UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
COORDENAÇÃO DOS CURSOS DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Estrutura Reconfigurável, Aplicada ao Processamento de Sinais Analógicos, Usando FPGA e Microcontrolador

Cleumar da Silva Moreira

Campina Grande - PB
Maio de 2000

Estrutura Reconfigurável, Aplicada ao Processamento de Sinais Analógicos, Usando FPGA e Microcontrolador

Cleumar da Silva Moreira

Dissertação de Mestrado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba – Campus II, em cumprimento às exigências para obtenção do grau de Mestre em Engenharia Elétrica

Área de Concentração: Processamento da Informação

Raimundo Carlos Silvério Freire, Dr.

Orientador

Elmar Uwe Kurt Melcher, Dr.

Orientador

Campina Grande, Paraíba, Brasil.

M835e Moreira, Cleumar da Silva
Estrutura reconfigurável, aplicada ao processamento de sinais analógicos, usando conjunto de portas programáveis em campo (Field programable gate array-FPGA)/Cleumar da Silva Moreira - Campina Grande - PB: UFPB, 2000. 103p.: il.

Inclui Bibliografia
Dissertação (mestrado) - UFPB/CCT-Engenharia Elétrica

1. Microletrônica 2. FPGA 3. Processamento de Sinais analógicos 4. Conversor A/D sigma - delta

CDU: 622.38

ESTRUTURA RECONFIGURÁVEL, APLICADA AO PROCESSAMENTO DE SINAIS ANALÓGICOS, USANDO FPGA E MICROCONTROLADOR

CLEUMAR DA SILVA MOREIRA

Dissertação Aprovada em 15.03.2000

PROF. ELMAR UWE KURT MELCHER, Dr., UFPB
Orientador

PROF. RAIMUNDO CARLOS SILVÉRIO FREIRE, Dr., UFPB
Orientador

PROF. ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFPB
Componente da Banca

PROF. GURDIP SINGH DEEP, Ph.D., UFPB
Componente da Banca

CAMPINA GRANDE - PB
Fevereiro - 2000

Dedicatória

Dedico este trabalho aos meus pais Valdemar e Creusa e aos meus irmãos Anderson e Cristiane.

Este trabalho também é dedicado à minha avó Maria Salaberga (*in memorian*).

Agradecimentos

Agradeço primeiramente a DEUS pela presença constante em minha vida, pelo meu discernimento e pela constante e prazerosa vontade de viver e ser feliz.

Agradeço aos professores Raimundo Carlos Silvério Freire e Elmar Uwe Kurt Melcher pela orientação necessária para a realização desse trabalho.

Agradeço aos professores José Sérgio da Rocha Neto e Gurdip Singh Deep pela cooperação e esclarecimentos em momentos de minha caminhada.

Agradeço aos meus amigos Márcio, Lúcio, Adolfo, Rômulo, Luís Alberto, Gustavo, pelo companheirismo e cooperação nesses anos de convivência.

Agradeço também a todos os colegas do LIEC, LEIAM e LPP, além de outros professores e funcionários da UFPB.

À CAPES que proporcionou o suporte financeiro para viabilizar a realização desse trabalho.

Resumo

Os Conjuntos de Portas Programáveis em Campo ou FPGAs (*Field Programmable Gate Arrays*), por possuírem características como reconfigurabilidade, flexibilidade alta e custos fixos (devido à fabricação, por exemplo) menores que os ASICs (Application Specific Integrated Circuits), são muito utilizados atualmente nas áreas mais variadas da Engenharia Elétrica. Uma utilização bem interessante desses dispositivos é como um co-processador junto a um processador de uso geral ou não flexível, com o objetivo de oferecer um aumento de flexibilidade, capacidade de processamento ou desempenho em relação às estruturas convencionais. As estruturas reconfiguráveis resultantes desse co-processamento podem ser implementadas com componentes discretos ou podem ser integradas em um único *chip*. Uma estrutura reconfigurável usando componentes discretos (no caso, um FPGA e um microcontrolador) e destinada ao processamento de sinais analógicos é apresentada nessa dissertação. Com os componentes dessa estrutura foi projetada uma placa que pode, dependendo da aplicação, substituir implementações com componentes analógicos discretos, ou integradas quando o interesse for a prototipagem rápida de sinais. Além disso, foi elaborado um fluxo de projeto visando a realização de aplicações nessa placa.

Abstract

Due to characteristics like fast prototyping, reconfigurability and flexibility, the application fields of FPGAs, actually, is increasing. One of these applications is a co-processor to a general-purpose processor (a microcontroller or a PC) that permits to obtain, in comparison with conventional structures, an increase of flexibility, performance or processing capacity. Following that, this dissertation presents a reconfigurable structure using a FPGA, a microcontroller, A/D and D/A converters and interface elements for analog signal processing. Additionally, a structure-based board was designed to support fast prototyping and a design flow was elaborated for applications realization.

Sumário

Dedicatória iv

Agradecimentos v

Resumo vi

Abstract vii

Sumário viii

Lista de Abreviaturas xi

Lista de Figuras xiii

Capítulo 1: Introdução 1

1.1. Motivações 3

1.2. Estrutura da Dissertação 4

Capítulo 2: Conjunto de Portas Programáveis em Campo (*Field Programmable Gate Arrays* – FPGAs) 6

2.1. *Field Programmable Gate Arrays* – FPGAs 10

2.1.1. Definições e Características Gerais 10

2.1.2. Chaves Programáveis 11

2.1.2.1. Antifuses 11

2.1.2.2. Tecnologia de Porta Flutuante 12

2.1.2.3. Células de Memória SRAM (*Static Random Access Memory*) 13

2.1.3. Blocos Lógicos Configuráveis (CLBs) 14

2.1.3.1. Arquiteturas de Granulação Fina 15

2.1.3.2. Arquiteturas de Granulação Grossa 16

2.1.4. FPGAs Com Memória Interna 17

2.1.5. FPGAs Reconfiguráveis 18

2.1.6. Fluxo de Projeto em um FPGA 19

2.1.7. Desvantagens e Limitações de um FPGA 21

2.1.8. Aplicações de FPGAs 22

2.1.9. Conclusões e Tendências Futuras 24

Capítulo 3: Estrutura Reconfigurável Aplicada ao Processamento de Sinais Analógicos 25

3.1. Componentes da Estrutura Reconfigurável 25

3.1.1. FPGA 25

3.1.1.1. Critério de Encapsulamento 26

3.1.1.2. Critério de Capacidade de Entrada e Saída 26

3.1.1.3. Escolha do FPGA 26

3.1.1.4. Esquema de Configuração do FPGA 27

3.1.2. Microcontrolador 29

3.1.3. Conversores A/D e D/A 29

3.1.4. Memórias de Dados e Programas 30

3.2. Detalhes da Placa Reconfigurável Proposta 31

3.3. Fluxograma para a Implementação de Aplicações 32

3.4. Conclusão 33

Capítulo 4: Testes Experimentais 34

Implementação de Filtros Digitais

4.1. Esquema de Implementação dos Filtros Digitais FIR e IIR 35

4.2. Filtros FIR 36

4.2.1. Descrição no FPGA 36

4.3. Filtros IIR 39

4.3.1. Descrição no FPGA 39

4.4. Conclusão 41

Conversor A/D Σ-Δ Monobit de 1ª Ordem

4.5. Estrutura de Implementação Utilizada 42

4.6. Modulador 43

4.6.1. Descrição dos Componentes que Compões o Modulador 43

4.6.1.1. Integrador 43

4.6.1.2. Quantizador de 1 Bit 44

4.6.1.3. Sobreamostragem do Sinal Resultante do Quantizador de 1 bit 44

4.6.1.4. Circuito que converte o Sinal TTL na saída do Flip-Flop para um Sinal Quadrado Simétrico 45

4.6.2. Resultados Obtidos 46

4.6.1.1. Sinal de Entrada com Nível Médio Nulo 46

4.6.1.2. Sinal de Entrada com Nível Médio Positivo 47
4.6.1.1. Sinal de Entrada com Nível Médio Negativo 48
4.7. Bloco de Processamento Digital 49
4.7.1. Descrição no FPGA 49
4.8. Resultados Obtidos 50
4.9. Conclusão 52

Capítulo 5: Conclusões e Trabalhos Futuros 53

5.1. Trabalhos Futuros 54

Apêndice A: Conversores A/D Sigma-Delta 55

A.1. Conversores A/D na Taxa de Nyquist ou Convencionais 56
A.1.1. Definições e Características Gerais 56
A.1.2. Tipos de Conversores A/D Convencionais 60
A.2. Conversores A/D Sobreamostrados 61
A.2.1. Definições e Características Gerais 61
A.2.2. Estruturas de Conversores Sobreamostrados 64
A.2.2.1. Predição 66
A.2.2.2. *Noise-Shaping* 67
A.3. Conversores A/D Sigma-Delta 68
A.3.1. Conversores A/D Sigma-Delta de 1ª Ordem 70
A.3.1.1. Modulador A/D Sigma-Delta de 1ª Ordem 70
A.3.1.2. Bloco de Processamento Digital 74
A.4. Conclusão 75

Apêndice B1: Esquemático da Placa Reconfigurável 77

Apêndice B2: Leiaute da Placa Reconfigurável (Camada Superior – *Top Layer*) 79

Apêndice B3: Leiaute da Placa Reconfigurável (Camada Inferior – *Bottom Layer*) 81

Apêndice C: Programas em Verilog HDL.....83

Bibliografia 88

Lista de Abreviaturas

ASIC	:	*Application Specific Integrated Circuit*
FPGA	:	*Field Programmable Gate Array*
SOC	:	*System-on-Chip*
FPD	:	*Field Programmable Device*
PROM	:	*Programmable Read Only Memory*
PLA	:	*Programmable Logic Array*
MMI	:	*Monolithic Memories*
AMD	:	*Advanced Micro Devices*
PAL	:	*Programmable Array Logic*
PLD	:	*Programmable Logic Device*
SPLD	:	*Simple Programmable Logic Device*
CAD	:	*Computer-Aided Design*
CPLD	:	*Complex Programmable Logic Device*
CLB	:	*Configurable Logic Block*
PLICE	:	*Programmable Logic Interconnect Circuit Element*
ONO	:	*Oxygen-Nytrogen-Oxygen*
EPROM	:	*Erasable* PROM
EEPROM	:	*Electrically* EPROM
HDL	:	*Hardware Description Language*
VHDL	:	*Very High Speed Integrated Circuits* HDL
RTL	:	*Register Transfer Level*
FIR	:	*Finite Impulse Response*
IIR	:	*Infinite Impulse Response*
FFT	:	*Fast Fourier Transform*
FPAA	:	*Field Programmable Analog Array*
FPMA	:	*Field Programmable Mixed Array*
FIPSOC	:	*Field Programmable System On-a-Chip*
HPFA	:	*Hybrid Field-Programmable Architecture*
UMC	:	*United Microelectronics Corporation*
ULA	:	Unidade Lógica Aritmética

THD	:	*Total Harmonic Distortion*
VLSI	:	*Very Large Scale Integration*
LSB	:	*Least Significant Bit*
DPCM	:	*Delta Pulse Code Modulation*
PDM	:	*Pulse Density Modulation*
PLCC	:	*Plastic J-Lead Chip Carrier*
PQFP	:	*Plastic Quad Flat Pack*
RQFP	:	*Power Quad Flat Pack*
sof	:	*SRAM Object File*
pof	:	*Programming Object File*
sbf	:	*Serial Bitstream File*
hex	:	*Hexadecimal (Intel-format) File*
ttf	:	*Tabular Text File*
rbf	:	*Raw Binary File*

Lista de Figuras

Figura 1.1 – Esquemático da Placa Montada em [9]

Figura 2.1 – Estrutura de uma memória PROM, apresentando seus planos não-programáveis AND e programável OR

Figura 2.2 - Estrutura de um PAL registrado que pode realizar funções combinacionais e sequenciais

Figura 2.3 - Estrutura de um CPLD, mostrando o mecanismo de roteamento global

Figura 2.4 - Arquitetura de um FPGA

Figura 2.5 - Estrutura de um antifuse usado no FPGA da empresa ACTEL

Figura 2.6 - Células de memória SRAM no controle de: a) transistores de passagem; b)multiplexadores

Figura 2.7 - Estrutura do CLB da Crosspoint

Figura 2.8 - Exemplo de um CLB com arquitetura de granulação grossa

Figura 2.9 Fluxo de Projeto em um FPGA

Figura 2.10 - Esquemático do FIPSOC, mostrando seus blocos internos e suas conexões com o mundo externo

Figura 3.1 - Diagrama de blocos da estrutura reconfigurável proposta

Figura 3.2 - Fluxograma a ser usado na implementação de aplicações usando a estrutura reconfigurável proposta

Figura 4.1 - Diagrama de blocos utilizado na implementação dos filtros digitais passa-baixas

Figura 4.2 - Estrutura de um filtro FIR de 8 *taps*, no qual a variável de entrada possui N bits e a variável de saída M bits

Figura 4.3 - Estruturas de filtros FIR: a) de 16 *taps* e b) 32 *taps*, baseados na estrutura de um filtro FIR de 8 *taps*

Figura 4.4 - Parte do código Verilog que realiza os elementos de atraso através de registradores de deslocamento baseados na freqüência do relógio associado do sinal de entrada

Figura 4.5 - Parte do código Verilog que implementa os multiplicadores e somadores

Figura 4.6 - Arquitetura na forma direta de um filtro IIR

Figura 4.7 - Parte do código Verilog que descreve os atrasos do sinal de entrada e de saída, de acordo com a freqüência de amostragem

Figura 4.8 - Parte do código Verilog que realiza as multiplicações dos coeficientes pela entrada e pelos termos atrasados da entrada (*n01* e *n11*) e saída (*d01* e *d11*)

Figura 4.9 - Diagrama de blocos em que são mostrados os componentes usados no teste experimental com conversor A/D Σ-Δ de 1ª ordem

Figura 4.10 - Integrador usado no modulador A/D Sigma-Delta

Figura 4.11 - Quantizador de 1 bit

Figura 4.12 - Esquema do circuito que sobreamostra o sinal na saída do quantizador de 1 bit

Figura 4.13 - Esquema do circuito que codifica a saída TTL Q do flip-flop para um sinal quadrado simétrico variando de $+V_{ref}$ a $-V_{ref}$

Figura 4.14 - Curvas do sinal entrada (nível médio nulo) e na saída do quantizador de 1 bit

Figura 4.15 - Curvas dos sinais de entrada e na saída do integrador

Figura 4.16 - Curvas dos sinais de entrada (nível médio positivo) e na saída do quantizador de 1 bit

Figura 4.17 - Curvas dos sinais de entrada e na saída do integrador

Figura 4.18 - Curvas dos sinais de entrada (nível médio negativo) e na saída do quantizador de 1 bit

Figura 4.19 - Curvas dos sinais de entrada e na saída do integrador

Figura 4.20 - Parte do código Verilog do decimador que implementa o *down-sampling*

Figura 4.21 - Filtro de 2ª ordem Sallen-Key colocado na saída do conversor D/A

Figura 4.22 - Curvas dos sinais de entrada (1 Vpp e 20 Hz) e de saída do conversor

Figura 4.23 - Curvas dos sinais de entrada (1 Vpp e 50 Hz) e de saída do conversor

Figura A.1 - Espectro de um sinal amostrado na taxa de Nyquist e a resposta do filtro *anti-aliasing* requerido

Figura A.2 - Diagrama de blocos de um típico conversor A/D convencional típico

Figura A.3 - Estrutura de um quantizador típico usado em conversores A/D, junto ao código binário gerado que usa complemento 2

Figura A.4 - Modelo de um quantizador, levando em consideração o erro de quantização

Figura A.5 - Densidade de probabilidade do ruído de quantização

Figura A.6 - Espectro das amostras de um sinal de entrada com uma freqüência duas vezes superior à freqüência de Nyquist

Figura A.7 - Densidades espectrais de potência do ruído de quantização para conversores sobreamostrados e na taxa de Nyquist

Figura A.8 - Diagrama de blocos de um conversor A/D sobreamostrado, exemplicado pelo estágio de codificação de um sistema PCM sobremaostrado

Figura A.9 - Diagrama de blocos que mostra o modulador de um conversor A/D sobreamostrado Preditivo, na qual é empregada a realimentação do sinal de saída através de um integrador

Figura A.10 - Característica de *Noise-Shaping*

Figura A.11 - Diagrama de blocos de um conversor sobreamostrado que produz a característica de *Noise-Shaping* mostrada na Figura A.10

Figura A.12 - Gráfico que mostra o forte compromisso entre largura de banda e resolução para os conversores A/D convencionais e Sigma-Delta

Figura A.13 - Conversor A/D sigma-delta passa-faixas de 8^a ordem [39]

Figura A.14 - Estrutura de um modulador A/D Sigma-Delta de 1^a ordem: a) modulador realizado em tempo contínuo; b) modulador realizado em tempo discreto

Figura A.15 - Diagrama de blocos de um modulador A/D Sigma-Delta de 2^a ordem

Figura A.16 - Blocos componentes de um típico decimador digital, com razão de decimação N

Lista de Tabelas

Tabela 2.1 - Comparação das Chaves Programáveis

Tabela 3.1 - Esquemas de configuração e seus bits de seleção

Tabela 4.1 - Especificações utilizadas para o cálculo dos coeficientes dos filtros FIR

Tabela 4.2 - Resultados obtidos através da compilação e análise de temporização dos filtros FIR

Tabela 4.3 - Resultados obtidos através da compilação e análise de temporização do filtro IIR de 2ª ordem

Tabela A.1 – Exemplos comerciais de conversores A/D Sigma-Delta

Capítulo 1

Introdução

Desde que Jack Kilby da Texas Instruments propôs, há mais de 40 anos, o primeiro circuito integrado, a microeletrônica desenvolve-se muito rapidamente com a redução do tamanho dos transistores, pesquisa de novos materiais, novos dispositivos e novas ferramentas CAD (*Computer-Aided Design*) de projeto e simulação do circuito e de seu leiaute. Esse desenvolvimento possibilitou a ampliação da gama de aplicações da microeletrônica que pode variar, atualmente, desde a montagem de microestruturas mecânicas (micro-turbinas ou micro-acelerômetros, por exemplo) a sistemas digitais e/ou analógicos completos contidos em um único *chip*, os chamados *System-on-Chip (SOC)* [1, 2].

Quando se fala em microprocessadores duas características, surgidas por causa desse avanço da microeletrônica, destacam-se [3]: o paralelismo e a reconfigurabilidade em *hardware*. O paralelismo significa que vários processadores dividem uma mesma tarefa, possibilitando, dessa forma, um aumento de velocidade que é proporcional ao número de processadores. Já a reconfigurabilidade em *hardware* traduz-se na interconexão rápida e elétrica de milhões de portas em poucos milisegundos, no tempo e lugar de uso com o objetivo de desenvolver as funções requeridas pelo usuário.

Vale lembrar que a reusabilidade ou reconfigurabilidade em *hardware* foi um fator utilizado nos primeiros computadores e que proporcionou ao computador ser uma ferramenta tão útil e revolucionária como é nos dias atuais [3]. Ironicamente, hoje não se podem modificar quaisquer características da arquitetura de qualquer microprocessador. Entretanto, com o uso de dispositivos reconfiguráveis em *hardware*, o programador é habilitado a criar o seu próprio microprocessador ou outro *hardware* especializado adequado para uma certa aplicação.

Assim, o projeto de processadores poderá ser modificado com o uso conjunto de dispositivos reconfiguráveis, que proporcionam um aumento de desempenho e flexibilidade por um custo mais baixo [3]. Essas arquiteturas de processadores que incorporam ou aliam a reconfigurabilidade em suas estruturas pertencem a um tipo de computação definido atualmente como Computação Reconfigurável [4].

Os Conjuntos de Portas Programáveis em Campo ou FPGAs[1] (*Field*-Programmable *Gate Arrays*) são os principais representantes dos circuitos reconfiguráveis em *hardware*[2]. Os FPGAs podem aliar-se a um microprocessador para a realização de funções digitais unicamente [5, 6, 7, 8, 9].

É evidente que não só os FPGAs existem para a realização de funções digitais, havendo também os ASICs (*Application Specific Integrated Circuits*) e processadores DSPs (*Digital Signal Processors*), por exemplo. Contudo, a característica de reconfiguração rápida dos FPGAs os tornam importantes e mais eficientes para aplicações que requeiram prototipagem rápida e que não sejam muito específicas e com volume de produção alto.

Além disso, os FPGAs, em relação aos ASICs, não incluem boa parte dos custos fixos, acarretando em um ônus menor quando se pensa em escala de produção reduzida. Para certas aplicações, os FPGAs são bem mais eficientes que um processador DSPs.

