção seja o mais próximo possível de um número inteiro de períodos do relógio 27 MHz. Duas abordagens para a correção integrada foram apresentadas, sendo uma baseada em contadores e outra em acumuladores. A primeira mostrou que ainda ocorrem erros de correção pelo gatilho do contador, causando picos no *jitter* de PCR, enquanto a segunda se mostrou mais estável, no sentido de manter o *jitter* em níveis baixos, próximos àquele da entrada, chegando a adicionar apenas $2,8ns$, como visto na Simulação 1 (ver Seção 6.2.1). Os resultados mostram que o método é eficaz e pode ser empregado em sistemas que exigem bases de tempo precisas.

As simulações realizadas embasaram as conclusões mencionadas acima, mostrando que a estrutura de semáforos é capaz de reduzir o número de operações aritméticas envolvidas no processo de correção e que a abordagem de correção de PCR integrada à adaptação de taxa pode minimizar o *jitter* inserido. Embora a utilização dos contadores, na abordagem integrada, seja a causadora de picos de *jitter*, mostrou-se que a utilização de acumuladores é capaz de processar o Fluxo de Transporte de forma mais transparente. Contudo, a abordagem integrada demanda maior carga computacional, dado o grande número de operações aritméticas com números em ponto flutuante.

A escolha da abordagem mais adequada é sempre motivada pela aplicação-alvo. A primeira abordagem, que envolve apenas a estrutura de semáforos, é direcionada para a minimização dos recursos necessários para a correção de PCR, em situações nas quais a disponibilidade de hardware é limitada. Seus resultados referentes ao *jitter* inserido são análogos aos métodos da literatura, variando apenas pela implementação do *hardware*. Já a abordagem de correção integrada visa a diminuição do *jitter* inserido na correção e exige um hardware com mais recursos computacionais, sendo mais indicada para aplicações em que a precisão da base de tempo de saída é uma prioridade.

Para trabalhos futuros, sugere-se a implementação, em FPGA, das metodologias propostas, de modo que se desenvolvam abordagens eficientes e otimizações adequadas aos circuitos-alvo. Além disso, sugere-se a expansão dos resultados a aplicações em dispositivos com funcionamento diferentes dos conversores de interface, como remultiplexadores e demultiplexadores. Isto permitiria a verificação dos resultados apresentados, em sistemas reais, e facilitaria a adoção das novas metodologias em processadores de TS comerciais, como os utilizados em sistemas de radiodifusão.

Apêndice A

Lista de publicações

Artigos completos publicados em congressos:

- Savino, H. e Filho, E. B. L. Correção de PCR integrada à adaptação de taxa para processadores de fluxos de transporte MPEG-2. *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2012, Ouro Preto, MG.*, 2012. páginas 436-447.

- Savino, H. e Filho, E. B. L. Um Contador Controlado por Semáforo Para a Correção de PCR. *SBrT 2012 - XXX Simpósio Brasileiro de Telecomunicações.* 2012.

- Savino, H. e Filho, E. B. L. PCR Jitter Control Based on a Semaphore-Controlled Counter. *20th International Conference on Software, Telecommunications and Computer Networks - SoftCOM 2012.* 2012.

Artigo publicado durante o desenvolvimento do presente trabalho, que utiliza o método de compressão de vídeo do Sistema Brasileiro de TV Digital em sinais de eletroencefalograma:

- Savino, H.; Filho, E. B. L. e Silva Júnior, W. S. da. Compressão de Sinais de EEG com JPEG2000 e H.264. *SBrT 2012 - XXX Simpósio Brasileiro de Telecomunicações.* 2012.

Artigos completos submetidos a congressos:

- Savino, H. e Filho, E. B. L. Joint PCR Correction and Rate Adaptation for Reducing PCR Jitter in MPEG-2 TS Processors. *10th IEEE Consumer Communications and Networking Conference - CCNC 2013.* (Submetido)

Por último, este trabalho recebeu recomendação, pela *IEEE Broadcast Technology Society*, para ser submetido ao *IEEE Transactions on Broadcasting.*

Apêndice B

Código dos simuladores desenvolvidos no Matlab

O diagrama em blocos, utilizado no ambiente Simulink/Matlab, pode ser visto na Figura 8.1. Pelo diagrama, o TS é lido pelo bloco TSin, o tratamento do *buffer* de entrada é executado no bloco BUFFER, o adaptador de taxa e o controle de saída de pacotes são realizados no bloco ADAPT e, finalmente, o bloco WriteTS é o responsável pela interface de saída do TS. O bloco CLOCK é o responsável pelo controle do contador local, fixado na frequência de 27MHz. Alguns dados podem ser visualizados durante a simulação: o número de pacotes com PCR processados, em Display PCR, o número máximo de pacotes que foram armazenados no *buffer*, em Display maxPCR, o número de pacotes que foram processados diretamente, utilizando-se apenas o contador, em Display Contador, o número de pacotes processados através de operações intermediárias, ou seja, com compensação, em Display Compensação, e o número máximo de pacotes que foram armazenados no *buffer*, em Display MaxBuffer.

Códigos em comum aos métodos

O código para TSin.c:

```
1  #define S_FUNCTION_NAME  TSin
2  #define S_FUNCTION_LEVEL 2
3
4  #define NUM_OUTPUTS           1
5
6  #define OUTPUT_0_DTYPE        uint8_T
7  #define OUTPUT_0_COMPLEX      COMPLEX_NO
8
9  #include "simstruc.h"
10 #include <stdio.h>
11
12 #define ARQUIVOFONTE     "MUX.ts"
13 //MUX=0.00000033156586622 //VLC=0.000002  //Sony=0.0000004  ...
       //MUX=0.00000033195
```

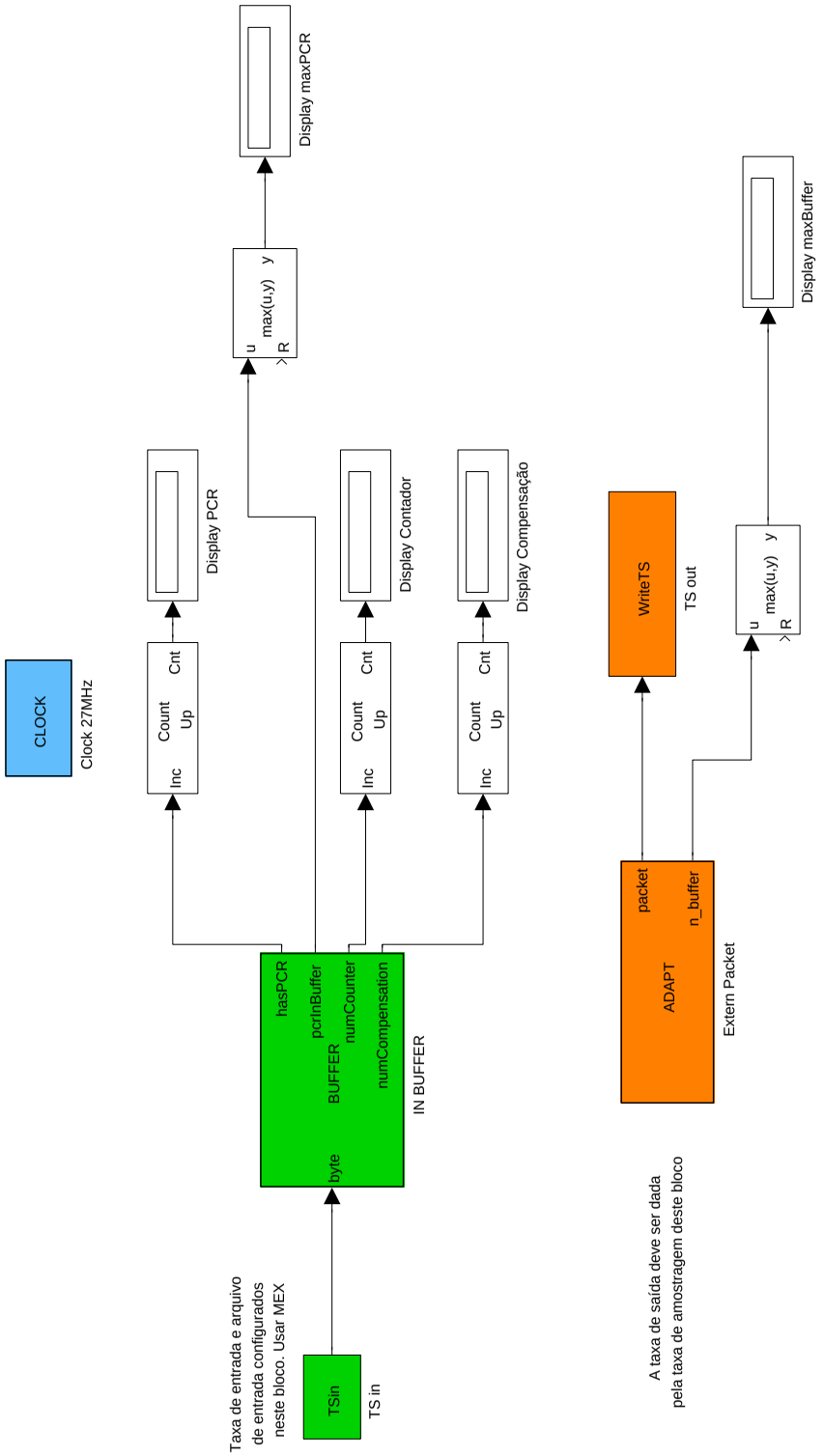


Figura 8.1: Diagrama de blocos do Adaptador de Taxa no Simulink/Matlab

```
14 #define RATE     ...
       3.31565866222456823492900511672 44e-7//0.000002//(8/24127936) ...
       //0.00000033156587  //  8/rate
15
16 static void mdlInitializeSizes(SimStruct *S)
17 {
18     ssSetNumSFcnParams(S, 0);
19     if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
20         return;
21     }
22
23     ssSetNumContStates(S, 0);
24     ssSetNumDiscStates(S, 0);
25
26     if (!ssSetNumInputPorts(S, 0)) return;
27     if (!ssSetNumOutputPorts(S, 1)) return;
28     ssSetOutputPortWidth(S, 0, 1);
29     ssSetOutputPortDataType(S,0,SS_UINT8);
30     ssSetOutputPortComplexSignal(S, 0, OUTPUT_0_COMPLEX);
31
32     ssSetNumPWork(S,1); //pointers in memory
33     ssSetNumSampleTimes(S, 1);
34 }
35
36 static void mdlInitializeSampleTimes(SimStruct *S)
37 {
38     ssSetSampleTime(S, 0, RATE);
39     ssSetOffsetTime(S, 0, 0.0);
40 }
41
42 #define MDL_START  /* Change to #undef to remove function */
43 #if defined(MDL_START)
44   static void mdlStart(SimStruct *S)
45   {
46       void** pwork = ssGetPWork(S);
47       FILE *datafile;
48
49       datafile = fopen(ARQUIVOFONTE,"rb");
50       pwork[0] =  datafile;
51  }
52 #endif /*  MDL_START */
53
54 static void mdlOutputs(SimStruct *S, int_T tid)
55 {
56     uint8_T        *y  = ssGetOutputPortSignal(S,0);
57     void** pwork = ssGetPWork(S);
58
59     fread(y, 1, 1, pwork[0]); //sends pwork to y output
60 }
61
62 static void mdlTerminate(SimStruct *S)
63 {
64     //close the file
65     void** pwork = ssGetPWork(S);
```

```
66        FILE *datafile;
67
68        datafile = pwork[0];
69        fclose(datafile);
70  }
71
72  /*=============================*
73   * Required S-function trailer *
74   *=============================*/
75
76  #ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a ...
       MEX-file? */
77  #include "simulink.c"      /* MEX-file interface mechanism */
78  #else
79  #include "cg_sfun.h"       /* Code generation registration function */
80  #endif
```