Uma aplicação interessante e em voga atualmente utiliza a união entre um FPGA e um microprocessador para a realização de sistemas que se baseiam na técnica de **Co-Projeto *Hardware/Software***, em que o FPGA (elemento de *hardware* reconfigurável) e o microprocessador (elemento de *software*) interagem entre si com o objetivo de realizar uma dada aplicação de uma forma mais eficiente que isoladamente [10, 11, 12].

A reconfigurabilidade em *hardware* não abrange apenas o domínio digital e se alastra atualmente pelos domínios analógico (FPAAs – *Field Programmable Analog Arrays*) e misto (FPMAs – *Field Programmable Mixed Arrays*). Entretanto, isso não é ainda bem aceito pelo mercado consumidor, embora circuitos integrados dessa natureza já estejam disponíveis para comercialização.

Uma idéia que cresce hoje em dia é o projeto de dispositivos reconfiguráveis que possuam reconfigurabilidade dinâmica (*run-time reconfigurability*), o que aumentaria a flexibilidade e facilitaria a vida do usuário-fim [3].

---

[1] O nome FPGA será usado durante o restante do texto por motivos de comodidade e pelo fato de ser mais utilizado.

[2] Os FPGAs são dispositivos que possuem uma reconfiguração do tipo estática, ou seja, a reconfiguração não é realizada on-line.

1.1. Motivações

Uma estrutura proposta recentemente no Departamento de Engenharia Elétrica da Universidade Federal da Paraíba [9], que pode ser vista como um exemplo da utilização de dispositivos reconfiguráveis e processadores de uso geral, incluiu em uma placa, conectada ao barramento ISA de 16 bits de um PC, dois FPGAs e circuitos adicionais para interface com o PC. Essa estrutura é mostrada na Figura 1.1 e foi objetivada para uso no processamento digital de imagens.

Essa implementação, o desejo de prosseguir o trabalho com FPGAs e, é claro, a reconfigurabilidade presente nos FPGAs foram as principais motivações para realização dessa dissertação, que apresenta uma estrutura reconfigurável aplicada à prototipagem de sinais analógicos. Usa, ao invés de um PC, um microcontrolador aliado a um FPGA e incorpora, para atender a aplicação-fim, conversores A/D e D/A relativamente rápidos.

Vale ressaltar que a reconfigurabilidade apresentada pelos FPGAs é mais eficiente quanto menores forem a especificidade desejada e escala de produção e quanto maior for a necessidade de prototipagem rápida. Se a necessidade for atender uma aplicação específica, com alta escala de produção, a opção mais interessante é projetar um ASIC. Já para a implementação de certas funções digitais, como filtros digitais ou algoritmos FFT, a opção mais adequada são os processadores DSPs, que possuem uma estrutura dedicada para a realização dessas funções.

Como a gama de aplicações destinada à essa estrutura é não-específica e por se desejar um protótipo para testes, a melhor opção foi a escolha de um FPGA.

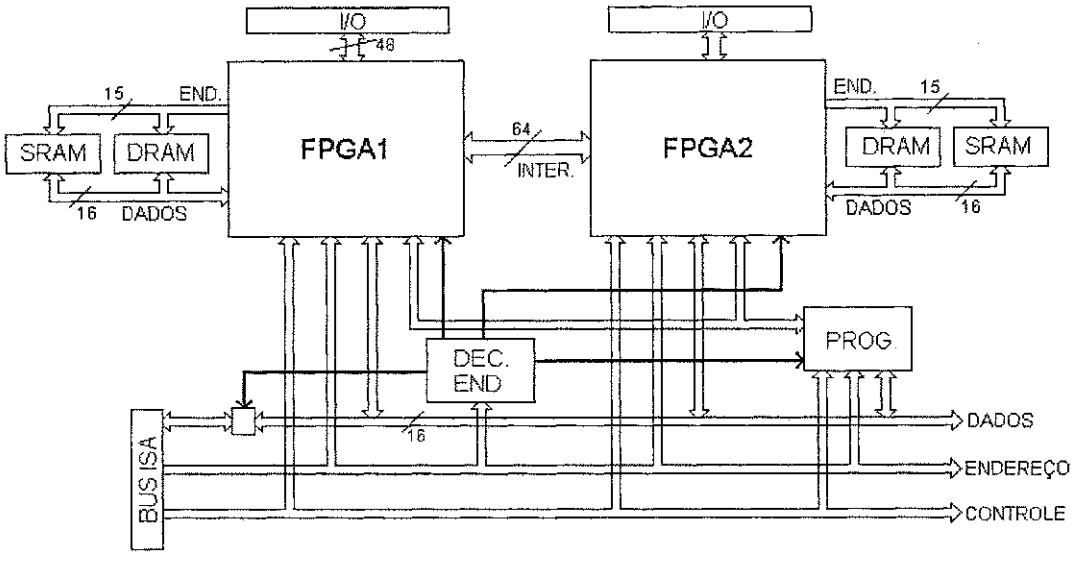


Figura 1.1 - Esquemático da placa montada em [9]

1.2. Estrutura da Dissertação

Nessa dissertação é apresentada basicamente a estrutura reconfigurável proposta e uma conceituação de FPGAs. Além disso, apresenta-se o projeto de uma placa que contém os componentes da estrutura proposta. Essa placa poderá, dependendo da aplicação, substituir implementações usando componentes discretos no processamento de sinais analógicos, desde que o interesse seja a prototipagem e/ou um tempo curto de desenvolvimento do projeto.

Dois testes experimentais foram realizados com a placa montada em [9] para validar a estrutura proposta e a sua gama de aplicações:

- No primeiro foram implementados filtros digitais FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*).
- No segundo teste foi implementado um conversor A/D Sigma-Delta monobit de 1ª ordem, em que o estágio de modulação ou analógico foi implementado com componentes discretos e o estágio de decimação ou digital no FPGA.

Visando atender ao tema definido para esse trabalho de dissertação, os capítulos restantes destacarão aspectos teóricos, práticos e do projeto da placa:

- **Capítulo 2 – FPGAs (*Field-Programmable Gate Arrays*)**: nesse capítulo será apresentado o estado da arte dos FPGAs, analisando suas características principais, metodologia de projeto e aplicações.

- **Capítulo 3 – Estrutura reconfigurável proposta**: nesse capítulo serão apresentados o projeto da placa de prototipagem e um fluxo de projeto elaborado para a realização de aplicações.
- **Capítulo 4 – Experiências:** aqui serão mostrados aspectos do projeto e de escolhas de arquiteturas dos filtros digitais e do conversor A/D Sigma-Delta monobit de 1ª ordem. Resultados práticos serão também apresentados.
- **Capítulo 5 – Conclusões e Trabalhos Futuros**
- **Apêndice A – Conversores A/D Sigma-Delta**: nesse apêndice é apresentada a teoria dos conversores A/D Sigma-Delta e, de um modo mais restrito, a teoria dos conversores A/D Sigma-Delta de 1ª ordem, ressaltando suas realizações tanto ao nível do modulador, como do decimador.
- **Apêndice B: Esquemático e Leiaute da Placa Reconfigurável**: Nesse apêndice é apresentado o esquemático e o leiaute da placa reconfigurável projetada.
- **Apêndice C – Programas em Verilog HDL:** nesse capítulo serão listados os códigos dos programas em Verilog HDL utilizados nos testes experimentais.

Capítulo 2

FPGAs- *Field Programmable Gate Arrays*

Um dos ramos da microeletrônica que mais se desenvolve atualmente é o dos dispositivos programáveis que realizam funções digitais. São circuitos que ao contrário dos ASICs (*Application Specific Integrated Circuits*), fabricados através de tecnologias como *Gate Arrays* ou *Standard Cells*, permitem ao próprio usuário configurar o dispositivo no campo de aplicação e são, por isso, denominados dispositivos programados em campo ou, abreviadamente, FPDs (*Field Programmable Devices*). Além disso, esses dispositivos já substituíram as realizações discretas de circuitos lógicos baseados nas tecnologias SSI (*Small Scale Integration*) e MSI (*Medium Scale Integration*) [10, 11].

Os precursores desses dispositivos foram as memórias programáveis PROM, *Programmable Read Only Memory*. A estrutura dessas memórias é composta por dois níveis de portas lógicas, como mostrado na Figura 2.1. O primeiro nível corresponde a um plano não-programável de portas AND (correspondendo ao decodificador) que gera todos os mintermos possíveis, conforme um certo número de entradas. O segundo nível é um plano programável de portas OR (correspondendo à tabela de endereços) que implementa a soma dos mintermos. Diz-se que uma PROM é programável, pois somente os pontos especificados na Figura 2.1 como pontos programáveis são fundidos (utilizam-se fusíveis para a sua programação). Vê-se por essa figura que apenas o plano OR é programável.

Um problema das memórias PROM para a realização de funções lógicas, pois não utilizam quaisquer mecanismos de minimização e, dessa forma, acarretam em "desperdício de lógica" [12].

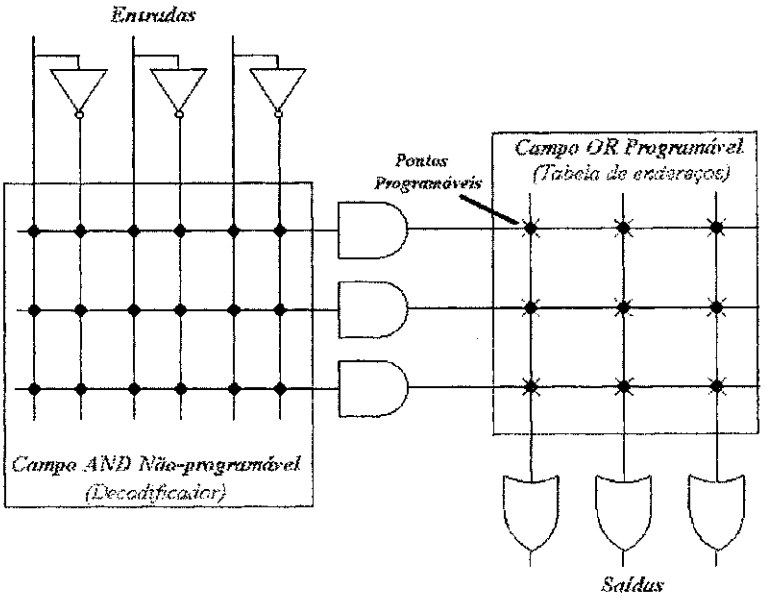


Figura 2.1 – Estrutura de uma memória PROM apresentando seus planos não-programáveis AND e programável OR

Uma solução para esse problema foi proposta no início da década de 70 pela empresa Philips®, através de um dispositivo que continha os planos AND e OR (Figura 2.1) programáveis. Com o plano AND programável, a minimização dos mintermos era possível.

A programação ou a configuração desses dispositivos era realizada também por fusíveis. Tais dispositivos foram denominados de PLAs (*Programmable Logic Arrays*). Uma grande vantagem desses dispositivos era a predição fácil dos atrasos de portas devido a sua estrutura. Contudo, apresentavam desempenho baixo e eram de fabricação difícil [13].

Uma alternativa encontrada pela indústria (MMI - *Monolithic Memories*, fundida posteriormente com a AMD – *Advanced Micro Devices*), para resolver esse problema, foi projetar um dispositivo que não contivesse o plano OR não-programável, pois para boa parte das aplicações essa programabilidade não era justificada.

Os dispositivos assim projetados foram chamados de PALs (*Programmable Array Logic*) ou PLDs (*Programmable Logic Devices*) e se constituíram na base da maioria dos FPDs hoje existentes. Os PALs ou PLDs (*Programmable Logic Devices*), assim como os PLAs, realizam funções seqüenciais quando registradores são inseridos na saída do plano OR não-programável, como está mostrado simplificadamente na Figura 2.2.

Os pontos programáveis mostrados na Figura 2.2 podem ser fusíveis, células de memorai EPROM (Erasable PROM) ou células de memória EEPROM (Electrically EPROM) [10]:

Os PALs juntamente com os PLAs formam os SPLDs (*Simple Programmable Logic Devices*), cujas características principais são:

- Esses dispositivos são programados mudando as características do ponto programável ou elemento de chaveamento;
- Por possuírem arquiteturas fixas, os atrasos de propagação são constantes e determinados antes mesmo que as chaves programáveis sejam programadas para a realização da função lógica desejada. Isso reduz os preços desses dispositivos [13];
- As ferramentas CAD (*Computer-Aided Design*) necessárias ao projeto são simples e possuem preços razoavelmente baixos.

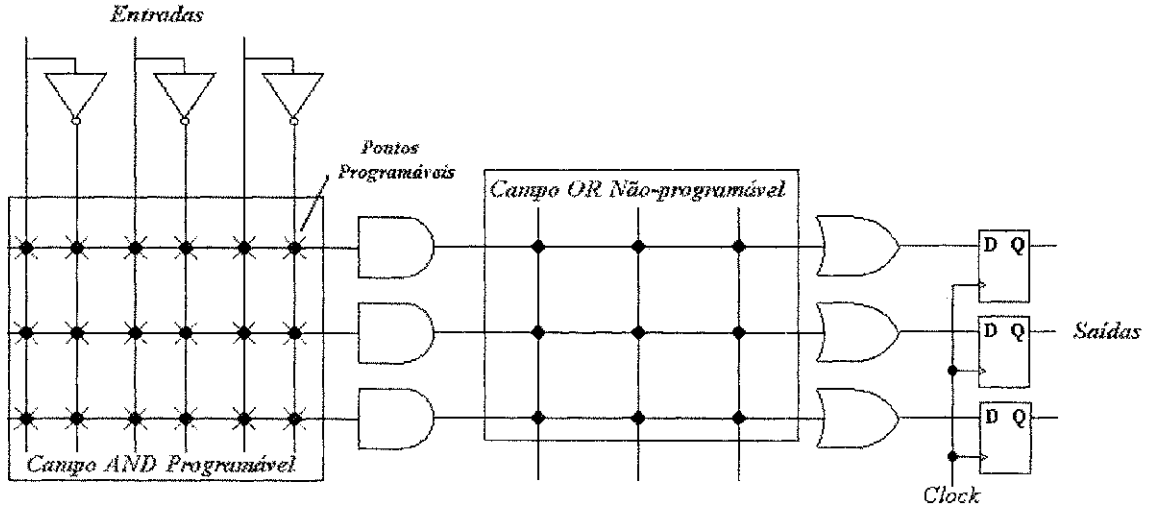


Figura 2.2 – Estrutura de um PAL que pode realizar funções combinacionais e seqüenciais

Apesar de apresentarem algumas características interessantes, os SPLDs possuem limitações de arquitetura e aplicações: são impossibilitados para a realização de projetos digitais complexos e eficientes, devido às suas arquiteturas fixas e o número de termos produtos ser restrito, assim como o número de registradores, entradas e saídas.

Para atender a demanda de dispositivos programáveis para a realização de funções complexas e com arquitetura sem muitas limitações de densidade de portas, os CPLDs (*Complex Programmable Logic Devices*) foram propostos. Em suma, correspondem a FPDs que

englobam num único encapsulamento vários PLDs (com capacidade máxima de 50 PLDs [13]), sendo que as ligações entre esses são realizadas através de um roteamento global programável, como está ilustrado na Figura 2.3. Dessa forma, um CPLD apresenta dois tipos de programabilidade em sua estrutura, uma devido aos PLDs presentes em sua estrutura e outra devido ao roteamento global.

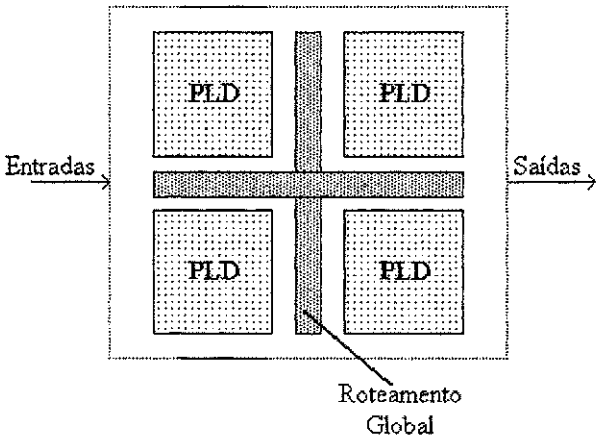


Figura 2.3 – Estrutura de um CPLD mostrando o mecanismo de roteamento global

Em relação aos SPLDs, os CPLDs possuem ferramentas CAD, usadas na programação das chaves internas e do roteamento global, mais caras e complexas e uma dificuldade maior para determinar os atrasos de propagação. Contudo, a utilização de SPLDs ou CPLDs deve está inerentemente ligado à aplicação-fim, pois às vezes pode ser que um simples circuito integrado SSI ou MSI resolva o problema. Uma outra dificuldade encontrada nos CPLDs está na realização de funções lógicas muito complexas, devido à carência de componentes lógicos, entradas e saídas.

Uma solução para esse problema foi estabelecida quando os FPGAs (*Field Programmable Gate Arrays*) foram propostos. As definições e as características principais desses dispositivos são mostradas a seguir.

2.1. FPGAs – *Field Programmable Gate Arrays*

2.1.1. Definições e Características Gerais

Em 1985, a empresa Xilinx propôs um FPD denominado de FPGA e que, ao invés de possuir uma arquitetura fixa como os SPLDs ou os CPLDs, apresentava uma arquitetura mais flexível baseada em aglomerações de portas lógicas de baixo *fan-in* [13].

Chan e Mourad [14] definem os FPGAs como dispositivos baseados na programabilidade dos PLDs (SPLDs e CPLDs) e na arquitetura das *Gate Arrays*. Essa última característica deve-se à sua arquitetura de linhas e colunas que é semelhante a uma *Gate Array*.

Os FPGAs dividem as funções lógicas a serem implementadas em blocos lógicos pequenos, denominados de CLBs (*Configurable Logic Blocks*) que contêm, comumente, *flip-flops*, multiplexadores e tabelas de busca (*Look-up Tables*). Com esses componentes internos, um CLB é apto a implementar uma variedade ampla de funções lógicas combinacionais e/ou seqüenciais.

O FPGA, além do CLB, é composto dos seguintes componentes: mecanismo de roteamento ou interconexão interna entre os CLBs e entre os CLBs e blocos de entrada e saída (localizados na periferia do circuito integrado). Esse mecanismo de roteamento é composto de muitas chaves normalmente [13]. Dessa forma, vê-se que, diferentemente dos PLDs, os FPGAs possuem interconexão programável. A Figura 2.4 apresenta a arquitetura de um FPGA.

Os itens a seguir descreverão os componentes principais, o fluxo de projeto e algumas áreas de aplicações dos FPGAs.

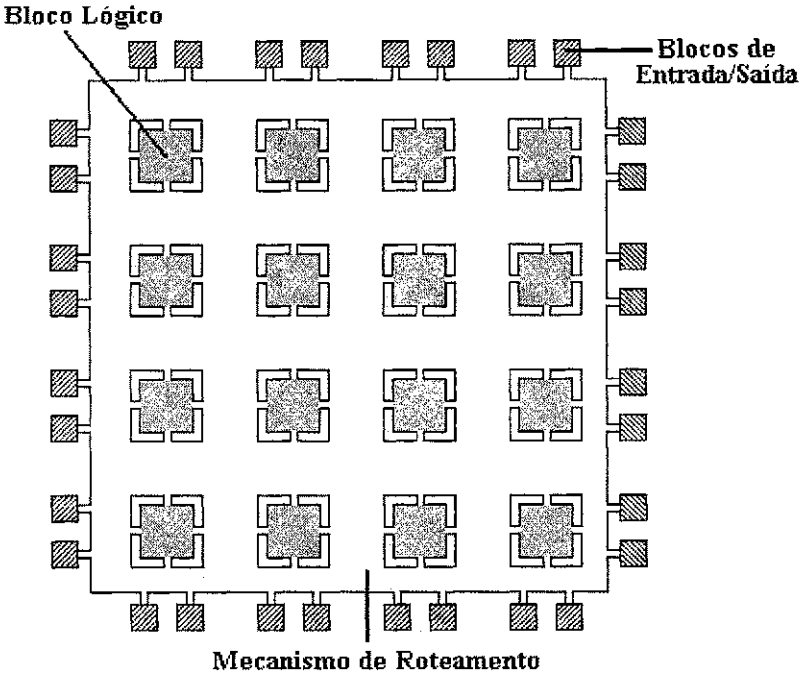


Figura 2.4 – Arquitetura de um FPGA

2.1.2. Chaves Programáveis

A programação ou a configuração interna dos FPGAs é realizada pelo mecanismo de roteamento que é composto de chaves programáveis, que podem ser de dois tipos basicamente: permanentes ou não permanentes. O primeiro tipo é aquele em que a programação só pode ser realizada uma única vez (*antifuses*). Já o segundo tipo é aquele em que a programação no FPGA pode ser repetida muitas vezes (células de memória EPROM – *Erasable* PROM, EEPROM – *Electrically* EPROM – ou SRAM – *Static Random Access Memory*).