O código para WriteTS.c:

```
1   /* Give S-function a name */
2   #define S_FUNCTION_NAME  WriteTS
3   #define S_FUNCTION_LEVEL 2
4
5   #define NUM_INPUTS          1
6   /* Input Port  0 */
7   #define IN_PORT_0_NAME      u0
8   #define INPUT_0_WIDTH       188
9   #define INPUT_DIMS_0_COL    1
10  #define INPUT_0_DTYPE       uint8_T
11  #define INPUT_0_COMPLEX     COMPLEX_NO
12  #define IN_0_FRAME_BASED    FRAME_NO
13  #define IN_0_BUS_BASED      0
14  #define IN_0_BUS_NAME
15  #define IN_0_DIMS           1-D
16  #define INPUT_0_FEEDTHROUGH 1
17  #define IN_0_ISSIGNED        0
18  #define IN_0_WORDLENGTH      8
19  #define IN_0_FIXPOINTSCALING 1
20  #define IN_0_FRACTIONLENGTH  9
21  #define IN_0_BIAS            0
22  #define IN_0_SLOPE           0.125
23
24  /* Include SimStruct definition and file I/O functions */
25  #include "simstruc.h"
26  #include <stdio.h>
27
28
29  /* Called at the beginning of the simulation */
30  static void mdlInitializeSizes(SimStruct *S)
31  {
32      ssSetNumSFcnParams(S, 0);
33      if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
34          return;
```

```
35      }
36
37      ssSetNumContStates(S, 0);
38      ssSetNumDiscStates(S, 0);
39
40      if (!ssSetNumInputPorts(S, NUM_INPUTS)) return;
41      ssSetInputPortWidth(S, 0, INPUT_0_WIDTH);
42      ssSetInputPortDataType(S, 0, SS_UINT8);
43      ssSetInputPortComplexSignal(S, 0, INPUT_0_COMPLEX);
44      ssSetInputPortDirectFeedThrough(S, 0, INPUT_0_FEEDTHROUGH);
45      ssSetInputPortRequiredContiguous(S, 0, 1); /*direct input signal ...
            access*/
46
47      if (!ssSetNumOutputPorts(S, 0)) return;
48
49      ssSetNumPWork(S,1);
50      ssSetNumSampleTimes(S, 1);
51  }
52
53
54  /* Set sample times for the block */
55  static void mdlInitializeSampleTimes(SimStruct *S)
56  {
57      ssSetSampleTime(S, 0, -1);
58      ssSetOffsetTime(S, 0, 0);
59  }
60
61  #define MDL_START  /* Change to #undef to remove function */
62  #if defined(MDL_START)
63    /* Function: mdlStart ...
          =======================================================
64     * Abstract:
65     *    This function is called once at start of model execution. If you
66     *    have states that should be initialized once, this is the place
67     *    to do it.
68     */
69    static void mdlStart(SimStruct *S)
70    {
71        /*at start of model execution, open the file and store the pointer
72         *in the pwork vector */
73        void** pwork = ssGetPWork(S);
74        FILE *datafile;
75
76        datafile = fopen("outputTS.ts","wb");
77        pwork[0] =  datafile;
78
79    }
80  #endif /*  MDL_START */
81
82
83  /* Function: mdlOutputs ...
       =====================================================
84   * Abstract:
85   *    In this function, you compute the outputs of your S-function
```

```
86   *    block. Generally outputs are placed in the output vector, ssGetY(S).
87   */
88  static void mdlOutputs(SimStruct *S, int_T tid)
89  {
90      //get pointer to the block's output signal
91      const uint8_T   *u0  = (const uint8_T*) ssGetInputPortSignal(S,0);
92
93      uint8_T test[188];
94      int i;
95
96      void** pwork = ssGetPWork(S);
97
98      for (i=0;i<188;i++){
99         test[i]=u0[i];
100     }
101     fwrite(&test, 1, 188, pwork[0]);
102
103 }
104
105
106 /* Function: mdlTerminate ...
        =====================================================
107  * Abstract:
108  *    In this function, you should perform any actions that are necessary
109  *    at the termination of a simulation.  For example, if memory was
110  *    allocated in mdlStart, this is the place to free it.
111  */
112 static void mdlTerminate(SimStruct *S)
113 {
114     //close the file
115     void** pwork = ssGetPWork(S);
116       FILE *datafile;
117
118       datafile = pwork[0];
119
120       fclose(datafile);
121 }
122
123 /*==============================*
124  * Required S-function trailer *
125  *==============================*/
126
127 #ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a ...
        MEX-file? */
128 #include "simulink.c"      /* MEX-file interface mechanism */
129 #else
130 #include "cg_sfun.h"       /* Code generation registration function */
131 #endif
```

O código para o bloco CLOCK:

```
1  function CLOCK
2  global clockCount;
```

```
3
4  clockCount=clockCount+1;
5  if(clockCount>2576980377599)
6      clockCount=0;
7  end
```

O código para o bloco BUFFER:

```
1  function [hasPCR,pcrInBuffer,numCounter,numCompensation] = BUFFER(byte)
2  global inBuffer; %10 buffer for input packets
3  global Pack_n;  %refers to number of packets available in inBuffer
4  global syncing;
5  global posByte;
6  global nHeader; %number of headers used to sync
7  global hasSync;
8  global sem_en;
9  global clockCount;
10 global stamps;
11 global byteCount;
12
13 numCounter=0;
14 numCompensation=0;
15 byteCount=byteCount+1;
16
17 pcrInBuffer=sem_en;
18
19 hasPCR=false; %reads the PCR flag, start as false
20
21 if (¬syncing) %havent found any header or lost sync
22     if(byte==71) %if byte incomming is a header
23         syncing=true; %start to sync
24     end
25 else %have already found a header and is looking forward to more headers
26     if(posByte>188)
27         posByte=uint8(1);
28     end
29     if(posByte==1) %reading header
30         if(byte==71) %is it a header?
31             if(¬hasSync)%has it already sync?
32                 nHeader=nHeader+1; %then increase number of headers found
33             end
34         else
35             nHeader=uint8(0); %if not a header start to count headers ...
                   again
36             syncing=false; %try to sync again
37         end
38     end
39     if(nHeader>uint8(4))
40         hasSync=true; %with more than n headers found it has a Sync State
41     else
42         hasSync=false;
43     end
44     if(hasSync)
```

```
45          inBuffer(Pack_n+1,posByte)=byte; %store byte in input Buffer
46
47          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48          %%%%%   OPERATIONS POST READ - ADAPTATION    %%%%
49          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50          if (posByte==188) %has read and entire packet
51              %ler pid
52              pid = readPID(Pack_n+1);
53              if(pid≠hex2dec('1fff')) %check if it is not a null packet
54                                      %null packets will not be stored in ...
                                           buffer
55                  hasPCR=checkPCR(Pack_n+1);
56                  if(hasPCR) %if it has a PCR packet
57                      PCRvalue=readPCR(Pack_n+1);
58                      if(sem_en==0) %if it is the first PCR in the buffer
59                          clockCount=PCRvalue;
60                      else
61                          %time_stamps here
62                          stamps(sem_en)=PCRvalue-clockCount;
63                      end
64                      sem_en=sem_en+1;
65                          if(sem_en==1)
66                              numCounter=numCounter+1;
67                          end
68                          if(sem_en>1)
69                              numCompensation=numCompensation+1;
70                          end
71                  end
72                  if(Pack_n<40)
73                      Pack_n=Pack_n+1; %increase the number of avaiable ...
                           packets, means it is stored
74                  else
75                      %Pack_n=Pack_n;
76                      %disp('erro');
77                  end
78              end
79          end
80      end
81      posByte=posByte+1; %do only count bytes if in syncing mode
82  end
83
84  function PCRflag = checkPCR(bufferPosition)
85  global inBuffer;
86  global bufferPCR; %same dimension of inBuffer: 10
87  global NPentrada;
88  global byteCount;
89
90  adapField = bitshift(bitand(inBuffer( bufferPosition,4 ), ...
        bin2dec('00110000')),-4);
91  PCRflag=false;
92  if(((adapField == bin2dec('10')) || (adapField == bin2dec('11'))) && ...
        (inBuffer(bufferPosition,5) > 0))
93      if(bitand(inBuffer(bufferPosition,6),bin2dec('00010000')))
94          PCRflag=true;
```

```
95          bufferPCR(bufferPosition)=PCRflag;
96          NPentrada(bufferPosition)=byteCount;
97          byteCount=0;
98      end
99  end
100
101 function PCRval = readPCR(bufferPosition)
102 global inBuffer;
103
104 PCRbase = double(inBuffer(bufferPosition,7))*(2^25)...
105         + double(inBuffer(bufferPosition,8))*(2^17)...
106         + double(inBuffer(bufferPosition,9))*(2^9)...
107         + double(inBuffer(bufferPosition,10))*2 ...
108         + double(bitand(inBuffer(bufferPosition,11), ...
              uint8(bin2dec('10000000'))))/128;
109 PCRext  = double(bitand(inBuffer(bufferPosition,11), ...
      uint8(bin2dec('00000001'))))*(2^8)...
110         + double(inBuffer(bufferPosition,12));
111 PCRval  = PCRbase*300 + PCRext;
112
113 function PIDvalue = readPID(bufferPosition)
114 global inBuffer;
115
116 PIDvalue  = uint16(bitand(inBuffer(bufferPosition,2), ...
      uint8(bin2dec('00011111'))))*(2^8)...
117           + uint16(inBuffer(bufferPosition,3));
```