Essas chaves programáveis são dispositivos CMOS, diferentemente dos dispositivos bipolares (fusíveis) usados nos PLDs.

A seguir essas chaves programáveis são mais bem caracterizadas.

2.1.2.1. *Antifuses*

Um *Antifuse* é um dispositivo de dois terminais que, no estado não-programado, apresenta uma resistência elevada entre seus terminais. Para sua programação, é necessária a aplicação de uma tensão de *15 a 20 V*, conforme o *antifuse*, entre os seus terminais. A aplicação dessa tensão leva o *antifuse* à fusão e cria uma ligação permanente e de baixa resistência, não

havendo a possibilidade de reprogramação da referida chave. Além da dessa tensão, há uma corrente associada de 5 mA ou superior que normalmente requer uma circuitaria externa para isolação (*buffers* de corrente formado por transistores).

Um *antifuse* é composto de três camadas, sendo duas condutoras e uma isolante, na qual a camada isolante fica entre as duas camadas condutoras. Quando é aplicada a tensão de programação entre essas camadas, a rigidez dielétrica da camada isolante é rompida e o dispositivo conduz.

Um exemplo comercial de um *antifuse*, fabricado pela ACTEL, é mostrado na Figura 2.5. Esse *antifuse* é denominado PLICE (*Programmable Logic Interconnect Circuit Element*) e a camada isolante é feita de um composto chamado ONO (*Oxygen-Nitrogen-Oxygen*). Outros *antifuses* usam silício amorfo entre as camadas condutoras.

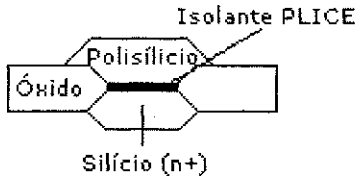


Figura 2.5 – Estrutura de um antifuse usado no FPGA da empresa ACTEL

2.1.2.2. Tecnologia de Porta Flutuante

As chaves programáveis baseadas na tecnologia de porta flutuante correspondem a células de memória EPROM e EEPROM. Como vantagem em relação aos *antifuses* está a reprogramabilidade, ou seja, a capacidade de ser programada mais de uma vez. No caso das EPROMs, a programação é eletrônica e o apagamento de seu conteúdo é realizado com a exposição à luz ultravioleta. Já as EEPROMs são apagadas e programadas eletronicamente.

As desvantagens dessas chaves programáveis são: há uma quantidade de passos adicionais de integração sobre um processo ordinário CMOS, o que acarreta num tempo de projeto maior e com maior ônus também e uma resistência alta (2 a 4 kΩ), no estado ligado, em relação às outras chaves programáveis [13].

2.1.2.3. Células de Memória SRAM (*Static Random Access Memory*)

Essas chaves programáveis se constituem em células de memória SRAM de 1 bit usadas para controlar transistores de passagem ou multiplexadores, como está mostrado na Figura 2.6.

O transistor de passagem, mostrado na Figura 2.6.a, estará em condução ou não, conforme o valor lógico armazenado na célula de memória SRAM. Se for armazenado 1 lógico, o transistor conduz. Caso seja armazenado 0 lógico, o transistor não conduz.

Já no multiplexador (Figura 2.6.b), o estado das células de memória SRAM conectadas definem os bits de endereçamento.

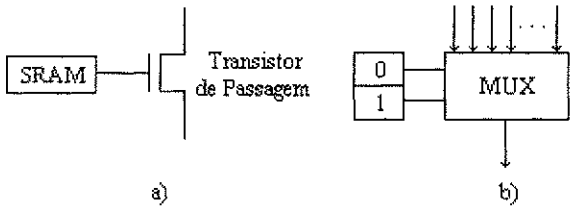


Figura 2.6 – Células de memória SRAM no controle de: a) transistores de passagem, b) multiplexadores

Uma desvantagem da memória SRAM é a sua volatilidade. Devido a isso, necessita-se, conforme a necessidade da aplicação, de meios de reter os dados contidos na célula de SRAM: bateria de backup ou uma memória auxiliar (magnética ou semicondutora), por exemplo.

Outra desvantagem diz respeito à área ocupada em um circuito integrado (são necessários 5 transistores para implementar uma célula SRAM).

As vantagens são a sua rápida reprogramabilidade e por não requerer passos adicionais de integração. A primeira vantagem é o principal motivo pelo qual boa parte dos FPGAs, hoje fabricados, são baseados em células de memória SRAM (FPGAs da Altera e Xilinx, por exemplo).

A Tabela 2.1 apresenta uma comparação entre as chaves programáveis descritas acima, assumindo uma tecnologia CMOS de 1,2 μm [13]. A primeira coluna dá o nome da tecnologia usada na chave programável. A Segunda coluna indica se a configuração é perdida quando a potência é removida do dispositivo. A terceira coluna indica se a tecnologia permite

reprogramação. A quarta coluna providencia o tamanho (área) relativo da chave programável. A quinta coluna refere-se à resistência das chaves no estado ligado e a sexta coluna à capacitância parasita presente no estado desligado. A sétima coluna dá o número de passos de processamento adicionais além do padrão CMOS.

Chave Programável	R (Ω) Chave ligada	C (fF) Chave desligada	Passos extras de fabricação
SRAM	0,5 – 2 k	10 – 20	0
Antifuse ONO	300 – 500	5	3
Antifuse amorfo	50 – 100	1,1 – 1,3	3
EPROM	2 – 4 k	10 – 20	3
EEPROM	2 – 4 k	10 -20	> 5

Tabela 2.1 - Comparação das Chaves Programáveis

2.1.3. Blocos Lógicos Configuráveis (CLBs)

O bloco lógico configurável constitui-se num conceito básico dentro de um FPGA, podendo ser tão simples como um transistor, ou tão complexo como um microprocessador e capaz de realizar muitas funções combinacionais e/ou seqüenciais.

Os blocos lógicos configuráveis são, normalmente, compostos dos seguintes componentes [13]:

- Pares de transistores;
- Portas lógicas de baixo *fan-in* (portas NAND de duas entradas, normalmente);
- Multiplexadores;
- Tabelas de Busca (*Look-up Tables*);
- Estruturas AND-OR de grande *fan-in*;
- Registradores.

A diferenciação existente entre os CLBs comerciais está no tamanho, no número e na variedade de componentes internos e, conseqüentemente, na capacidade de implementação.

Um fator de mérito que se associa às arquiteturas de blocos lógicos e que se torna um mecanismo de classificação desses é a **granularidade**.

Pode-se definir granularidade de diversas maneiras, como: o número de funções booleanas que o bloco lógico pode implementar; o número de portas AND de duas entradas; o número total de transistores; a área normalizada total ou o número de entradas e saídas; a quantidade de dispositivos lógicos presentes na estrutura. Essa indefinição existe, pois, para a maioria das arquiteturas de blocos lógicos, a lógica e o mecanismo de roteamento são altamente relacionados, sendo difícil separar suas contribuições para a arquitetura [13].

Um conceito normalmente utilizado para granularidade refere-se à quantidade de dispositivos lógicos existentes no CLB. Através dessa definição, pode-se classificar os CLBs em dois tipos: CLBs com arquitetura de granulação fina (*Fine-Grained Architectures*) ou CLBs com arquitetura de granulação grossa (*Coarse-Grained Architectures*). Os primeiros correspondem a CLBs com poucos dispositivos lógicos. Já os segundos apresentam muitos dispositivos lógicos.

Vale ressaltar que a utilização de uma dessas arquiteturas depende essencialmente da aplicação e de certos compromissos de área ou desempenho, por exemplo. A seguir essas duas arquiteturas de CLBs serão caracterizadas e exemplificadas.

2.1.3.1 Arquiteturas de Granulação Fina

Os blocos lógicos com arquiteturas de granulação fina consistem, normalmente, de poucos componentes ou dispositivos lógicos. Às vezes, assemelham-se a uma célula básica de um MPGA (*Mask-Programmable Gate Array*), podendo consistir de poucos transistores, como no FPGA da Crosspoint mostrado na Figura 2.7. O bloco lógico desse FPGA corresponde a um simples par de transistores.

Um outro exemplo de um bloco lógico com essa arquitetura é mostrado na Figura 2.8. Esse bloco lógico é da empresa Plessey e possui uma RAM de configuração que permite habilitar o *latch* como transparente ou não.

A grande e principal vantagem do uso de blocos lógicos de granulação fina, é que os blocos úteis[3] são completamente utilizados. Isto é devido ao fato que é mais fácil usar portas lógicas pequenas de modo eficiente [13].

Entretanto, a principal desvantagem dos CLBs de granulação fina é que eles requerem um número grande de segmentos de fio e chaves programáveis.

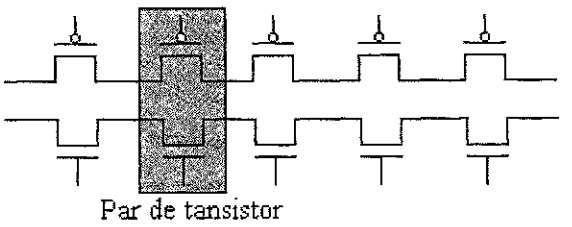


Figura 2.7 – Estrutura do CLB da Crosspoint

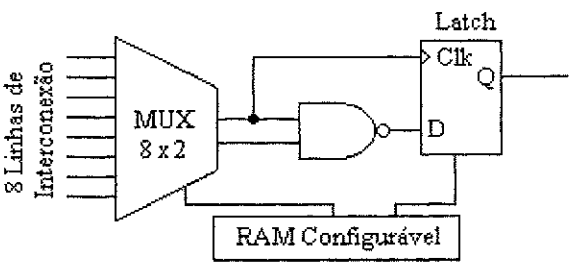


Figura 2.8 – Exemplo de um bloco lógico com arquitetura de granulação fina

2.1.3.2 Arquiteturas de Granulação Grossa

Os CLBs de granulação grossa possuem blocos lógicos mais densos e baseiam as realizações de funções lógicas combinacionais em um componente denominado Tabela de Busca (*Look-up Table*). Essa tabela de busca se constitui em um bloco de memória SRAM que armazena o conteúdo da tabela verdade de uma determinada função lógica, sendo que para **k** entradas, o tamanho dessa tabela é igual a $\mathbf{2^k x 1}$.

A vantagem de uma LUT é a sua alta funcionalidade, enquanto que sua desvantagem é que para um número de entradas maior que 5, elas tornam-se tecnologicamente inviáveis [13].

Os CLBs de granulação são compostos, além da LUT, de registradores, portas lógicas e multiplexadores controlados. Para exemplificar, a Figura 2.9 apresenta o CLB da família XC3000 da Xilinx.

Rose, El Gamal e Sangiovanni-Vicentelli [13] realizaram uma síntese das pesquisas realizadas no efeito da granularidade do bloco lógico na densidade e no desempenho de um FPGA. Por essas pesquisas, observa-se que se a granularidade de um CLB aumenta, o número de blocos lógicos para a implementação de uma dada função se reduz. Para um bloco lógico

[3] Em todos os FPGAs, assim como em todos os MPGAs, apenas uma fração dos CLBs disponíveis pode ser usada em qualquer projeto.

com menor granularidade é necessário um número maior de blocos lógicos e, assim, ocupa uma maior área de silício.

No caso do efeito da granularidade no desempenho de um FPGA, as pesquisas mostraram que o aumento da granularidade leva a uma menor número de canais de fiação e, conseqüentemente, a implementação de uma dada função será mais rápida em relação a um FPGA de menor granularidade.

Conforme essas pesquisas indicam, os FPGAs que possuem CLBs com arquiteturas de granulação fina são mais lentos e encontram densidades menores que aqueles empregando blocos de granulação grossa.

2.1.4. FPGAs com Memória Interna

Atualmente, boa parte dos FPGAs possui internamente, além da parte lógica, dispositivos de armazenamento [15]. Normalmente são utilizados blocos de memória RAM estática. As vantagens da presença de memórias internas nos FPGAs são:

- Redução da quantidade de circuitos integrados adicionais para implementação do sistema e, conseqüentemente, redução do custo total do sistema. Essa vantagem aplica-se apenas aos CLBs com arquiteturas de granulação grossa;
- Aumento da freqüência máxima disponível, devido à rapidez de acesso a memória;
- Redução da quantidade de pinos de entrada e saída, pois não há necessidade de pinos de endereçamento ou de dados para a memória.

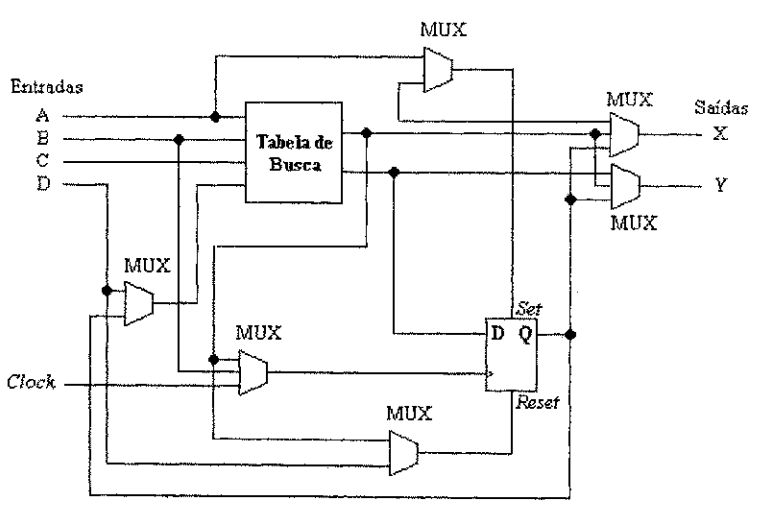


Figura 2.9 – Exemplo de um CLB com arquitetura de granulação grossa

A implementação de memórias RAM baseia-se no uso de pequenas LUTs[4] nas arquiteturas de granulação fina (por exemplo, os FPGAs da família XC4000 da Xilinx), e grandes matrizes de memória (**8kbits x 1**, por exemplo) nas arquiteturas de granulação grossa (por exemplo, o FPGA da família FLEX10K da Altera).

Quando se deseja implementar uma memória RAM grande (8k x 1, por exemplo) num CLB de granulação fina, necessita-se de muitas LUTs. Isso, por sua vez, requer a utilização de grandes decodificadores de endereços e multiplexadores de dados que irão requerer mais LUTs.

Por outro lado, memórias implementadas em CLBs de granulação grossa são mais densas e rápidas.

Até agora foi visto os elementos que compõem um FPGA. O próximo item tratará de uma classe de FPGAs bastante utilizada atualmente. Um desses dispositivos foi usado nos testes experimentais que são mostrados no capítulo 5.

2.1.6. FPGAs Reconfiguráveis

Os FPGAs que são baseados em células de memória SRAM ou em chaves de tecnologia de porta flutuante, diferentemente dos *antifuses*, podem ser configurados ou programados mais de uma vez, ou seja, podem ser reconfigurados ou reprogramados. São, assim, chamados FPGAs reconfiguráveis.

A idéia de reconfigurabilidade em *hardware* não é recente e data do início da computação. Entretanto, devido às dificuldades tecnológicas da época, a reconfigurabilidade restringia-se apenas aos componentes de *software*. Com o surgimento dos dispositivos programados em campo, especialmente os FPGAs Reconfiguráveis, esse cenário começou a mudar. Vale ressaltar que a noção de reconfigurabilidade em *hardware* é independente da aplicação, dependendo apenas dos componentes internos do FPGA Reconfigurável.

Entre as aplicações mais comuns que um FPGA Reconfigurável pode engendrar estão:

- **Prototipagem rápida:** prototipagem rápida refere-se ao tempo de desenvolvimento de uma aplicação para o mercado. Geralmente, os FPGAs Reconfiguráveis

[4] Por exemplo, LUTs de 4 entradas podem implementar memórias RAM de 16 bits.

possuem um tempo de prototipagem significantemente menor que um ASIC, sendo assim mais recomendado para aplicações que requeiram prototipagem rápida.

- **Substituição lógica:** esse campo de aplicações refere-se ao projeto de sistemas lógicos completos em um único FPGA Reconfigurável, diferentemente de outras alternativas como CIs usando tecnologia LSI ou MSI, ou mesmo, alguns PLDs ou CPLDs.
- **Computadores Padrões (*Custom Computers*)**: nessa aplicação, um certo número de FPGAs Reconfiguráveis são combinados usualmente com memória local e dispositivos de controle para produzir uma plataforma de computação reconfigurável. Os computadores padrões usam a natureza reconfigurável dos FPGARs, de modo que as tarefas específicas da aplicação podem ser implementadas por software, obtendo-se um desempenho próximo ao de um ASIC. Algumas aplicações típicas de computadores padrões são: processamento de dados com largura de banda alta e condicionamento do sinal provindo de sensores e transdutores.

Uma limitação dos FPGAs Reconfiguráveis está no fato que não é possível a reconfiguração *on-line*, ou seja, durante o decurso da aplicação.

Assim, além do conhecimento dos principais componentes de um FPGA e de seu representante, o próximo item tratará de um aspecto importante no projeto de uma aplicação em um determinado FPGA.

2.1.7. Fluxo de Projeto em um FPGA

Sendo o FPGA um ASIC tipo *semi-custom* [10, 11], seu fluxo de projeto é bem semelhante ao de um ASIC, exceto pelo fato que se encontra já fabricado e, assim, alguns passos não são realizados. Esse fluxo de projeto pode ser entendido como um algoritmo ou fluxograma utilizado com o objetivo final de programar o dispositivo para uma aplicação determinada.

A Figura 2.10 apresenta esse fluxo de projeto com os passos ou os níveis de abstração a serem seguidos para configurar ou programar um FPGA.

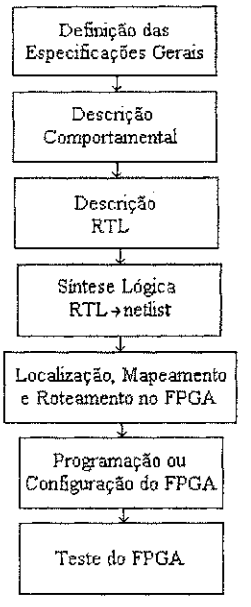


Figura 2.10 – Fluxo de Projeto em um FPGA

O primeiro passo ou o nível mais alto de abstração consiste na escrita das especificações gerais do projeto. Esse passo é normalmente realizado pelo usuário que, de acordo com as necessidades da aplicação, descreve o comportamento do sistema em um conjunto de especificações. Nessas especificações são descritas a funcionalidade e a arquitetura completa do circuito a ser projetado, com suas unidades funcionais e a interação entre estas, de modo a promover a transformação de um dado de entrada para um dado de saída desejado.

Com essas especificações deve-se criar uma descrição comportamental do sistema, ou seja, uma análise do projeto em termos de sua funcionalidade, desempenho e outras questões de alto nível e na decomposição das unidades funcionais dentro de uma série de procedimentos, identificando se estas ações serão realizadas em *hardware* ou em *software*. Essa é uma etapa que já pode ser descrita em uma linguagem descritiva de hardware (HDL - *Hardware Description Languages*), como VHDL (*Very High Speed Integrated Circuits* HDL) ou Verilog HDL. Devido a características da própria linguagem, a linguagem VHDL é melhor para esse tipo de descrição do que a linguagem Verilog.

A partir da descrição comportamental deve-se estabelecer o nível de fluxo de dados ou RTL (*Register Transfer Level*), ou seja, a decomposição do nível algorítmico ou comportamental em uma série de transformações de dados dentro do projeto. Esse nível pode ser visto como a definição do modo de entrada do projeto. Nesse caso, pode-se utilizar uma linguagem HDL ou um esquemático.

O próximo passo consiste na síntese lógica em que uma ferramenta CAD (*Computer-Aided Design*) converte o nível RTL em uma descrição (*netlist*) ao nível de portas e conexões entre essas. Em seguida, o circuito é simulado para uma verificação lógica e funcional. Normalmente, essa simulação é realizada no ambiente em que se escreveu a descrição RTL do circuito.

A partir do *netlist* gerado, o mapeamento, a localização e o roteamento (roteamento entre os CLBs e os blocos de entrada e saída) na arquitetura do FPGA são realizados de forma automática por uma ferramenta CAD (normalmente, corresponde a um *software* proprietário da empresa que fabrica o FPGA). É nesse nível que o arquivo de configuração é gerado.

Utilizando esse arquivo de configuração passa-se ao nível de programação do dispositivo. Assim, pode-se desenvolver o teste do circuito para verificar se este possui a funcionalidade e o desempenho desejados.