Códigos para o método dos Contador Controlado por Semáforos

O código para o bloco ADAPT:

```
1  function [packet,n_buffer] = ADAPT
2  global inBuffer;
3  global Pack_n;
4  global nullPacket;
5  global bufferPCR;
6  global sem_en;
7  global sem_sai;
8
9  n_buffer=Pack_n;
10
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12 %%%% ADAPTATION BLOCK - START %%%%
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 if (Pack_n<2) %this is the buffer threshold size
15     packet=nullPacket;
16 else
17     packet=inBuffer(1,1:188)';
18         inBuffer=circshift(inBuffer,-1);
```

```
19          Pack_n=Pack_n-1;
20
21          %if it has a PCR flag
22          if(bufferPCR(1)==true) %check PCRflag
23              %%comment this part to disable jitter correction
24               if(sem_sai>0)
25                  [B7 B8 B9 B10 B11 B12] = adaptation(1);
26                  packet(7:12)=[B7 B8 B9 B10 B11 B12];
27               else
28                  [B7 B8 B9 B10 B11 B12] = adaptation(0);
29                  packet(7:12)=[B7 B8 B9 B10 B11 B12];
30               end
31              %%comment this part to disable jitter correction
32              sem_sai=sem_sai+1;
33              if(sem_sai==sem_en)
34                  sem_sai=uint8(0);
35                  sem_en=uint8(0);
36              end
37          end
38          %after PCR value treated rotate pcrflag indicator buffer
39          bufferPCR(1)=false; %zero PCR on buffer
40          bufferPCR=circshift(bufferPCR,-1);
41  end
42  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43  %%%% ADAPTATION BLOCK - END %%%%
44  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45
46  function [B7 B8 B9 B10 B11 B12] = adaptation(mode)
47  global sem_sai;
48  global clockCount;
49  global stamps;
50
51  if(mode) %addition and compensation
52      pcr=stamps(sem_sai)+clockCount;
53  else %pcr clock insertion
54      pcr=clockCount;
55  end
56
57  ext = mod(pcr,300);
58  base = (pcr-ext)/300;
59
60  B7=uint8(mod(fix(base/(2^25)),2^8));
61  B8=uint8(mod(fix(base/(2^17)),2^8));
62  B9=uint8(mod(fix(base/(2^9)),2^8));
63  B10=uint8(mod(fix(base/2),2^8));
64  B11=uint8(mod(base,2)*2^7+126+fix(ext/2^8));
65  B12=uint8(mod(ext,2^8));
```

Códigos para o método da Adaptação de Taxa integrada à Correção de PCR

O código para o bloco ADAPT:

```
function [packet,n_buffer] = ADAPT
global Pack_n;
global nullPacket;
global bufferPCR;
global NPentrada;
global packetsSent;
global flagtoempty;
global ΔSE;
global n_atraso;
global Δsaida;
global NPsaida;

n_buffer=Pack_n;

TPsaida=27000000*8*188/43000000; %numero de ticks de clock por pacote ...
    (usa taxa de saida)
TPentrada=27000000*8/24127936; %numero de ticks de clock por byte na ...
    entrada

% check if buffer is overloaded
% output packets stored in buffer directly, not managing correction
if (Pack_n>38 || flagtoempty) %this is the buffer threshold size
    flagtoempty=true;
    %esvaziar conferindo se eh pcr, porem segue continuando a contagem
        if(bufferPCR(1)==true) %check PCRflag
            packet=outfrombuffer(); %enviou pacote com PCR
            %ΔSE=mod(ΔSE+TPsaida,1);
            NPsaida=1;
        else
            packet=outfrombuffer(); %enviar pacote normalmente
            %ΔSE=mod(ΔSE+TPsaida,1);
            NPsaida=NPsaida+1;
        end
    if Pack_n==1
        flagtoempty=false;
    end
%
% normal operation here, count packets between PCRs
%
else if (Pack_n>2)
        %if it is a PCR containing Packet
        if(bufferPCR(1)==true) %check PCRflag
            %deteccao de pacote com pcr
            %calcular quando deve ser enviado o pacote
            if(packetsSent==uint8(0)) %packetsSent agora eh usado para ...
                contar quantas posicoes a frente e necessario enviar o pcr
                %recuperacao do DELTAentrada
                DELTAentrada=mod(NPentrada(1)*TPentrada,1);
                %valor esperado de DELTAsaida
                Esaida=DELTAentrada-ΔSE; %esperado
                %vetor de Atrasos
                Atrasos=mod(NPsaida*TPsaida+TPsaida*[0:10],1);
                [Δsaida,n_atraso]=min( mod( 1+( ...
                    Atrasos-mod(1+Esaida,1) ),1 ) );% maior ou igual
```

```
51                  ΔSaida=Atrasos(n_atraso);
52                  %novo ΔSE
53                  ΔSE=DELTAsaida-Esaida;
54                  %inicia a contagem de atraso de pacote com PCR
55                  packetsSent=uint8(1);
56              end
57                  %enviar agora o pacote do buffer
58                  if(packetsSent==uint8(n_atraso))
59                      %envia pacote com PCR
60                      packet=outfrombuffer();
61                      %zera contagem e habilita nova configuracao de ...
                            envio de pacotes com PCR
62                      packetsSent=uint8(0);
63                      %reinicia a contagem de pacotes enviados desde o ...
                            ultimo PCR
64                      NPsaida=1;
65                  %tratamento para enviar pacote nulo
66                  else
67                      packet=nullPacket;   %se nao envia pacote nulo
68                      packetsSent=packetsSent+1;    %e incrementa ...
                            contagem de pacotes de atraso enviados
69                      NPsaida=NPsaida+1;
70                  end
71          else
72              packet=outfrombuffer(); %se nao envia pacote normalmente
73              NPsaida=NPsaida+1;
74          end
75      %if there are no packets in buffer send null packets
76      else
77          packet=nullPacket;
78          NPsaida=NPsaida+1;
79       end
80  end
81
82  function [B7 B8 B9 B10 B11 B12] = adaptation(mode)
83  global sem_sai;
84  global clockCount;
85  global stamps;
86
87  if(mode) %addition and compensation
88      pcr=stamps(sem_sai)+clockCount;
89  else %pcr clock insertion
90      pcr=clockCount;
91  end
92
93  %pcr=pcr-5000;
94
95  ext = mod(pcr,300);
96  base = (pcr-ext)/300;
97
98  B7=uint8(mod(fix(base/(2^25)),2^8));
99  B8=uint8(mod(fix(base/(2^17)),2^8));
100 B9=uint8(mod(fix(base/(2^9)),2^8));
101 B10=uint8(mod(fix(base/2),2^8));
```

```
102 B11=uint8(mod(base,2)*2^7+126+fix(ext/2^8));
103 B12=uint8(mod(ext,2^8));
104
105 function outpacket = outfrombuffer()
106 global inBuffer;
107 global Pack_n;
108 global bufferPCR;
109 global NPentrada;
110 global sem_en;
111 global sem_sai;
112
113 outpacket=inBuffer(1,1:188)';
114 %rearrange buffer
115   inBuffer=circshift(inBuffer,-1);
116   Pack_n=Pack_n-1;
117
118 %correction
119         %if it has a PCR flag
120         if(bufferPCR(1)==true) %check PCRflag
121             if(sem_sai>0)
122                 [B7 B8 B9 B10 B11 B12] = adaptation(1);
123             else
124                 [B7 B8 B9 B10 B11 B12] = adaptation(0);
125             end
126             %%comment this part to disable jitter correction
127             outpacket(7:12)=[B7 B8 B9 B10 B11 B12];
128
129             sem_sai=sem_sai+1;
130             if(sem_sai==sem_en)
131                 sem_sai=uint8(0);
132                 sem_en=uint8(0);
133             end
134         end
135         %after PCR value treated rotate pcrflag indicator buffer
136         bufferPCR(1)=false; %zero PCR on buffer
137         bufferPCR=circshift(bufferPCR,-1);
138         NPentrada(1)=0; %zero PCR on buffer
139         NPentrada=circshift(NPentrada,-1);
```