A maioria das ferramentas de síntese é relacionada ao fluxo de dados, ou seja, a partir da descrição RTL o arquivo de configuração do FPGA pode ser gerado automaticamente. Algumas ferramentas de síntese comportamental já existem e podem criar descrições RTL a partir de uma descrição comportamental ou algorítmica do circuito. Contudo, elas são ainda limitadas.

2.1.8. Desvantagens e Limitações de um FPGA

Apesar de apresentarem flexibilidade e reconfigurabilidade altas e serem programados facilmente usando linguagens HDL, os FPGAs são sujeitos a algumas deficiências [10, 11, 13, 14]:

- Possuem alto custo para grandes volumes de produção, quando comparados aos ASICs;
- A implementação de funções combinacionais em um FPGA é mais difícil que em um PLD, devido ao atraso de propagação ser maior. Recentemente, foi proposta uma arquitetura reconfigurável que combina as potencialidades de um FPGA, para a realização de funções seqüenciais e DSP (filtros digitais, algoritmos FFTs - *Fast Fourier Transforms*, por exemplo), e de um PLD, para a realização de funções combinacionais [16].

- As chaves programáveis, devido às não-idealidades inerentes de suas estruturas, apresentam um resistor e um capacitor parasitas (Tabela 2.1). Esses, por serem acrescentados a todas as conexões dentro do circuito, proporcionam a redução da freqüência de *clock* máxima e do desempenho do FPGA.
- Quando se deseja uma arquitetura de FPGA bastante flexível, mais bits de programação e chaves programáveis serão necessários, acarretando em uma área menor disponível para os outros elementos do circuito;
- Boa parte da área de um FPGA é ocupada pela estrutura de roteamento (70 a 90 % da área total), implicando numa redução do desempenho. Estruturas diferentes de roteamento são propostas para minimizar esse problema, como em [13].
- As ferramentas CAD utilizadas para o projeto do FPGA são, normalmente, complexas e caras. Contudo, são mais baratas que para ASICs;

2.1.9. Aplicações de FPGAs

Os FPGARs são bastante utilizados para a prototipagem de circuitos nas aplicações mais diversas, como: processamento digital de imagem [9]; implementação de funções de processamento digital de sinais (filtragem FIR – *Finite Impulse Response* ou IIR – *Infinite Impulse Response*, algoritmos FFTs, etc.), em que podem atuar como co-processadores junto a DSPs (*Digital Signal Processors*), microcontroladores ou processadores comuns, de modo a proporcionar um aumento de velocidade e desempenho na implementação dessas funções [17, 18, 19]; na geração de pulsos PWM (*Pulse Width Modulation*) para o controle de inversores trifásicos [20, 21]; em co-projetos do tipo *hardware-software* [22, 23]; comutador para redes [24].

Além de aplicada para a implementação de circuitos digitais, a idéia de reconfigurabilidade ou reprogramabilidade levou a pesquisa dos circuitos reconfiguráveis analógicos, os chamados FPAAs (*Field Programmable Analog Arrays*), e mistos, os FPMAs (*Field Programmable Mixed Arrays*) [25, 26]. Os blocos analógicos de tais dispositivos são realizados através de tecnologias, como capacitores chaveados e correntes comutadas.

Algumas empresas já comercializam FPAAs (Zetex Semiconductors) e FPMAs (Sidsa). Entretanto, tais dispositivos não são amplamente utilizados devido a dificuldades ine-

rentes relacionadas à implementação analógica em CI (ruído, interferências e baixo desempenho, por exemplo) e à falta de credibilidade e aceitação pelo mercado consumidor.

Em [26] é proposto um FPMA que contém um FPGA, um FPAA e um microcontrolador 8051 embarcado que permite, segundo seu fabricante (Sidsa), realizar funções analógicas e digitais conjuntamente ou em separado (Figura 2.11). Esse circuito, denominado de FIPSOC™ (*Field Programmable System-on-A-Chip*), foi o primeiro dessa natureza e consiste numa tendência atual, ou seja, a união de estruturas analógicas e digitais em um mesmo *chip* (os chamados *Systems On-a-Chip*), com o objetivo de aumentar o desempenho, a segurança quanto à propriedade intelectual, etc.

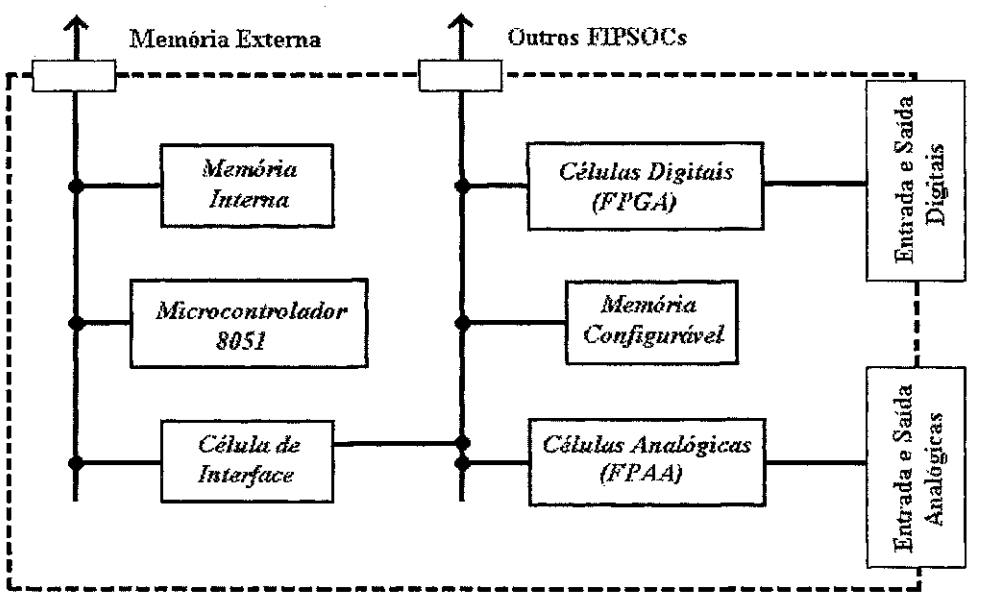


Figura 2.11 – Esquemático do FIPSOC, mostrando seus blocos internos e o roteamento interno e externo

Outra aplicação para os FPGAs Reconfiguráveis está nos co-projetos de *Hardware/Software*, em que atuam como elementos de *hardware* e um PC ou microcontrolador são os elementos de suporte ao *software* [22, 23].

2.1.9. Conclusão e tendências futuras

Os FPGAs são dispositivos lógicos programáveis usados nas mais diversas aplicações e que apresentam vantagens e limitações em relação aos ASICs e aos PLDs, em termos de

desempenho em freqüência, dissipação de potência, tempo de prototipagem e quantidade de portas lógicas.

Além disso, com o desenvolvimento de novas tecnologias de integração, estruturas de FPGAs com maior densidade de portas e melhor desempenho são propostas. Para exemplificar:

- Foi noticiado, no início do ano de 1999, um FPGA da Xilinx em cooperação com a UMC (*United Microelectronics Corporation*), que pode operar a uma freqüência de 1 GHz, utilizando tecnologia de 0.18 μm, interconexão de cobre (ao invés do alumínio, normalmente utilizado) e alimentado com 1.8 V [27];
- A empresa Altera anunciou, em 1998, a sua mais nova família de FPGAs (APEX20K) que pode possuir, para determinadas configurações, mais de 1 milhão de portas [28]. Conjuntamente, a essas novas estruturas de FPGAs, *softwares* adequados surgem para oferecer o devido suporte de simulação e programação (*software* Quartus™ da Altera, por exemplo).

Todavia, dispositivos novos, com boa densidade de portas e bom desempenho em freqüência são ainda caros, assim como as ferramentas CAD que usadas para sua programação.

Desse modo, é notado que a utilização de FPGAs está em voga, proporcionando, para projetos de pequeno e médio porte, vantagens em termos de velocidade e flexibilidade de aplicações, além da reconfigurabilidade e da facilidade de programação.

CAPÍTULO 3

Estrutura Reconfigurável Aplicada ao Processamento de Sinais Analógicos

Esse capítulo apresenta a estrutura reconfigurável proposta nessa dissertação. O objetivo dessa estrutura é oferecer uma opção nova para o processamento de sinais analógicos, caracterizada pela reconfigurabilidade e por um tempo curto de desenvolvimento de um projeto.

Além dos componentes da estrutura reconfigurável, esse capítulo apresenta também uma placa projetada para realizar fisicamente essa estrutura. A classe de aplicações que essa placa abrange é bastante ampla, tendo em vista que os componentes utilizados possibilitam atender aplicações com uma carga elevada de processamento.

3.1. Componentes da Estrutura Reconfigurável

A estrutura reconfigurável proposta é constituída por um microcontrolador, um FPGA, conversores A/D e D/A, memórias de dados e programas e interface serial com um PC, para a configuração ou programação dos dispositivos.

Uma característica adicional a essa estrutura é a possibilidade de ser autônoma, pelo uso de uma bateria. Os itens a seguir descrevem os componentes dessa estrutura.

3.1.1. FPGA

O FPGA escolhido foi um dispositivo da família de dispositivos FLEX10K da Altera, devido à familiaridade do autor com esses dispositivos. Além da família de FPGAs, um outro

fator interessante, a ser definido, é a classe de dispositivos dessa família a ser utilizada. Isso foi realizado conforme critérios de encapsulamento e capacidade de entrada e saída.

3.1.1.1. Critério de Encapsulamento

Conforme o critério de encapsulamento, uma alternativa seria o encapsulamento tipo PLCC (*Plastic J-Lead Chip Carrier*). Contudo, apenas um dispositivo possui tal encapsulamento, o EPF10K10 de 84 pinos [28].

Uma alternativa seria a utilização de encapsulamentos tipo SMD, como: PQFP (*Plastic Quad Flat Pack*) ou RQFP (*Power Quad Flat Pack*). Nesses, há a dificuldade de encontrar soquetes apropriados a preços razoáveis e em soldar o componente à placa. Todavia, como foi descrito em [9], foi assimilada uma técnica de soldagem adequada para esses dispositivos e, além disso, há uma gama ampla de dispositivos que o utilizam, variando do EPF10K10 (10 mil portas) ao EPF10K100A (100 mil portas).

3.1.1.2. Critério de Capacidade de entrada e saída

Já o critério de capacidade de entrada e saída é mais crucial e importante, conforme a aplicação sendo realizada. Por exemplo, aplicações de processamento digital de imagens necessitam, geralmente, de um maior número de entradas e saídas que aplicações de filtragem.

Assim, um dispositivo como o EPF10K10 é uma alternativa menos flexível que um EPF10K20 ou um EPF10K40, pois, além destes possuírem uma quantidade de portas maior, a quantidade de pinos de entrada e saída disponíveis é também maior.

3.1.1.3. Escolha do FPGA

Obedecendo a um certo compromisso entre os critérios acima descritos, foi constatado que a melhor opção são os FPGAs que possuem encapsulamento tipo SMD, com 208 ou 240 pinos. Como a diferença de preços não é tão díspare, foram escolhidos FPGAs de 240 pinos.

Desse modo, podem ser utilizados na placa proposta FPGAs variando do EPF10K20 ao EPF10K100A com encapsulamentos PQFP e RQFP tipo SMD. Entretanto, para o projeto do esquemático e do leiaute foi escolhido um dispositivo de 20 mil portas.

Um outro aspecto importante para o projeto da placa é a determinação do esquema de configuração usado pelo FPGA, que será descrita a seguir.

3.1.1.4. Esquema de Configuração do FPGA

Os FPGAs da família FLEX10K são programados ou configurados através de células de memória SRAM. Assim, necessariamente, uma memória externa ao FPGA é solicitada. Além disso, a cada vez que o sistema for ligado, o a arquivo de configuração deve ser carregado para o FPGA. Esse processo é conhecido como **configuração**.

Após o processo de configuração, o dispositivo FLEX10K, restabelece ("reseta") seus registradores e habilita os pinos de entrada e saída. Essa etapa é conhecida como **inicialização** e junto com a etapa de configuração forma o **modo de comando**. Após esse modo, o dispositivo entra propriamente no modo de operação do sistema, denominado **modo do usuário**.

Há dois esquemas de realizar o carregamento ou *download* do arquivo de configuração para o FPGA: um passivo, no qual o FPGA (dispositivo escravo) é configurado através de um processador externo (dispositivo mestre); outro ativo, no qual os sinais de sincronização e controle são gerados pelo próprio FPGA, sendo que uma memória EPROM externa realiza o carregamento do arquivo binário de configuração.

Dois pinos de controle (MSEL0 e MSEL1) permitem realizar a escolha do esquema de configuração. A Tabela 3.1 apresenta os estados desses pinos (0 lógico, ligado a *GND*, ou 1 lógico, ligado a *VCC*) e os esquemas de configuração correspondentes.

MSEL1	MSEL0	Esquema de Configuração
0	0	Ativo (EPROM) ou Passivo Serial
1	0	Passivo Paralelo Síncrono
1	1	Passivo Paralelo Assíncrono

Tabela 3.1 – Esquemas de configuração e seus bits de seleção

Há três esquemas de configuração passivos (Tabela 3.1), sendo que a diferença entre eles está no modo de carregamento do arquivo de configuração para o FPGA (serial ou paralela) e se o FPGA é sincronizado ou não com o dispositivo mestre.

No esquema de configuração serial (Passivo Serial), a memória utilizada para o carregamento deve ser serial, requerendo, assim, a utilização de um protocolo de comunicação e dispositivos de *interface*. Entretanto, a Altera fornece um cabo de programação apropriado, o cabo *Bit Blaster*, para carregamento serial. Esse esquema é o mais rápido no carregamento do arquivo de configuração dentre os esquemas passivos.

Os modos passivos paralelos podem ser divididos, conforme o FPGA seja sincronizado ou não com o dispositivo mestre, em: Passivo Paralelo Síncrono e Passivo Paralelo Assíncrono. No esquema assíncrono, o FPGA é mapeado em memória, facilitando, dessa forma, a interface e não ocasionando problemas de sincronização, como no esquema síncrono.

Já o esquema ativo oferece um tempo de carregamento menor que os esquemas passivos e não necessita de processador externo. Entretanto, é necessário utilizar a memória EEPROM EPC1 e o cabo serial de configuração fornecido pela Altera.

O esquema de configuração escolhido foi o modo Passivo Paralelo Assíncrono por sua simplicidade. Uma desvantagem do esquema escolhido é a velocidade de carregamento que é baixa, em relação aos esquemas ativo e passivo serial usando o cabo *Bit Blaster*. Entretanto, o tempo de configuração demorará alguns segundos, o que é plenamente aceitável.

Um outro aspecto a ser destacado na escolha do esquema de configuração a ser usado, é a determinação do arquivo de configuração binário. Conforme o esquema de configuração escolhido, os seguintes arquivos podem ser gerados pelo MAXPLUSII para configurar o dispositivo [28]:

- SRAM *Object File* (.sof): criado quando se realiza a configuração passiva serial usando um cabo de configuração serial fornecido pela Altera (FLEX *Download Cable*).
- *Programming Object File* (.pof): usado quando se usa uma EPROM (EPC1 *Configuration* EPROM) de configuração fornecida pela Altera.
- *Serial Bitstream File* (.sbf): usado de modo semelhante ao arquivo ".sof".
- *Hexadecimal (Intel-Format) File (.hex)*: usado quando a configuração é com a EPROM EPC1 da Altera.
- *Tabular Text File* (.ttf): providencia uma versão separada por vírgulas para a configuração do dispositivo, podendo ser no código fonte do microprocessador usando

comandos de inclusão. Um arquivo ".ttf" pode ser importado aproximadamente dentro de qualquer linguagem *Assembly* ou compilador de linguagem de alto nível.

- *Raw Binary File* (.rbf) é um arquivo binário que é usado para as configurações passivas. Uma restrição é que o LSB de cada byte do dado deve ser carregado primeiro.

Conforme as características acima e ao fato de Ter-se escolhido o esquema de configuração passivo paralelo assíncrono, o arquivo de configuração escolhido foi o arquivo ".rbf".

Com essa escolha, encerra-se a análise do FPGA O próximo item trata de um outro componente, o microcontrolador.

3.1.2. Microcontrolador

O microcontrolador escolhido foi um dispositivo da família de microcontroladores de oito bits da Motorola, o MC68HC11A8, que possui 256 bytes de RAM e 512 bytes de EEPROM, além de periféricos embarcados, como um conversor A/D de 8 bits, um contador autônomo, etc. [29]. A razão básica para a escolha foi devido à familiaridade do autor com tal dispositivo.

A função do microcontrolador é controlar as vias de endereço, dados e controle e gerar os sinais de controle e sincronismo para a operação correta do FPGA e dos outros componentes. Por esses requisitos e pelo fato de ser necessária a inclusão de uma memória de programa externa, o microcontrolador foi escolhido operar no modo multiplexado expandido, em que são disponibilizados 64 KB de espaço de endereçamento [29].

Para o projeto da placa, foi utilizado um dispositivo com 52 pinos e encapsulamento tipo PLCC.

O item a seguir trata da escolha dos conversores A/D e D/A.

3.1.3. Conversores A/D e D/A

A inclusão de conversores A/D e D/A na estrutura reconfigurável teve o objetivo de atender à gama de aplicações a qual essa foi destinada. A escolha desses conversores foi baseada em critérios de disponibilidade e facilidade de uso.

O conversor A/D escolhido foi o conversor de 8 bits ADC0820, cujas características principais são [30]:

- Usa uma técnica de conversão denominada *half-flash*, que consiste de 32 comparadores e dois conversores A/D *flash* de 4 bits, um para os 4 bits mais significativos e outro para os 4 bits menos significativos;
- Oferece um tempo de conversão de 1.5 μs e dissipa 75 mW de potência;
- Possui internamente um circuito *Sample-Hold*;
- Possui facilidade de interface com microprocessadores;
- As entradas e saídas lógicas são compatíveis com níveis TTL e MOS.

Já o conversor D/A escolhido foi o conversor de 8 bits AD7528, cujas características principais são [31]:

- O conversor é dual, ou seja, apresenta dois conversores D/A R-2R de 8 bits;
- O ciclo de carregamento é muito semelhante ao ciclo de escrita de uma memória RAM, proporcionando facilidade de comunicação com processadores e portas de saída;
- Pode ser alimentado na faixa de 5 V à 15 V, podendo dissipar uma potência inferior a 15 mW.

3.1.4. Memórias de Dados e Programas

Duas memórias foram incluídas à placa para armazenamento de programas e dados. A memória de programas deve armazenar o arquivo de configuração do FPGA e pode conter o arquivo de *boot* do microcontrolador, se este ultrapassar a capacidade de memória EEPROM interna.

Há duas possibilidades para a escolha dessa memória: memórias EPROM ou EEPROM. Pelo fato da memória EPROM necessitar de um apagador externo e a memória EEPROM não, foi preferido utilizar essa última por motivos de comodidade e de economia.

Conforme o tipo de carregamento (serial ou paralelo), há dois tipos de memória EEPROM. As EEPROM seriais possuem um protocolo de comunicação não muito trivial, devido à presença de algumas etapas para operações de leitura ou escrita. Além disso, há a necessida-

de um *hardware* externo para propiciar a comunicação via RS-232. Já as memórias EEPROM paralelas são acessadas facilmente, sem a presença de componentes externos ou de protocolos de comunicação um pouco complexos.

Devido aos fatores acima relacionados, as memórias EEPROM paralelas foram escolhidas para o armazenamento do arquivo de configuração do FPGA. O tamanho dessa memória estabelecido foi de 512 kbit por permitir armazenar o arquivo de configuração de um FPGA de até 40 mil portas [28]. Contudo, pode-se utilizar uma memória de capacidade maior e, assim, dispositivos programáveis com capacidade superior a 40 mil portas poderão ser usados.

3.2. Detalhes da Placa Reconfigurável Projetada

A Figura 3.1. apresenta o diagrama de blocos da placa reconfigurável, com os barramentos de comunicação e as ligações entre os seus elementos (o esquemático e o leiaute da placa projetada encontram-se no Apêndice C). Vale ressaltar que a ligação com o PC é efetuada apenas durante a programação do FPGA e do microcontrolador. Ao fim da programação, a conexão com o PC é desfeita e a placa encontra-se pronta para operar.

A freqüência de base da placa reconfigurável precisa ser um múltiplo de 8 MHz, que é a freqüência de operação do microcontrolador. Os resultados obtidos (ver capítulo 5) mostram que uma freqüência adequada para o FPGA está abaixo de 63,69 MHz (ver capítulo 5). Foi escolhida uma freqüência de 32 MHz por simplificar a divisão de freqüência (Figura 3.1).

Por fim, o leiaute da placa apresentou dimensões de 21,62 x 16,54 cm, sendo montado em 4 camadas na qual as duas camadas intermediárias são de sinal.

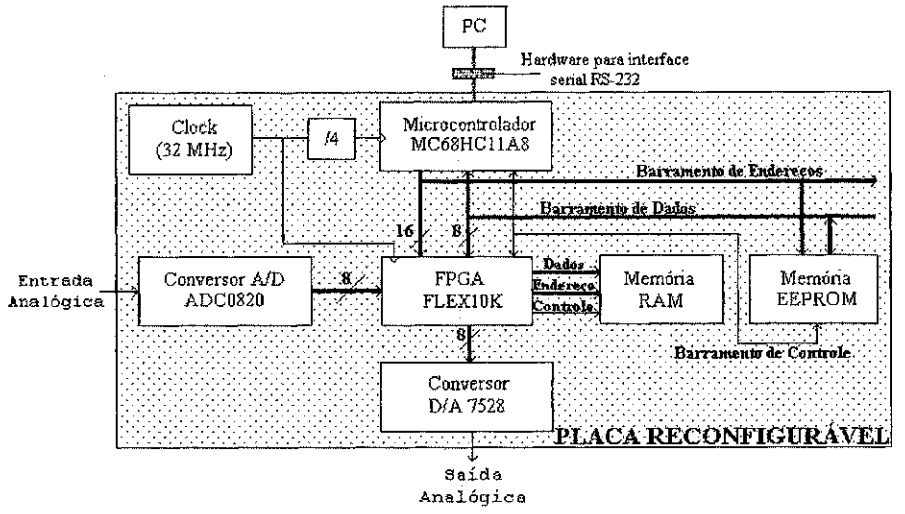


Figura 3.1 – Diagrama de blocos da estrutura reconfigurável proposta

3.3. Fluxograma para a Implementação de Aplicações

De modo a conduzir a implementação de uma aplicação específica na placa reconfigurável projetada, foi elaborado um fluxograma mostrado na Figura 3.2. Esse fluxograma é constituído de três partes, conforme o ambiente em que o projeto se encontre:

- **Usuário**: essa primeira parte compreende as tarefas iniciais a serem realizadas, em que o usuário da placa deve estabelecer as especificações gerais e dividir as funções a serem realizadas nos componentes;
- **PC**: essa parte é a programação propriamente dita do FPGA e do microcontrolador, sendo realizada através de um PC. Nessa programação são escritos os códigos Assembler ou C para o microcontrolador e Verilog HDL para o FPGA. Após os códigos terem sido escritos, passa-se a sua compilação e depuração. Os resultados finais dessa parte são os arquivos de configuração a serem usados no microcontrolador (".s19") e FPGA (".rbf");
- **Placa reconfigurável**: a última parte do projeto é o carregamento ou o *download* dos arquivos de configuração, gerados na etapa anterior, para a placa através da interface serial (Figura 3.1). Com essa parte finalizada, o PC pode ser desconectado e a placa pode iniciar sua operação.

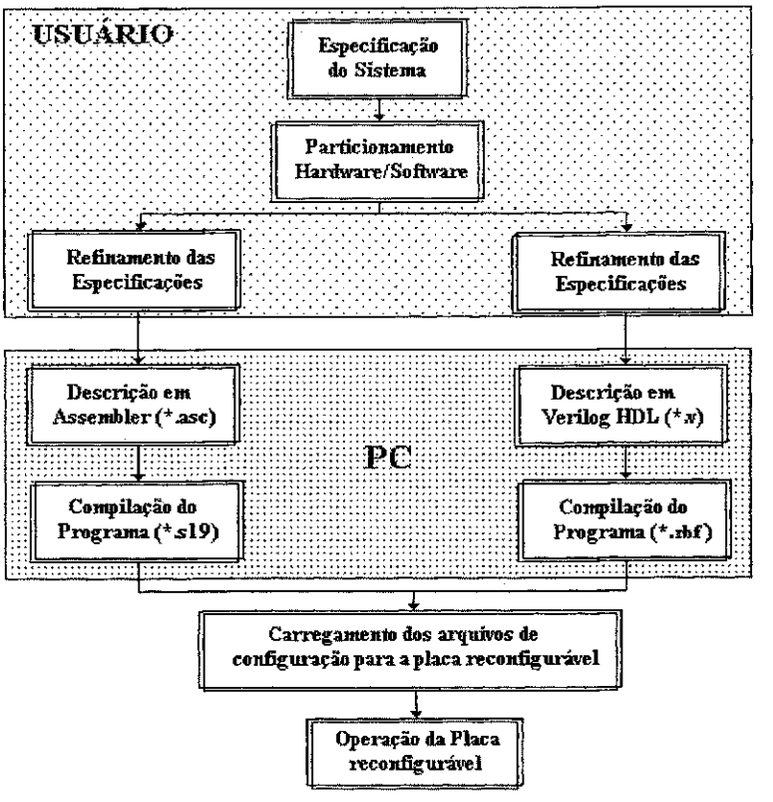


Figura 3.2 – Fluxograma a ser usado para a implementação de aplicações usando a estrutura reconfigurável proposta

3.4. Conclusão

Esse capítulo apresentou as características da estrutura reconfigurável proposta, tanto ao nível da placa reconfigurável projetada e de seus componentes, como ao nível de um fluxograma elaborado para orientar o desenvolvimento de aplicações nessa placa.

O objetivo do projeto da placa foi verificar a viabilidade da realização da estrutura reconfigurável. Futuramente, conforme o custo e outros aspectos, essa placa poderá ser enviada para fabricação.

As aplicações fins relacionam-se ao processamento de sinais analógicos. Contudo, conforme as especificações iniciais do projeto, pode ser usada para co-projetos *hardware/software* que, atualmente, estão em voga.

Um problema que pode ser notado em relação a outras alternativas, como placas que usam DSPs, está no fato da reconfigurabilidade, presente na estrutura proposta, ser *off-line*. Contudo, dependente da aplicação, os FPGAs oferecem uma maior velocidade de processamento que os DSPs, além de possuir uma programação mais fácil através das HDLs.

Capítulo 4

Testes Experimentais

Utilizando a placa projetada em Morais [9], dois testes experimentais foram realizados com o objetivo principal de validar a idéia da estrutura reconfigurável proposta. Tal validação se fez de duas formas básicas: uma que consistiu na validação da interface entre o FPGA e conversores A/D e D/A e outra na validação da interface entre o FPGA e o microcontrolador.

Os testes experimentais foram: a implementação de estruturas de filtros digitais passa-baixas e a implementação de um conversor A/D Σ-Δ monobit de 1ª ordem. Utilizaram-se, em ambos os testes, conversores A/D e D/A.

Esse capítulo foi dividido em duas partes, uma que trata da implementação dos filtros digitais e outra da implementação do conversor A/D Σ-Δ monobit de 1ª ordem. Aspectos de projeto, como escolhas de componentes e algoritmos de filtros digitais, resultados e conclusões são apresentados.

Implementação de Filtros Digitais

Essa se constitui na primeira parte desse capítulo e tratará da realização de algumas estruturas de filtros digitais passa-baixas FIR e IIR, em que uma plataforma de prototipagem reconfigurável [9], contendo um FPGA de 20 mil portas, realizou as estruturas digitais, um microcontrolador de 8 bits realizou a conversão A/D de um sinal analógico de entrada e um conversor D/A, também de 8 bits, converteu o sinal digital (saída do FPGA) para um sinal analógico. Dessa forma, a estrutura reconfigurável proposta pôde ser comprovada através das validações das interfaces entre o FPGA e o microcontrolador e entre o FPGA e o conversor D/A.

Os itens a seguir apresentarão detalhes da implementação, como o cálculo dos coeficientes, a descrição dos filtros digitais, além de resultados e conclusões.

4.1. Esquema de Implementação dos Filtros Digitais FIR e IIR

Para a realização de filtros digitais em FPGAs, a idéia básica é descrevê-los em uma linguagem HDL, com os coeficientes sendo calculados em MATLAB ou em C, por exemplo. Esses coeficientes devem ser armazenados para a posterior utilização na compilação do programa no FPGA. Tal armazenamento pode ser realizado, basicamente, de três formas: armazenamento numa memória, que pode ser implementada ou não no programa do filtro; os coeficientes são disponibilizados como variáveis de entrada; ou utilizando arquivos tipo ASCII (".dat", por exemplo).

Nos filtros digitais implementados, os coeficientes foram calculados no MATLAB e os códigos HDL dos filtros foram descritos em Verilog no programa MAX®PLUS2[5]. Os coeficientes foram armazenados em arquivos ".dat" e "chamados" dentro do código Verilog.

Para testar os filtros digitais implementados no FPGA FLEX10K20 (20 mil portas), um sinal de entrada analógico foi aplicado. O conversor A/D de 8 bits , contido no microcontrolador MC68HC11, converte o sinal analógico de entrada para um sinal digital (entrada do FPGA). Para verificar se o filtro funcionava, adicionou-se um conversor D/A de 8 bits na saída do FPGA, para verificar a recuperação do sinal analógico. Além disso, foram acrescentados filtros passa-baixas na entrada do microcontrolador e na saída do conversor D/A para evitar *aliasing* e retirar sinais espúrios (provindos de erro de conversão D/A), respectivamente.

A Figura 4.1 descreve o diagrama de blocos usado na implementação dos filtros digitais.

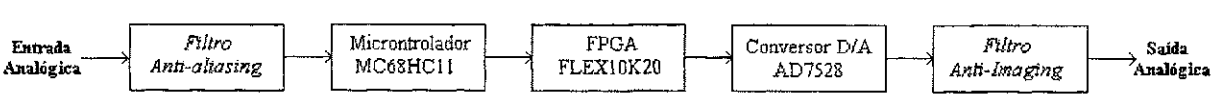


Figura 4.1 – Diagrama de blocos utilizado na implementação dos filtros digitais passa-baixas

Utilizando a estrutura mostrada na Figura 4.1, três filtros FIR (*Finite Impulse Response*) e um filtro IIR (*Infinite Impulse Response*) foram implementados.

[5] Essa ferramenta é marca registrada da empresa Altera e utilizada para o projeto com os seus CPLDs ou FPGAs até a família FLEX10K.

Os itens a seguir descreverão essas implementações, com a apresentação dos resultados experimentais e de conclusões.

4.2. Filtros FIR

A arquitetura escolhida dos filtros FIR foi a de fase linear [32, 33], sendo que três filtros com *8, 16 e 32 taps* foram implementados (Vide apêndice B para uma descrição das arquiteturas principais de filtros FIR).

No restante desse item serão apresentadas as arquiteturas dos filtros FIR, como o cálculo dos coeficientes foi realizado e resultados experimentais.

4.2.1. Descrição no FPGA

A Figura 4.2 apresenta a arquitetura de um filtro FIR simétrico de fase linear de *8 taps*, com variáveis de entrada e saída com *N* e *M bits*, respectivamente. A implementação dos filtros de 16 e 32 *taps* pode ser feita a partir da arquitetura do filtro FIR de *8 taps*, como está mostrado na Figura 4.3.

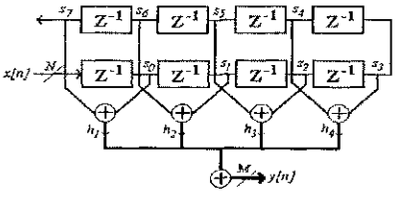


Figura 4.2 – Estrutura de um filtro FIR de 8 taps, no qual a variável de entrada possui N bits e a variável de saída M bits

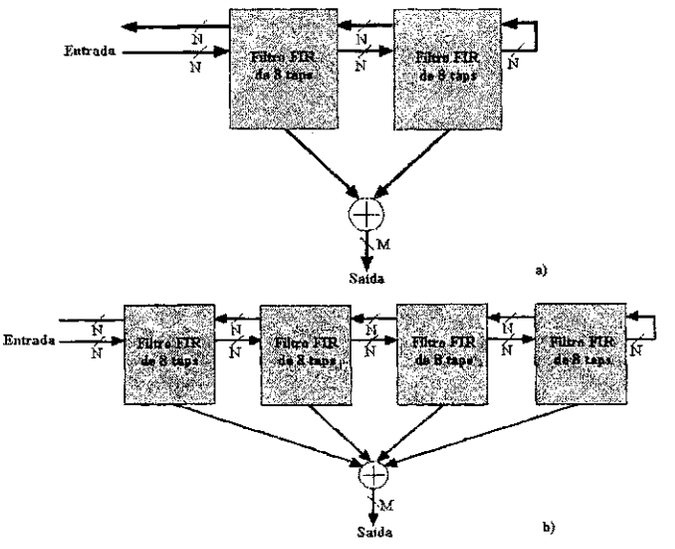


Figura 4.3 – Estruturas de filtros FIR: a) de 16 taps e b) 32 taps, baseadas na estrutura de um filtro FIR de 8 taps

De acordo com a estrutura mostrada na Figura 4.2, escreveu-se o código Verilog HDL para o filtro FIR linear de *8 taps*, no qual a variável de entrada de *8 bits* (sinal resultante do conversor A/D) foi submetida a oito atrasos, baseados na freqüência do sinal de entrada. Isso pode ser observado na Figura 4.4, em que as variáveis s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7 correspondem a registradores de deslocamento que possuem o valor da variável *entrada* (*x[n]* na Figura 4.2), a cada ciclo da freqüência de amostragem (*clk_a*).

Os filtros de 16 e 32 *taps* têm uma descrição em Verilog muito semelhante à realizada com o filtro de 8 *taps*. A diferença está na inclusão de mais registradores de deslocamento. Os programas completos desses filtros encontram-se mostrados no apêndice C.

```
always @(posedge clk_a)
begin
    s0 <= entrada;
    s1 <= s0;
    s2 <= s1;
    s3 <= s2;
    s4 <= s3;
    s5 <= s4;
    s6 <= s5;
    s7 <= s6;
end
```

Figura 4.4 – Parte do código Verilog que realiza os elementos de atraso através de registradores de deslocamento baseados na freqüência do relógio associado ao sinal de entrada

As especificações usadas para o cálculo dos coeficientes dos filtros FIR de 8, 16 e taps estão mostradas na Tabela 4.1.

Filtro FIR	[1]f_p (Hz)	f_s (Hz)	δ_p	A_s (dB)	[2]f_a (Hz)
8 *taps*	80	120	0.04	30.5	250
16 *taps*	80	120	0.004	48	260
32 *taps*	80	120	0.0001	80	280

[1] As especificações, mostradas nessa tabela, são definidas no Apêndice A
[2] Freqüência de amostragem

Tabela 4.1 – Especificações utilizadas para o cálculo dos coeficientes dos filtros FIR

A partir das especificações mostradas na Tabela 4.1. e usando a aproximação ótima no MATLAB, os coeficientes foram obtidos. Esses coeficientes foram salvos em arquivos ".dat". Contudo, antes disso, foi realizado um arredondamento (multiplicação por 128 ou 2^{n-1}, em que n é o número de bits da variável de entrada), de modo que problemas eventuais, devido ao efeito do comprimento finito da palavra no processamento [32, 33], não ocorressem.

Pelo fato de ter-se usado uma estrutura linear, apenas 4 coeficientes foram armazenados em um arquivo ".dat". Através da diretiva "*`include*" esse arquivo foi chamado no código Verilog, como está mostrado na Figura 4.5.

Conforme essa figura, há duas saídas, uma de 17 bits (y) e outra truncada com 8 bits (yt). A saída foi truncada para complementar o arredondamento dos coeficientes, realizado no MATLAB.

```
wire [8:0]soma1 = s0 + s7;
wire [8:0]soma2 = s1 + s6;
wire [8:0]soma3 = s2 + s5;
wire [8:0]soma4 = s3 + s4;

`include "fir.dat"

wire [16:0]y = -(h1*soma1) - (h2*soma2)
               + (h3*soma3) + (h4*soma4);

wire [7:0]yt = (y>>7) + (y[6]);
```

Figura 4.5 – Parte do código Verilog que implementa os somadores e multiplicadores

Nessa mesma figura são apresentados os somadores e os multiplicadores realizados com os operadores matemáticos "+" e "*". Nas operações realizadas, foi usada aritmética de ponto fixo.

Com o código escrito, foi realizada a sua compilação. O resultado dessa compilação foi a geração de um arquivo de configuração do FPGA (".ttf") [28]. Antes propriamente de se fazer o teste com o sinal analógico, procedeu-se à análise de temporização objetivando obter o

atraso mínimo de portas dentro do FPGA. Esse atraso de portas, por sua vez, determina a freqüência máxima de *clock* que pode ser utilizada para operar o FPGA na aplicação determinada.

Na Tabela 4.2, estão mostrados os valores das freqüências máximas para cada um dos filtros FIR implementados, assim como o número de elementos lógicos usados são mostrados para os três filtros FIR realizados.

Filtro FIR	Elementos lógicos	Freqüência Máxima (MHz)
8 taps	418	64,93
16 taps	725	64,51
32 taps	496	63,69

Tabela 4.2 – Resultados obtidos compilação e análise de temporização dos filtros FIR

4.3. Filtros IIR

Nesse item, será apresentada a implementação de um filtro IIR passa-baixas de 2ª ordem em sua arquitetura direta, usando os mesmos componentes do teste dos filtros FIR.

4.3.1. Descrição no FPGA

O filtro IIR de 2ª ordem possui uma estrutura muito simples, pois são necessários apenas quatro coeficientes [32, 33], conforme mostrado na Figura 4.6.

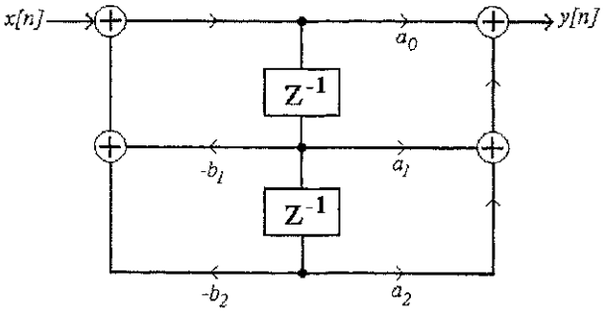


Figura 4.6 – Arquitetura na forma direta de um filtro IIR de 2ª ordem

De forma semelhante à realização do filtro FIR, mostrado no item anterior, a arquitetura de um filtro IIR compõe-se basicamente de somadores, multiplicadores e elementos de atraso. A diferença está na característica de realimentação presente nesse filtro.

O sinal de entrada *x[n]* (Figura 4.6), que é o sinal resultante da conversão A/D, e o sinal de saída *y[n]* foram submetidos a dois atrasos baseados na freqüência de amostragem f_a, conforme mostrado na Figura 4.7. Nessa figura, os termos n01 e n11 representam os termos atrasados da variável de entrada (*x[n]*), enquanto que d01 e d11 os termos atrasados da variável de saída (*y[n]*).

```
always @(posedge clk_a)
begin
  n01 <= entrada;
  n11 <= n01;

  d01 <= saida;
  d11 <= d01;
end
```

Figura 4.7 – Parte do código Verilog que descreve os atrasos do sinal de entrada e de saída, de acordo com a freqüência de amostragem

Na Figura 4.8 está apresentada a parte do código Verilog que realiza os somadores e os multiplicadores com os operadores "+" e "*", respectivamente. Nessa figura, os termos *a0*, *a1*, *a2*, *b1* e *b2* correspondem aos coeficientes calculados.

O cálculo desses coeficientes, realizado no MATLAB, foi baseado num filtro de Butterworth, com freqüência de corte igual a 100 Hz e freqüência de amostragem igual a 210,5 Hz.

```
wire [16:0] y = a0*entrada + (a1*n01)
                           - (b1*d01) + (a2*n11) - (b2*d11);
wire [7:0] yt = (y>>6) + y[5];
```

Figura 4.8 – Parte do código Verilog que realiza as multiplicações dos coeficientes pela entrada e pelos termos atrasados da entrada (n01 e n11) e saída (d01 e d11)

$$H(z) = \frac{0.8449 + 1.787z^{-1} + 0.8449z^{-2}}{1.0 + 1.7786z^{-1} + 0.8008z^{-2}} \qquad (4.1)$$

A variável de saída *y* (Figura 4.8) possui *17 bits*, o que levou aos mesmos procedimentos de arredondamento e truncamento realizados com os filtros FIR. Isso resultou na vari-

ável de saída *yt* (entrada para o conversor D/A). O programa completo do filtro IIR de 2ª ordem encontra-se no Apêndice C.

Em seguida, o código Verilog foi compilado e o arquivo ".ttf" de configuração do FPGA foi gerado. Além disso, uma análise de temporização foi realizada, sendo que a Tabela 4.3 mostra resultados para a freqüência máxima e o número de portas lógicas utilizadas.