Referências Bibliográficas

[1] ABNT/NBR 15601. *Televisão digital terrestre – Sistema de transmissão.* Brasil, 1a edição, 2007.

[2] W. Fischer. *Digital Video and Audio Broadcasting Technology.* Springer: Berlin, Heidelberg, New York, 2010.

[3] X. Peng, J. Song, K. Wang. Counter-set based PCR jitter correction method for DVB-T system. In *International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007*, pp. 2940–2943, set. 2007.

[4] ISO/IEC 13818-1. *Information Technology - Generic coding of moving pictures and associated audio information - Part 1:Systems.* Geneva, Switzerland, 3a edição, 2007.

[5] ETSI EN 50083-9. *Cable distribution systems for television, sound signals and interactive multimedia signals; Part 9: Interfaces for CATV/SMATV Headends and Similar Professional Equipment for DVB/MPEG-2 Transport Streams.* 2003.

[6] C. Tryfonas and A. Varma. Timestamping schemes for MPEG-2 systems layer and their effect on receiver clock recovery. *IEEE Transactions on Multimedia*, vol.1(num.3), pp. 251–263, set. 1999.

[7] S. I. Lee, S. B. Cho, J. H. Kim, H. H. Jeon, D. G. Oh. Implementation of MPEG-2 TS remultiplexer and data transport unit for HDTV satellite broadcasting. *IEEE Transactions on Consumer Electronics*, vol.43(num.3), pp. 324–329, ago. 1997.

[8] W. Xingdong, Y. Songyu, L. Longfei. Implementation of MPEG-2 transport stream remultiplexer for DTV broadcasting. *IEEE Transactions on Consumer Electronics*, vol.48(num.2), pp. 329–334, mai. 2002.

[9] L. Longfei, Y. Songyu, W. Xingdong. Implementation of a new MPEG-2 transport stream processor for digital television broadcasting. *IEEE Transactions on Broadcasting*, vol.48(num.4), pp. 348–352, dez. 2002.