Elementos Lógicos	Freqüência Máxima (MHz)
167	64.93

Tabela 4.3 – Resultados obtidos da compilação e análise de temporização do Filtro IIR de 2ª ordem

4.4. Conclusão

Os itens anteriores descreveram a implementação de filtros digitais FIR e IIR no FPGA FLEX10K20. Os resultados experimentais obtidos, de uma certa forma, validaram a estrutura reconfigurável proposta nessa dissertação, através da boa comunicação percebida entre o FPGA e o microcontrolador e entre o FPGA e o conversor D/A.

É claro que testes mais apropriados poderiam ser realizados e que mostrassem as possibilidades de se unir um FPGA e um microcontrolador para o processamento de sinais analógicos. Entretanto, entende-se que os testes experimentais realizados, embora simples e sem usar todas as possibilidades dos componentes da estrutura, foram satisfatórios.

Conversor A/D Σ-Δ Monobit de 1ª Ordem

O conversor A/D Σ-Δ monobit de 1ª ordem é constituído de dois blocos principais: um analógico e outro digital (vide Apêndice A[6]). Um conversor desse tipo foi implementado, com uma resolução de aproximadamente 8 bits, sendo que o bloco analógico foi realizado com

componentes discretos em um *proto-board*, enquanto que o bloco digital foi realizado usando a placa projetada em [9].

Nos itens que se seguem, aspectos de projeto e resultados experimentais, tanto do bloco analógico quanto do digital, são apresentados.

4.5. Estrutura de Implementação Utilizada

O bloco analógico, ou o modulador A/D Σ-Δ monobit de 1ª ordem, foi implementado no modo analógico contínuo, usando-se componentes discretos e comerciais. Já o bloco de processamento digital foi implementado no FPGA FLEX10K20, com os algoritmos do filtro passa-baixas e do *down-sampler* descritos em linguagem Verilog-HDL (vide Apêndice B).

Esse conversor foi testado com sinais de entrada analógicos. Para verificar se o conversor funcionava conforme o projetado, colocou-se, na saída do FPGA, o conversor D/A AD7528. Na Figura 4.9, o diagrama de blocos desse conversor é mostrado, juntamente com os filtros *anti-alising* e *anti-imaging* inseridos no *front-end* e no *back-end* do Conversor A/D Σ-Δ monobit de 1ª ordem, respectivamente.

Os itens a seguir tratam do projeto do modulador e do bloco de processamento digital, com resultados experimentais e de simulação.

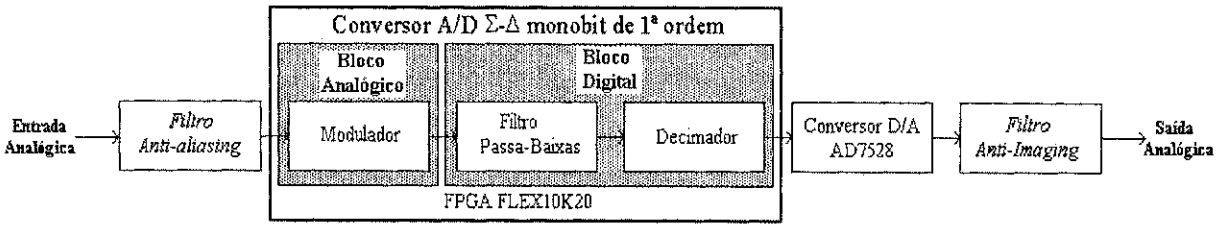


Figura 4.9 – Diagrama de blocos que mostra os componentes usados no teste experimental com o conversor A/D Σ-Δ de 1ª ordem

[6] Nesse apêndice são apresentadas as características mais importantes dos conversores A/D Sigma-Delta, como suas vantagens em relação aos conversores A./D Flash ou Aproximações Sucessivas.

4.6. Modulador

O modulador A/D Σ-Δ monobit de 1ª ordem foi montado com componentes discretos em um *proto-board*. Os itens a seguir descrevem seus componentes e os resultados obtidos no teste experimental.

4.6.1 –Descrição dos Componentes que Compõem o Modulador

4.6.1.1. Integrador

O integrador utilizado foi um integrador real (com perdas) mostrado na Figura 4.10. Nessa figura, o resistor R_I foi utilizado para limitar o ganho em baixas freqüências e descarregar continuamente o capacitor C e o resistor R_B foi usado para compensar o efeito das correntes de polarização do amplificador operacional (amp-op) TL081.

O valor de R_B deve ser tal que satisfaça a seguinte relação [34]:

$$R_B = R // R_1$$

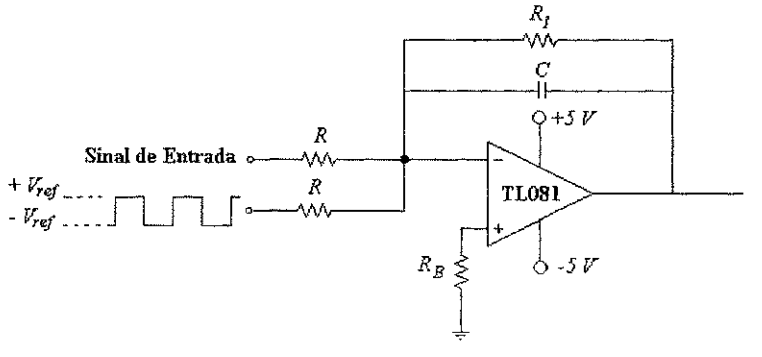


Figura 4.10 – Integrador usado no modulador A/D Sigma-Delta

Um problema com os integradores é a influência da tensão de *offset* do amp-op que leva à acumulação ou à integração contínua de carga no capacitor de realimentação (mesmo quando a tensão de entrada vai a zero) e, consequentemente, à saturação do amp-op. Isso pode ser evitado com a utilização de circuitos de *auto-reset* que zeram ou "resetam" a tensão nesse capacitor em instantes adequadamente definidos. No teste realizado não foi utilizado nenhum circuito de *auto-reset*, pois uma das entradas do integrador era um sinal pulsado que variava

simetricamente, ou seja, quando o resultado da soma das entradas do integrador era negativo, a carga armazenada no capacitor C era "zerada" (Figura 4.10).

4.6.1.2. Quantizador de 1 Bit

O quantizador de *1 bit* é um circuito (Figura 4.11) que gera um sinal quadrado a partir da comparação do sinal de saída do integrador com o sinal de terra.

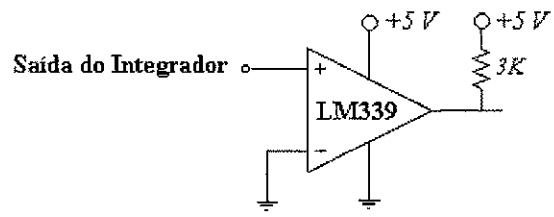


Figura 4.11 – Quantizador de 1 bit

4.6.1.3. Sobreamostragem do Sinal Resultante do Quantizador de 1 Bit

Para se obter a resolução desejada (no caso, 8 bits), o sinal quadrado resultante do quantizador de 1 bit foi sobreamostrado com uma freqüência de sobreamostragem calculada em torno de 8050 Hz.

O circuito da Figura 4.12 realizou essa sobreamostragem. Esse circuito usa o LM555, configurado como um multivibrador astável[7].

[7] Esse circuito não foi implementado no FPGA, pois necessitava-se variar o valor da freqüência de sobreamostragem e isso é complicado em um FPGA, visto que não é disponibilizada a operação on-line.

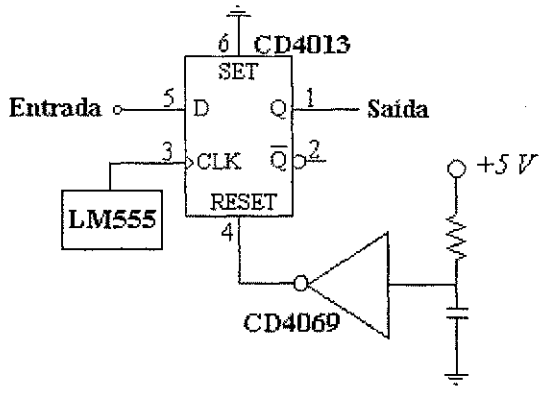


Figura 4.12 – Esquema do circuito que sobreamostra o sinal na saída do quantizador de 1 bit

4.6.1.4. Circuito que Converte o sinal TTL na Saída do Flip-Flop para um Sinal Quadrado Simétrico

O objetivo do circuito, mostrado nesse item, foi produzir um sinal simétrico que foi realimentado e somado ao sinal de entrada, gerando o sinal de erro. Assim, o sinal TTL da saída do *flip-flop* foi levado a ter uma variação simétrica de + *Vref* a – *Vref*, ou seja:

- Sinal lógico alto na saída do *flip-flop* (+5 *V*) = $+V_{ref}$
- Sinal lógico baixo na saída do *flip-flop* (*0 V*) = $-V_{ref}$

Esse sinal simétrico foi convertido para analógico por um conversor D/A de um bit (um resistor simples) e somado ao sinal de entrada. Uma observação a ser feita foi que a amplitude do sinal de entrada foi variada entre + *Vref* e – *Vref*, pois, caso contrário, o integrador não produziria um sinal variante e, assim, não seria gerado um pulso quadrado na saída do comparador.

Na Figura 4.13 mostra-se esse circuito formado por duas chaves analógicas CD4066. As tensões de referência +*Vref* e –*Vref* foram fixadas em + 1V e – 1V. Para evitar problemas de carregamento, foi colocado na saída dessas chaves um *buffer* de corrente, não mostrado na figura.

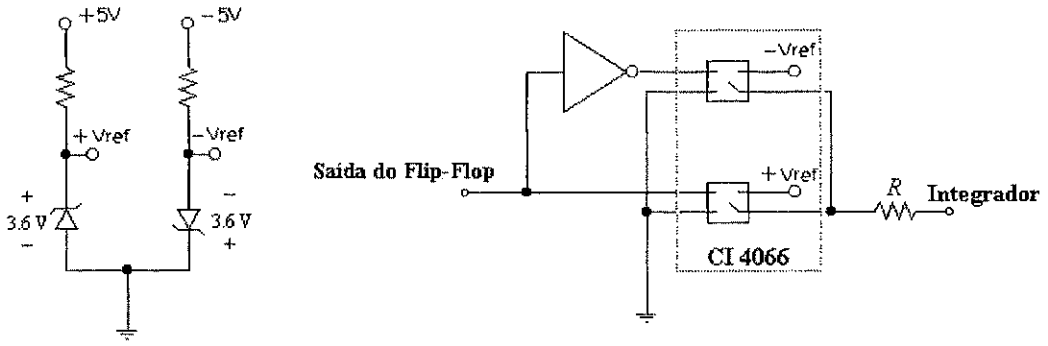


Figura 4.13 – Esquema do circuito que codifica a saída TTL Q do flip-flop para um sinal quadrado simétrico variando de $+V_{ref}$ a - V_{ref}

4.6.2 – Resultados Obtidos

O modulador A/D Sigma-Delta foi montado em *proto-board* e testado. O sinal de entrada foi uma senóide com freqüência variando de 20 a *100 Hz*. Essa faixa de freqüência foi escolhida arbitrariamente.

Nesse item são apresentados os resultados dos testes para sinais de entrada com níveis médios nulo, positivo e negativo.

4.6.2.1. Sinal de Entrada com Nível Médio Nulo

Primeiramente, foi aplicado um sinal de entrada de 1 Vpp, com nível médio nulo e freqüência de 20 Hz à estrutura do modulador Σ-Δ. O resultado obtido na saída do quantizador de 1 bit encontra-se mostrado na Figura 4.14.

Como pode ser visto por essa figura, o sinal quantizado apresenta a mesma quantidade de uns e zeros. Esse sinal, quando subtraído ao sinal de saída digital realimentado e, subseqüentemente, integrado, produz o sinal mostrado na Figura 4.15. Nesse sinal, pode-se perceber que a ação do integrador tenta, a cada instante de amostragem, levar o sinal de saída a possuir um valor médio[8] nulo, como o da entrada. Esse é a principal característica dos moduladores Σ-Δ (Apêndice A).

[8] O cálculo do valor médio da saída é [36, 37]: Nível médio = [(quantidade de uns) - (quantidade de zeros)] / (Período de amostragem)

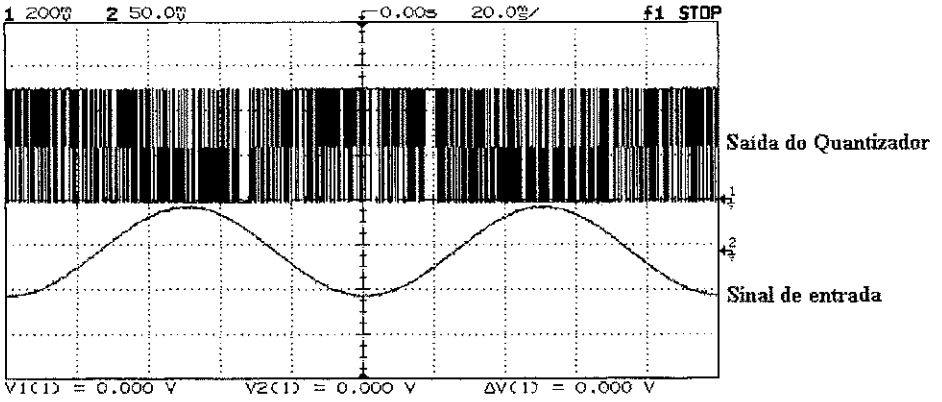


Figura 4.14 – Curvas dos sinais de entrada (nível médio nulo) e na saída do quantizador de 1 bit

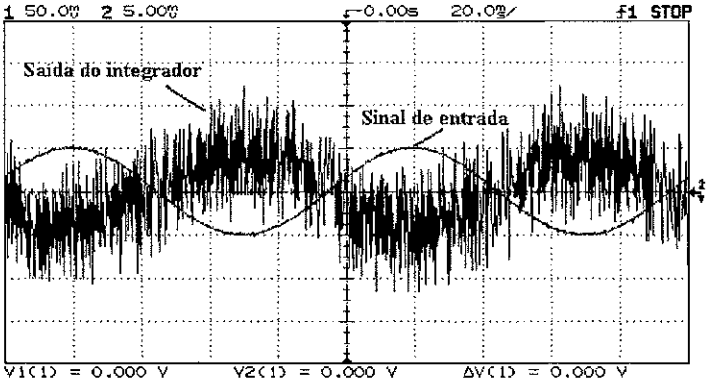


Figura 4.15 – Curvas dos sinais de entrada e na saída do integrador

4.6.2.2. Sinal de Entrada com Nível Médio Positivo

Em seguida, aplicou-se um sinal de entrada senoidal de 1 Vpp e com freqüência de 20 Hz. Esse sinal possuía um valor médio positivo, em torno de 0,25 V. Com esse sinal de entrada obteve-se a curva de saída do quantizador de 1 bit, mostrada na Figura 4.16.

Nessa figura, há uma região que apresenta apenas zeros, correspondendo à região em que o sinal de entrada é maior que o sinal analógico (semiciclo positivo da senóide) realimentado, levando a um resultado positivo no somador. Dessa forma, o integrador funciona de modo a se obter um nível médio na saída do quantizador igual ao nível médio da entrada. A ação do integrador continuará até que o sinal quantizado realimentado seja maior que o sinal de entrada (Apêndice A). A saída do integrador é mostrada na Figura 4.17.

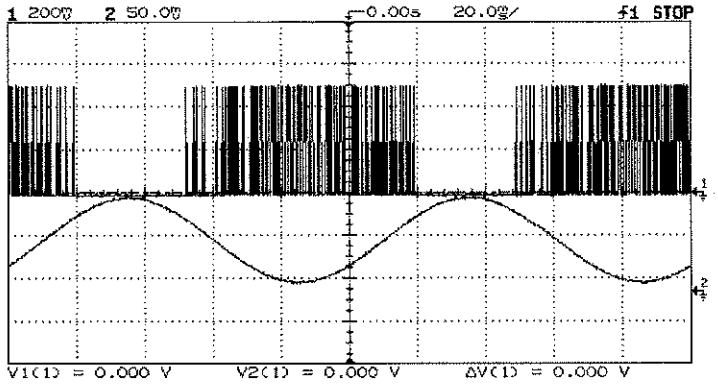


Figura 4.16 – Curvas dos sinais de entrada (nível médio positivo) e na saída do quantizador de 1 bit

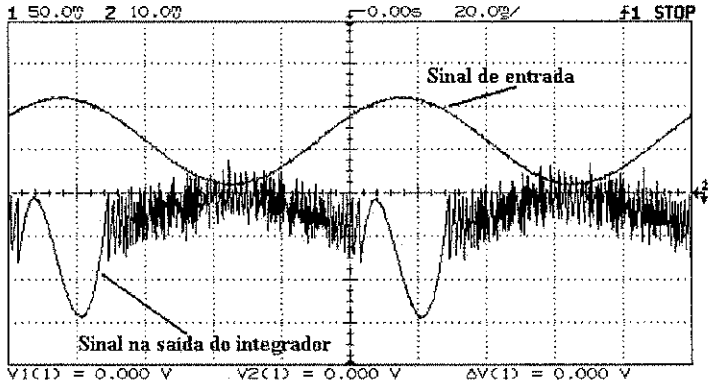


Figura 4.17 – Curvas dos sinais de entrada (nível médio positivo) e na saída do integrador

4.6.2.3. Sinal de Entrada com Nível Médio Negativo

Por fim, foi aplicado um sinal de entrada senoidal com 1 Vpp, 20 Hz e nível médio negativo, em torno de -0,25 V. O resultado na saída do quantizador de 1 bit é mostrado na Figura 4.18. Como pode ser visto por essa figura, há uma maior concentração de uns do que de zeros, correspondendo aos intervalos de amostragem em que o sinal de entrada é menor que o sinal analógico realimentado.

O integrador opera de modo a corrigir o nível médio da saída para que se iguale ao nível médio do sinal de entrada. Isso é mostrado na Figura 4.19.

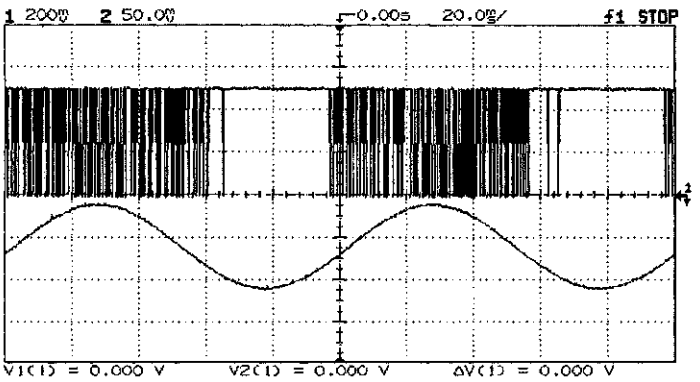


Figura 4.18 – Curvas dos sinais de entrada (nível médio negativo) e na saída do quantizador de 1 bit

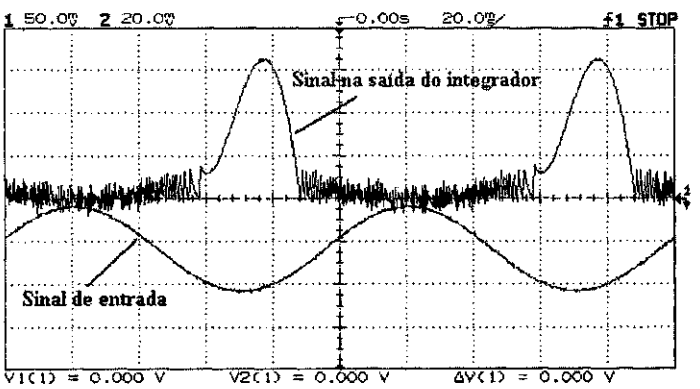


Figura 4.19 – Curvas dos sinais de entrada (nível médio negativo) e na saída do integrador

4.7. Decimador Digital

O decimador foi realizado no FPGA e seus componentes foram descritos em código Verilog HDL. Os itens a seguir tratarão disso, apresentando os resultados obtidos.

4.7.1 – Descrição no FPGA

Antes de propriamente realizar a decimação, foi necessário converter o formato serial da saída do modulador para paralelo, pois o filtro passa-baixas utilizado no decimador foi projetado como um filtro FIR paralelo. Essa conversão foi realizada no FPGA, usando-se um registrador de deslocamento de *8 bits*.