[10] Y. He, J. Zhou, Y. Zhou. Implementation of TS de-multiplexer with FPGA in DVB_IP gateway for network TV. In *IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4, nov. 2005.

[11] Y.-P. Chen, T.-N. Chien, P.-H. Cheng, S.-J. Chen. An agile and low cost FPGA implementation of MPEG-2 TS remultiplexer for CATV head-end equipment. In *10th International Symposium on Pervasive Systems, Algorithms, and Networks, ISPAN*, pp. 722–726, dez. 2009.

[12] M. Machmerth and C. Stoerte. Accumulator based PCR restamping. In *IEEE 13th International Symposium on Consumer Electronics, ISCE '09*, pp. 863–865, mai. 2009.

[13] R. Prasad, S. Hara. *Multicarrier Techniques for 4G Mobile Communications.* Artech House, 1a edição, 2003.

[14] R. Prasad. *Ofdm for Wireless Communications Systems.* Artech House, 1a edição, 2004.

[15] ATSC A/53. *ATSC Digital Television Standard: Part 1 - Digital Television System.* 2009.

[16] ETSI EN 300 744. *Digital Video Broadcasting (DVB): Framing structure, channel coding and modulation for digital terrestrial television.* 2009.

[17] ARIB STD-B31. *Transmission system for digital terrestrial television broadcasting.* 2009.

[18] ISO/IEC 14496-3. *International Standard ISO/IEC 14496 – Information technology – Coding of audio-visual objects – Part 3: Áudio.* 2005.

[19] ISO/IEC 14496-10. *International Standard ISO/IEC 14496 – Information technology – Coding of audio-visual objects – Part 3: Advanced Video Coding.* 2005.

[20] H. Benoît. *Digital Television: Satellite, Cable, Terrestrial, Iptv, Mobile TV in the DVB Framework.* Focal Press, 3a edição, 2008.

[21] ABNT/NBR 15602:1. *Televisão digital terrestre – Codificação de vídeo, áudio e multiplexação – Parte 1: Codificação de vídeo.* Brasil, 1a edição, 2007.

[22] K. Jack. *Video Demystified: A Handbook for the Digital Engineer.* Newnes, 4a edição, 2005.

[23] B. V. Veen, S. S. Haykin. *Sinais e Sistemas.* Bookman, 1a edição, 2001.

[24] ITU-T Recommendation H.264. *Advanced video coding for generic audiovisual services*. 03/2005.

[25] ABNT/NBR 15602:2. *Televisão digital terrestre – Codificação de vídeo, áudio e multiplexação – Parte 2: Codificação de áudio*. Brasil, 1a edição, 2007.

[26] ABNT/NBR 15602:3. *Televisão digital terrestre – Codificação de vídeo, áudio e multiplexação – Parte 2: Sistemas de multiplexação de sinais*. Brasil, 1a edição, 2007.

[27] C. Armbrust, J. Taylor. *DVD Demystified Third Edition*. McGraw-Hill Prof Med/Tech, 3a edição, 2005.

[28] L. Boroczky, A. Y. Ngai e E. F. Westermann. Statistical multiplexing using MPEG-2 video encoders. *IBM Journal of Research and Development*, vol.43(num.4), pp. 511–520, dez. 1999.

[29] S. B. Wicker. *Error control systems for digital communication and storage*. Prentice Hall, 1a edição, 1995.

[30] ETSI TR 101 154. *Digital Video Broadcasting (DVB): Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications*. 2000.

[31] U. Reimers. *DVB - The Family of International Standards for Digital Video Broadcasting*. Springer: Berlin, Heidelberg, New York, 2nd edição, 2004.

[32] M. Uehara. Application of MPEG-2 systems to terrestrial ISDB (ISDB-T). volume 94, pp. 261–268, jan. 2006.

[33] J. M. Rosenberg, K. R. Coombes, J. E. Osborn, G. J. Stuck, B. R. Hunt, R. L. Lipsman. *A Guide to MATLAB: For Beginners and Experienced User*. Focal Press, 2a edição, 2006.

[34] S. M. Kay. *Intuitive Probability and Random Processes using MATLAB*. Springer, 1a edição, 2006.

[35] ETSI TR 101 290. *Digital Video Broadcasting (DVB): Measurement guidelines for DVB systems*. 2001.

[36] A. Gilat. *Matlab com Aplicações em Engenharia*. Bookman, 2a edição, 2006.

[37] S. L. Netto, P. S. R. Diniz, E. A. B. da Silva. *Processamento Digital de Sinais*. Bookman, 1a edição, 2004.