Após essa conversão, a estrutura do decimador foi descrita. O filtro passa-baixas foi um filtro FIR paralelo, cujos coeficientes foram calculados no MATLAB usando o algoritmo de Remez [32, 33], a partir das seguintes especificações:

- $f_p = 50$ Hz
- $f_s = 250$ Hz
- $\delta_p = 0.08$
- $A_s = 20$ dB
- $f_a = 8050$ Hz

Com essas especificações, um filtro FIR de 32 *taps* foi encontrado. A descrição Verilog desse filtro foi a mesma utilizada do item 4.2.3, com o acréscimo do *down-sampling* que consistiu na amostragem do sinal resultante do modulador na freqüência de Nyquist, conforme mostrado na Figura 4.20.

```
always @(posedge clkdec)
    begin
            saida <= yt;
    end
```

Figura 4.20 – Parte do código Verilog do decimador que implementa o down-sampling

O sinal *clkdec* (aproximadamente 100 Hz), por ser baseado na freqüência de amostragem, foi gerado externamente usando um contador.

Com os resultados da compilação e de uma análise de temporização foi verificado o uso de 167 elementos lógicos e uma freqüência de clock máxima de 81,30 MHz.

4.8 – Resultados Obtidos

Nessa seção, são apresentados os resultados dos experimentos realizados para o conversor A/D Sigma-Delta monobit de 1ª ordem.

O sinal de entrada aplicado foi senoidal e com nível médio positivo. Vale ressaltar que foram obtidos também resultados para sinais de entrada com níveis médios nulo e negativo, mas que não são apresentados por comodidade.

Para retirar harmônicas indesejadas resultantes da conversão D/A, foi colocado um filtro de 2ª ordem *Sallen-Key*, cuja freqüência de corte foi 100 Hz.. O circuito desse filtro encontra-se mostrado na Figura 4.21.

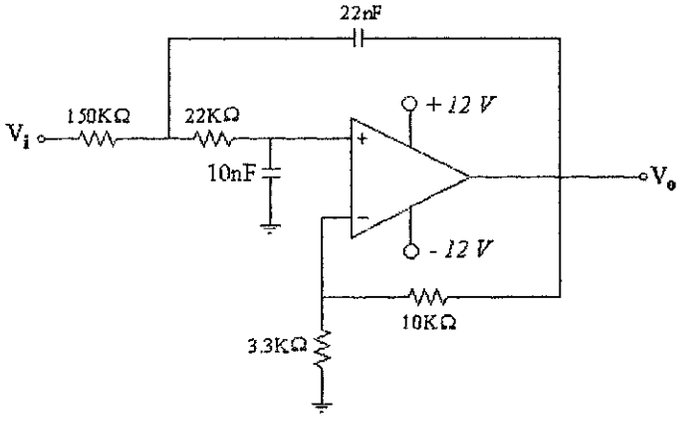


Figura 4.21 – Filtro de 2ª ordem Sallen-Key colocado na saída do conversor D/A

Com o filtro de saída montado, foram realizados os testes com o conversor. As Figuras 4.22 e 4.23 apresentam os resultados obtidos na saída do conversor D/A 7528 para dois sinais senoidais de 1 Vpp e com freqüências de 20 e 50 Hz, respectivamente.

Por essas figuras é observado um atraso e uma certa deformação nas formas de saída em relação à entrada. O atraso é conseqüência da utilização do filtro de 2ª ordem e a distorção é devido ao fato do modulador ter sido montado em um *proto-board*.

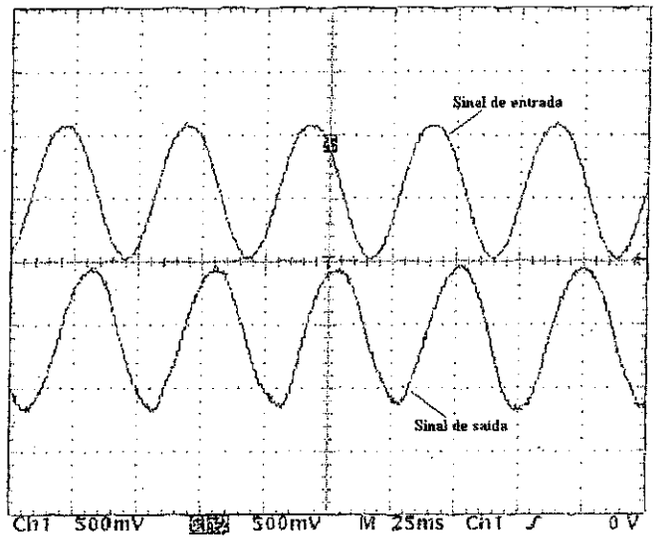


Figura 4.22 – Curvas dos sinais de entrada, com 1 Vpp e 20 Hz, e na saída do conversor

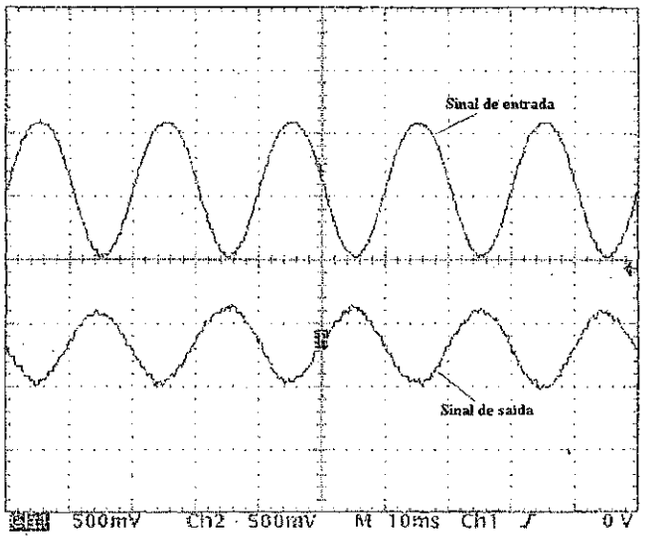


Figura 4.23 – Curvas dos sinais de entrada, com 1 Vpp e 50 Hz, e na saída do conversor

4.9. Conclusão

Nessa segunda parte das experiências foram realizados testes com um conversor A/D Sigma-Delta monobit de 1ª ordem, cujo estágio de modulação (parte analógica do conversor) foi implementado no modo analógico contínuo e o estágio de decimação (parte digital do conversor) num FPGA. Resultados experimentais foram apresentados, assim como aspectos de projeto tanto do modulador quanto do decimador.

Essa implementação, mais complexa que a dos filtros digitais, também validou a realização da estrutura proposta pela comunicação boa entre o FPGA e o microcontrolador e o FPGA e um conversor D/A.

Capítulo 5

Conclusões e Trabalhos Futuros

Essa dissertação apresentou uma estrutura reconfigurável para prototipagem rápida, aplicada ao processamento de sinais analógicos e composta por um FPGA, um microcontrolador, conversores A/D e D/A, memórias de dados e programas e elementos de interface serial com um PC. Uma placa foi projetada para essa estrutura, sendo que o leiaute e o esquemático foram apresentados.

Para a utilização dessa placa reconfigurável foi elaborado um fluxo de projeto contendo a definição das especificações gerais, o particionamento das funções a serem realizadas entre os componentes da placa e, por fim, a programação dos componentes. Junto a esse fluxo não é proposto qualquer mecanismo de automatização, em que essas fases do projeto pudessem ser realizadas em cooperação e geradas a partir de um único código de *software*. Isso ainda continua sendo uma aspiração dos pesquisadores na área de co-projeto *hardware/software*.

Essa estrutura reconfigurável pode, dependendo da aplicação destinada e das especificações de entrada, substituir implementações analógicas com componentes discretos e integradas, quando o interesse for a prototipagem rápida e o aumento de flexibilidade. Entre algumas das aplicações possíveis estão o acionamento de inversores trifásicos, o processamento de sinais provindo de sensores e o processamento matemático de sinais adquiridos da rede elétrica para a medição de parâmetros como a tensão eficaz.

As grandes vantagem dessa estrutura em relação a outras estruturas, analógicas ou mistas (partes analógicas ou digitais) são a reconfigurabilidade, um tempo de desenvolvimento curto da aplicação, a facilidade e a rapidez da reconfigurabilidade. Essa reconfigurabilidade, entretanto, é limitada ao tamanho do dispositivo (número de portas lógicas) e ao fato de ser realizada off-line.

5.1 – Trabalhos Futuros

Como continuação do trabalho aqui realizado sugere-se alguns trabalhos futuros que visam ampliar as possibilidades da estrutura proposta:

- Montar a placa projetada e testá-la em aplicações no processamento de sinais analógicos;
- Aplicar algumas idéias de co-projeto hardware/software e estudar a possibilidade de automatização de algumas das etapas de projeto do sistema reconfigurável, objetivando melhorar a interface com o usuário-fim;

Apêndice A

Conversores A/D Sigma-Delta

O uso de processadores de sinais digitais aplicados ao processamento de sinais analógicos vem crescendo muito devido aos avanços da tecnologia VLSI, que possibilita a realização de processadores digitais muito rápidos e de alto poder computacional. Em virtude disso, conversores A/D de alta resolução são normalmente requeridos para explorar o poder computacional desses processadores quando se faz necessária operação *on-line*.

Entretanto, o projeto de estruturas analógicas de conversores A/D convencionais (o sinal a ser convertido é resultante da amostragem do sinal de entrada na taxa de Nyquist), de alta resolução em tecnologia VLSI é complicado, pois são necessários geralmente componentes analógicos de precisão alta ou tolerância muito baixa [35, 36, 37].

Uma técnica que pode ser utilizada para implementar conversores A/D de alta resolução é realizar a amostragem do sinal de entrada em uma taxa muito superior à taxa de Nyquist. Essa técnica é denominada sobreamostragem e é especialmente atraente para implementação VLSI, pois os conversores A/D que usam essa técnica (conversores sobreamostrados) possuem a maior parte de sua estrutura, digital [35].

O conversor A/D Sigma-Delta é um conversor sobreamostrado bastante utilizado hoje em dia, devido às características de se conseguir resoluções altas e uma relação sinal-ruído alta a partir de uma estrutura realimentada.

Nesse apêndice serão tratadas as características principais dos conversores A/D convencionais e sobreamostrados e, em especial, do conversor A/D Sigma-Delta.

A.1. Conversores A/D na Taxa de Nyquist ou Convencionais

A.1.1. Definições e Características Gerais

Os conversores A/D na taxa de Nyquist ou conversores convencionais caracterizam-se por realizarem a conversão de sinais analógicos discretos, resultantes da amostragem do sinal de entrada em uma taxa que obedece ao critério de Shannon-Nyquist. A Figura A.1 apresenta o espectro em freqüência de um sinal analógico discreto desse tipo. Nessa figura, está mostrada também a resposta do filtro *anti-aliasing* (filtro passa-baixas inserido na entrada do conversor para limitar a banda do sinal de entrada e evitar, assim, o chamado efeito *aliasing*). A curva desse filtro assemelha-se muito a de um filtro passa-baixas ideal. Isso se deve às especificações severas desse filtro em termos de freqüência de corte ($f_B = 1/2 \cdot f_s$). Thompson [38] mostra que o decaimento na curva (não mostrada na Figura A.1) deve estar em torno de 90 dB por oitava.

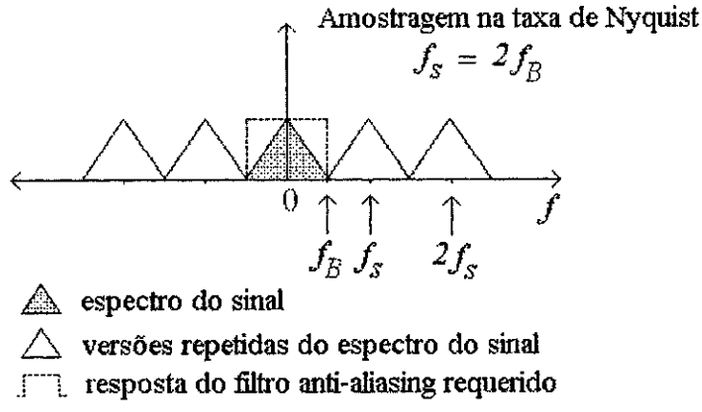


Figura A.1 - Espectro de um sinal amostrado na taxa de Nyquist e a resposta do filtro anti-aliasing requerido

A Figura A.2 apresenta um diagrama de blocos que ilustra o processo de conversão A/D convencional típico. Nessa figura, observa-se um circuito (conversor contínuo para discreto ou, simplesmente, conversor C/D), no *front-end* do processo de conversão, cuja função é converter o sinal analógico contínuo de entrada ($x_a(t)$) em um sinal analógico discreto ($x[n]$), que será usado nos processos posteriores de quantização e codificação. Esse sinal discreto é necessário, pois o processo de conversão não é instantâneo.

Duas estruturas podem ser utilizadas para a implementação desse conversor C/D: uma chamada *sample-and-hold* e outra chamada *track-and-hold*.

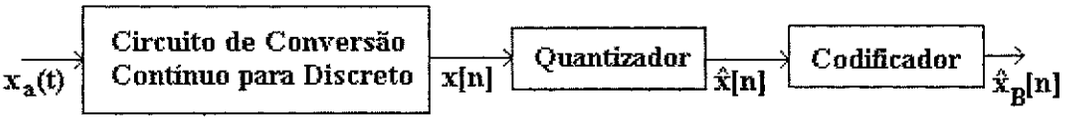


Figura A.2 – Diagrama de blocos de um conversor A/D convencional típico

Os outros dois blocos da Figura A.2 correspondem ao quantizador e ao codificador. O primeiro se constitui num sistema não-linear, cujo propósito é transformar as amostras de entrada, representadas pelo sinal *x[n]*, num conjunto finito de valores. Essa operação pode ser representada como [32, 33]:

$$\hat{x}[n] = Q(x[n]) \tag{A.1}$$

em que $\hat{x}[n]$ é referida como a amostra quantizada.

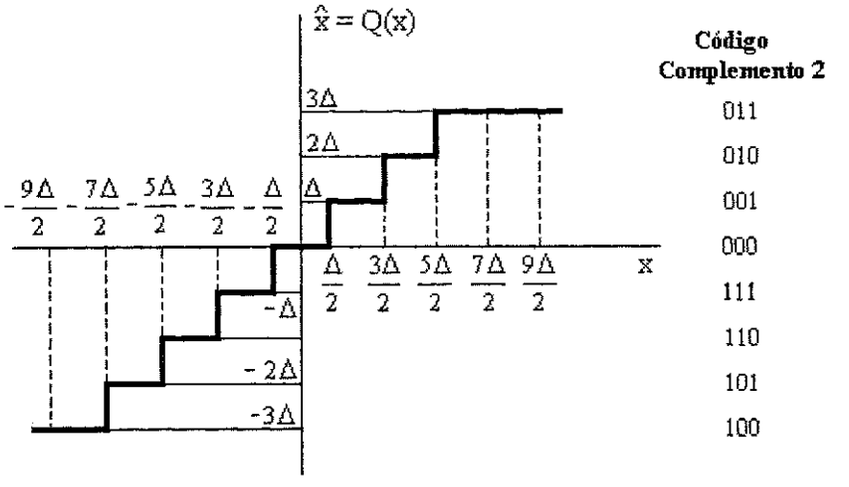


Figura A.3 – Estrutura de um quantizador típico usado em conversores A/D, junto ao código binário gerado que usa complemento 2

Já o codificador define a conversão ou a codificação das amostras quantizadas em números binários, usando determinados esquemas de codificação, como o código de complemento de 2, código *offset*, etc. A Figura A.3 ilustra os processos de quantização e codificação realizados em um conversor A/D, juntamente com a codificação usando o código de complemento 2.

Conforme mostrado na Figura A.3, há um parâmetro chamado X_m, que é denominado nível de escala completa do conversor A/D. Esse parâmetro define a relação entre as palavras binárias codificadas e os níveis de quantização. Os valores de X_m dependem do projeto, mas valores típicos são 10, 5 ou 1 volt [32, 33].

Ainda conforme a Figura A.3 pode-se definir o chamado passo de quantização, que é um parâmetro importante na caracterização de um conversor A/D:

$$\Delta = \frac{2X_m}{2^{n+1}} = \frac{X_m}{2^n} \qquad \text{(A.2)},$$

em que n é o número de bits usado pelo codificador. Vale ressaltar que os níveis quantizados menores correspondem ao bit menos significante da palavra do código binário.

Um outro aspecto interessante a ser destacado é que o valor da amostra quantizada $\hat{x}[n]$ não será, geralmente, igual ao valor da amostra verdadeira $x[n]$ (Figura A.3). A diferença existente é denominada **Erro de Quantização** e é definido como:

$$e[n] = \hat{x}[n] - x[n] \qquad \text{(A.3)}$$

Por exemplo, para o quantizador de 3 bits da Figura A.3, se $\Delta/2 \leq x[n] \leq 3\Delta/2$, então $x[n] = \Delta$. Desse modo, o erro de quantização varia de $-\Delta/2$ a $\Delta/2$.

Levando em consideração o erro de quantização, um modelo adequado para um quantizador é mostrado na Figura A.4. Nesse modelo, o erro de quantização $e[n]$ é visto como um ruído adicionado ao sinal $x[n]$ na entrada do quantizador.

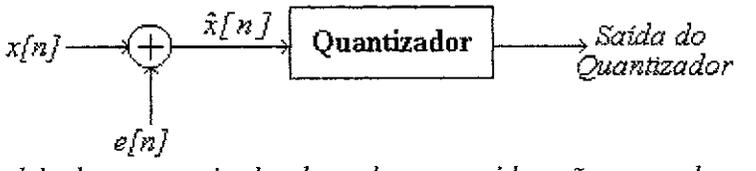


Figura A.4 - Modelo de um quantizador, levando em consideração o erro de quantização

Por não se saber, normalmente, o valor de $e[n]$, modelos estatísticos são utilizados para representar os efeitos da quantização. A representação estatística dos erros de quantização é baseada nas seguintes suposições[9] [32, 33]:

1. A seqüência de erro $e[n]$ é uma seqüência de amostras de um processo estacionário randômico;

[9] As suposições acima são mais claramente válidas para sinais complexos (sinais de voz ou de música, por exemplo) e com passos de quantização suficientemente pequenos [OPPENHEIM].

2. A seqüência de erro *e[n]* é descorrelacionada com a seqüência de entrada *x[n]*;
3. O erro de quantização é um processo de ruído branco;
4. A distribuição de probabilidade do processo do erro é uniforme sobre a faixa do erro de quantização.

Desse modo, para o quantizador de 3 bits, mostrado na Figura A.3, se o passo de quantização Δ é pequeno, pode-se dizer que o erro de quantização *e[n]* é uma variável aleatória uniformemente distribuída de -Δ/2 a Δ/2. Assim, a densidade de probabilidade de primeira ordem é como a curva mostrada na Figura A.5.

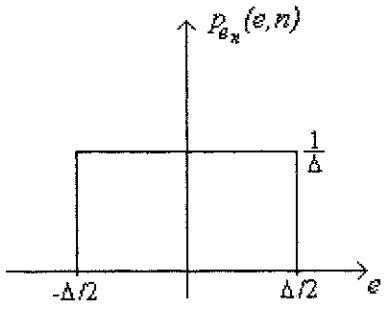


Figura A.5 - Densidade de primeira ordem do ruído de quantização

Para completar o modelo estatístico desse erro de quantização, supõe-se que as amostras sucessivas do ruído são descorrelacionadas uma com as outras e *e[n]* é descorrelacionado com *x[n]*. Assim, *e[n]* é assumido ser uma seqüência de ruído branco uniformemente distribuída, cujo valor médio é zero. Já a sua variança pode ser vista como o valor médio quadrático ou rms de *e[n]* [32, 33], ou seja:

$$\sigma_{e_n}^2 = \xi\left\{|e_n|^2\right\} = \int_{-\infty}^{\infty} |e|^2 \, p_{e_n}(e,n)de \qquad \text{(A.4)}$$

em que $\xi[*]$ é a esperança ou valor esperado de *, p_{e_n} é a função densidade de probabilidade do erro de quantização.

De acordo com a Figura A.5, a equação (A.4) fica como:

$$\sigma_{e_n}^2 = \int_{-\Delta/2}^{\Delta/2} |e|^2 \frac{1}{\Delta} de \qquad \text{(A.5)}$$

$$\sigma_{e_n}^2 = \frac{\Delta^2}{12} \tag{A.6}$$

A partir da equação (A.2), pode-se definir, assim, a expressão da variança do ruído de quantização para um quantizador com *(n + 1)* bits e com um valor de escala completa X_m:

$$\sigma_e^2 = \frac{2^{-2n} X_m^2}{12} \tag{A.7}$$

Um parâmetro útil que possibilita avaliar o efeito causado pela introdução do ruído de quantização é a relação sinal-ruído (SNR – *Signal-Noise Ratio*). Essa relação é definida como a razão entre os valores rms do sinal de entrada e do ruído de quantização. Em decibéis tem-se:

$$SNR(dB) = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \tag{A.8}$$

em que σ_x^2 representa o valor rms do sinal de entrada.

Generalizando, tem-se que para um quantizador com *(n + 1)* bits, a relação sinal-ruído é dada por:

$$SNR\,(dB) = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\log\left(\frac{12 \cdot 2^{2n} \cdot \sigma_x^2}{X_m^2}\right)$$

$$\mathrm{SNR\,(dB)} = 6{,}02 \cdot \mathrm{n} + 10{,}8 - 20\log\left(\frac{\mathrm{X_m}}{\sigma_x}\right) \tag{A.9}$$

Conforme a equação (A.9), para cada bit extra adicionado ao comprimento da palavra das amostras quantizadas (duplicação dos níveis de quantização), há um conseqüente aumento de aproximadamente 6 dB na relação sinal-ruído, desconsiderando erros estáticos de conversão [36, 37]. Normalmente, o aumento da resolução (quantidade de bits em cada amostra quantizada) do conversor é uma alternativa usada por projetistas de conversores A/D convencionais para a redução do ruído de quantização ou aumento da relação sinal-ruído.

A.1.2. Tipos de Conversores A/D convencionais

Há várias estruturas de conversores A/D convencionais, que podem ser escolhidas para uma determinada aplicação, conforme as necessidades de resolução e velocidade na conver-

são. Os mais rápidos são os conversores Flash, mas possuem a resolução normalmente restringida a 8 bits, devido a problemas tecnológicos na sua implementação em um circuito integrado (área, potência dissipada e não-idealidade dos componentes). Já os conversores de aproximações sucessivas, outro exemplo de conversores convencionais, alcançam resoluções altas (16 bits), entretanto são mais lentos que os conversores Flash.

A implementação de conversores convencionais de resolução alta (16 bits ou mais) em tecnologia integrada VLSI acarreta na necessidade de componentes analógicos com tolerância mínima e uma quase invulnerabilidade ao ruído e à interferência causada por agentes externos [36, 37]. Isso não é muito fácil de se conseguir e é por isso que conversores A/D dessa natureza são caros.

Uma alternativa para solucionar os problemas dos conversores convencionais é amostrar o sinal de entrada em uma taxa superior à taxa de Nyquist (duas vezes ou mais), ou seja, realizar a sobreamostragem do sinal de entrada. Os conversores que usam essa alternativa são destacados no próximo item.

A.2. Conversores A/D Sobreamostrados

A.2.1. Definições e Características Gerais

Os conversores A/D sobreamostrados baseiam-se no princípio da sobreamostragem, ou seja, convertem um sinal amostrado numa taxa muito superior à taxa de Nyquist. O fator de sobreamostragem (M) é o número que descreve essa característica [36, 37]:

$$M = \frac{f_{os}}{2f_B} = 2^r \qquad (A.10)$$

A Figura A.6 apresenta o espectro de algumas amostras do sinal de entrada geradas com um fator de sobreamostragem *M* igual a 2.

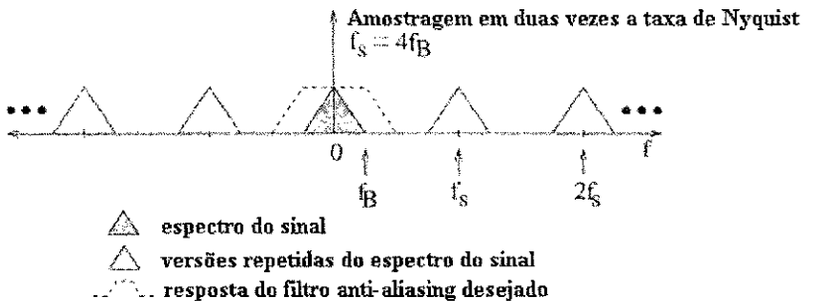


Figura A.6 - Espectro das amostras de um sinal de entrada com uma freqüência duas vezes superior à freqüência de Nyquist

Na Figura A.6, além do espectro das amostras do sinal de entrada, está mostrada também a resposta em freqüência do filtro *anti-aliasing*. Pelo fato da freqüência de amostragem ser muito alta, as especificações desse filtro não são tão severas como nos conversores convencionais e, por isso, esse filtro possui características menos abruptas de decaimento próximo à freqüência de corte. Essa é uma vantagem dos conversores sobreamostrados em relação aos convencionais.

Outra vantagem pode ser vista a partir da Figura A.7. Nessa figura, é mostrada uma comparação entre as variâncas do erro de quantização presente no processo de quantização dos conversores convencionais e sobreamostrados. Em ambos os gráficos das variâncas, a área sob a curva é a mesma. Entretanto, devido à freqüência de amostragem ser maior nos conversores sobreamostrados, a variança do erro de quantização é mais reduzida em amplitude por um fator igual à freqüência de sobreamostragem, ou seja [36]:

$$\sigma_{eos}^2 = \sigma_e^2 \cdot \left(\frac{2 \cdot f_B}{f_{os}} \right) \qquad \text{(A.11)},$$

em que σ_{eos}^2 é a variança do ruído de quantização em um conversor sobreamostrado e σ_e^2 é a variança do erro de quantização em um conversor convencional.

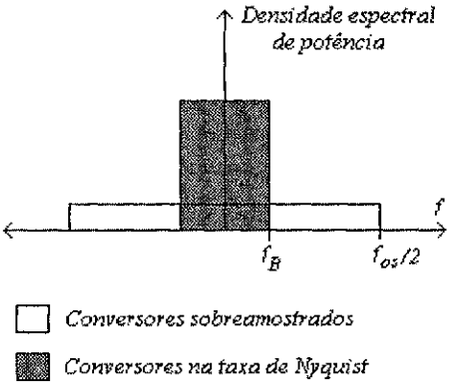


Figura A.7 – Densidades espectrais de potência do ruído de quantização para conversores sobreamostrados e na taxa de Nyquist

A redução da variança do erro de quantização em um conversor sobreamostrado pode ser também verificada através do cálculo da relação sinal-ruído. A partir das equações (A.8) e (A.11), a relação sinal-ruído para o conversor A/D sobreamostrado é dada por:

$$SNR(dB) = 10\log\left(\sigma_x^2/\sigma_{eos}^2\right) = 10\log\left[\sigma_x^2 \Big/ \left(\sigma_e^2\left(\frac{2f_B}{f_{os}}\right)\right)\right] \qquad \text{(A.12.a)}$$

$$SNR(dB) = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) + 10\log\left(\frac{f_{os}}{2f_B}\right) \qquad \text{(A.12.b)}$$

Substituindo-se a equação (A.10) em (A.12.b), tem-se:

$$SNR(dB) = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) + 10\log\left(2^r\right) \qquad \text{(A.13.a)}$$

$$SNR(dB) = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) + 3.01r \qquad \text{(A.13.b)}$$

Seja um quantizador de (n+1) bits. Dessa forma, substituindo (A.9) em (A.13.b), obtém-se:

$$\boxed{\mathrm{SNR\ (dB)} = 6{,}02 \cdot \mathrm{n} + 10{,}8 - 20\log\left(\frac{\mathrm{X_m}}{\sigma_\mathrm{x}}\right) + 3.01\mathrm{r}} \qquad \text{(A.14.a)}$$

$$\boxed{\mathrm{SNR(dB)} = 10{,}8 - 20\log\left(\frac{\mathrm{X_m}}{\sigma_\mathrm{x}}\right) + 6.02 \cdot (\mathrm{n} + 0.5\mathrm{r})} \qquad \text{(A.14.b)}$$

De acordo com a equação (A.14.a), observa-se que a cada duplicação do fator de sobreamostragem *M*, ou incremento em *r*, há um correspondente aumento de *3 dB* na relação sinal-ruído, ou, conforme a equação (A.14.b) um aumento de *0.5 bit* na resolução do conversor sobreamostrado.

A seguir são destacadas algumas estruturas de conversores sobreamostrados, baseadas na realimentação do sinal digital convertido.

A.2.2. Estruturas de Conversores Sobreamostrados

Os conversores sobreamostrados, em sua maioria, possuem uma estrutura de realização diferente em relação aos conversores convencionais. Por amostrar em uma taxa muito superior à taxa de Nyquist, não se faz necessário, na maioria das vezes, circuitos que convertam o sinal contínuo em discreto (*Sample-and-Hold* ou *Track-and-Hold*, por exemplo). Como já foi dito acima, devido à essa mesma sobreamostragem, as especificações do filtro anti-aliasing não são tão severas.

Uma outra diferença da estrutura de realização dos conversores sobreamostrados está na necessidade de um estágio de processamento digital. Esse estágio é necessário por duas razões:

- A primeira tem a ver com a redução da taxa de amostragem, pois o estágio de conversão digital para analógico (conversor D/A), que segue o conversor A/D normalmente, opera com a freqüência de Nyquist. Desse modo, o sinal digital na saída do conversor sobreamostrado deve ser decimado ou passar por um *down-sampling* [32, 33].
- A segunda refere-se à retirada dos componentes fora da banda, presentes no sinal convertido. Para isso, utiliza-se de um processo de filtragem digital.

Um exemplo de estrutura de um conversor sobreamostrado que leva em consideração todas as ponderações acima levantadas é mostrado na Figura A.8. Nessa figura, há uma nomenclatura diferente da utilizada para os conversores convencionais. O bloco **modulador** realiza, baseado em um certo *clock* ou freqüência de sobreamostragem, a conversão do sinal analógico de entrada, já devidamente filtrado pelo filtro anti-aliasing, em um sinal digital cujas amostras são sincronizadas na freqüência de sobreamostragem. O bloco **filtro digital** realiza a filtragem do sinal na saída do modulador (a freqüência de corte desse filtro é igual à

largura de banda do sinal de entrada), de forma a retirar sinais espúrios normalmente presentes. O bloco **registrador** realiza a decimação do sinal digital, ou seja, reduz a freqüência das amostras à taxa de Nyquist.

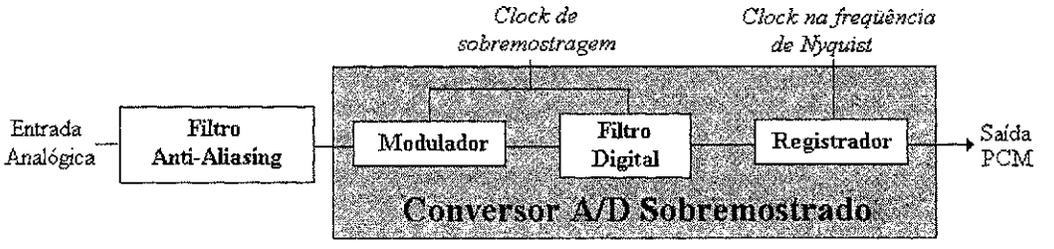


Figura A.8 – Diagrama de blocos de um conversor A/D sobreamostrado, exemplificado pelo estágio de codificação de um sistema PCM sobreamostrado

Um dos problemas de um conversor sobreamostrado diz respeito ao estágio de filtragem digital (Figura A.8), pois esse filtro deve possuir uma freqüência de amostragem igual à freqüência de sobreamostragem e a sua freqüência de corte deve ser igual à freqüência de Nyquist. Sendo assim, esses filtros, devido a essas especificações, são geralmente complexos e de ordem elevada.

Embora essa complexidade seja uma desvantagem em relação aos conversores convencionais, não é tão problemática quando o objetivo for projetar conversores A/D de resolução alta, pois é mais fácil providenciar, em tecnologia VLSI, circuitos digitais rápidos do que circuitos analógicos precisos.

Uma outra desvantagem dos conversores sobreamostrados está no campo de aplicações que se restringem, basicamente, às aplicações de freqüências baixa e média, como: áudio digital, instrumentação e telefonia digital. Isso ocorre porque a freqüência de amostragem ($M.f_b$), dependendo das especificações do projeto, poderá ser de um valor tal que inviabilize tecnologicamente a implementação do conversor.

Há dois tipos básicos de conversores sobreamostrados: os conversores preditivos e os conversores que produzem *Noise-Shaping*. Esses conversores utilizam a realimentação do sinal digital para produzir características denominadas Predição e *Noise-Shaping*, respectivamente. O resultado final é a redução da variança do ruído de quantização na largura de banda do sinal de entrada e o aumento da relação entre a resolução do conversor e o fator de sobreamostragem.

Os próximos itens descreverão definições de Predição e *Noise-Shaping* e dos conversores sobreamostrados que produzem essas características.

A.2.2.1. Predição

A Figura A.9 apresenta a estrutura de um conversor sobreamostrado que produz a característica de Predição. Esse conversor é denominado conversor A/D Preditivo [35].

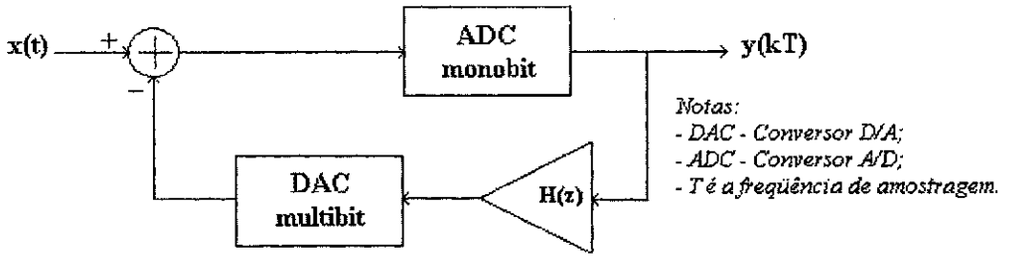


Figura A.9 – Diagrama de blocos que mostra o modulador de um conversor A/D sobreamostrado Preditivo, na qual é empregada a realimentação do sinal de saída através de um integrador

Nessa estrutura, o sinal de saída resultante do bloco modulador é realimentado através de uma função *H(z)* (normalmente um integrador). Sendo assim, o sinal resultante, após a passagem pelo conversor D/A multibit, é um valor estimado do sinal de entrada. Daí o nome Predição. Esse valor estimado é então subtraído do sinal de entrada real e um sinal de erro é assim gerado. Em seguida, tal sinal de erro é quantizado por um quantizador de um bit apenas, dando reinício ao processo.

A idéia nesses conversores é reduzir a faixa dinâmica do sinal de entrada no quantizador e, portanto, reduzir o passo de quantização Δ e, consequentemente, o erro de quantização. Os conversores A/D Delta e DPCM (Delta PCM) são exemplos de conversores sobreamostrados preditivos [35].

Outra característica desses conversores diz respeito à possibilidade de se obter amostras com resoluções altas usando um quantizador de apenas um bit, o que é grandemente vantajoso em relação aos conversores A/D convencionais.

Uma desvantagem os conversores sobreamostrados preditivos é que quaisquer erros de integração ou do conversor D/A serão quantizados e levados ao processador digital. Após esse processamento digital, há, no estágio de decodificação, um integrador que acumulará os erros

do conversor preditivo e gerará, assim, um sinal analógico resultante diferente do sinal de entrada.

A.2.2.2. Noise-Shaping

Para solucionar as desvantagens apresentadas pelos conversores A/D preditivos na transmissão de sinais, *Cutler* em 1954 [37], propôs a estrutura de um conversor que, usando a realimentação do sinal digital na saída do modulador, produzia uma alteração na forma de onda da variança do ruído de quantização, ou seja, ao contrário dessa variança apresentar um aspecto contínuo na largura de banda do sinal de entrada (Figura A.8), apresentava um aspecto como o mostrado na Figura A.10. Por essa figura, nota-se que a variança do ruído de quantização apresenta-se reduzida na largura de banda do sinal de entrada e ampliada fora.

A estrutura realimentada que permite obter uma característica dessas é mostrada na Figura A.11. Nessa figura, nota-se que a integração é realizada no caminho direto e não no caminho de realimentação. Outra característica é que tanto o quantizador, como o conversor D/A possuem um bit apenas. Entretanto, outras estruturas de conversores A/D, como será destacado no item, possuem quantizadores e conversores D/A com maior número de bits.

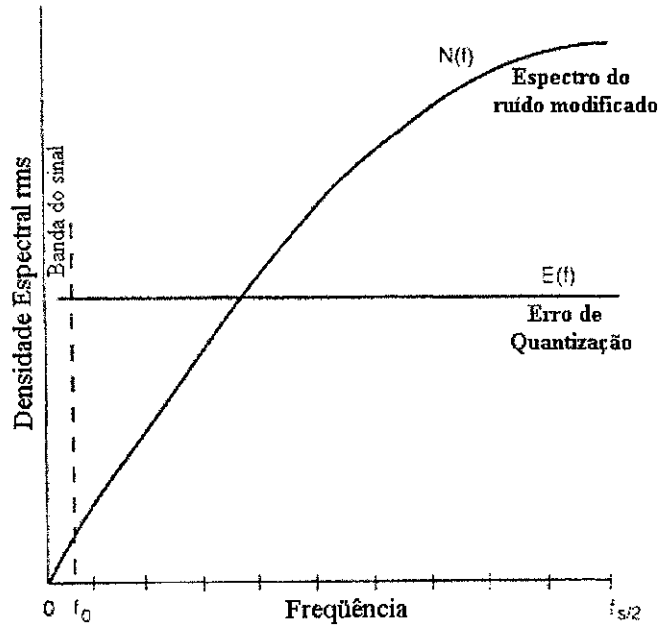


Figura A.10 – Característica de Noise-Shaping

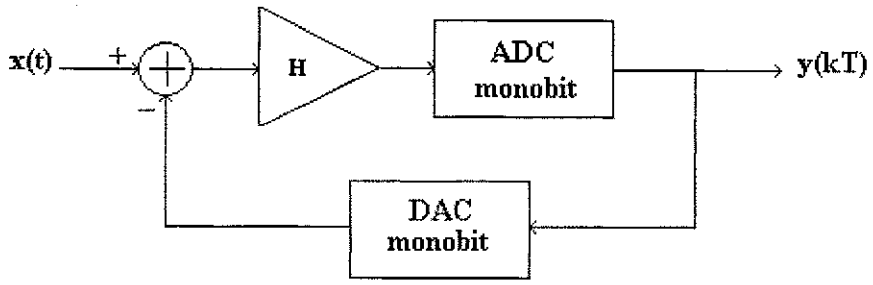


Figura A.11 – Diagrama de blocos de um conversor sobreamostrado que produz a característica de Noise-Shaping mostrada na Figura A.10

Um exemplo de conversores sobreamostrados que produzem a característica de *Noise-Shaping* é o conversor A/D Sigma-Delta, cujas definições e características serão mostradas no próximo item.

A.3. Conversores A/D Sigma-Delta

Em 1962, os pesquisadores Inose e Yasuda propuseram um modulador que, usando sobreamostragem e uma estrutura de realimentação que produzia a característica de *Noise-Shaping*, propunha-se solucionar as deficiências dos moduladores PCM convencionais e Delta. Esse modulador foi proposto para telemetria e chamado de modulador Delta-Sigma, pois baseava seu funcionamento em operações sucessivas de diferenciação (termo Delta ou Δ) e acumulação (termo Sigma ou Σ). O acumulador ou integrador encontrava-se no caminho direto e não no caminho de realimentação, como nos moduladores Delta. Isso propiciou a vantagem de não ser necessário um integrador no estágio de decodificação.

Baseados na idéia proposta por Inose e Yasuda, outros pesquisadores, como John Candy [37], começaram a pensar em conversores A/D Delta-Sigma ou Sigma-Delta (nome consagrado devido ao uso). Tais conversores incorporavam, além do modulador, um estágio de processamento digital após o modulador. Esse estágio, denominado Decimação, se faz necessário para retirar sinais espúrios e para que o sinal seja reamostrado para a taxa de *Nyquist*.

Como vantagens dos conversores A/D Sigma-Delta (Σ-Δ), em relação aos conversores A/D convencionais, estão:

- Maior robustez às variações dos componentes e às imperfeições do circuito.

- Menor sensibilidade aos erros de canal, devido à ausência de erros de acumulação no decodificador.
- Redução do ruído de quantização na largura de banda do sinal de entrada.
- Possibilidade de se obter resoluções altas com quantizadores de poucos bits.

Uma limitação dos conversores A/D Σ-Δ diz respeito ao forte compromisso existente entre resolução e largura de banda do sinal de entrada que pode ser convertida. Isso é mostrado na Figura A.10, que ilustra a comparação desse conversor com dois conversores convencionais (*Flash* e Aproximações Sucessivas). Nessa figura, é mostrado no eixo vertical a resolução das amostras que pode ser obtida e no horizontal, a largura de banda do sinal de entrada ou a faixa de aplicações na qual esse conversor pode ser utilizado.

Desse modo, o conversor Σ-Δ pode alcançar amostras com resoluções maiores que os dois outros conversores mostrados. Entretanto, apresenta uma faixa de aplicações limitada às freqüências baixas e médias. Vale ressaltar que os conversores analisados aqui são todos conversores A/D passa-baixas, ou seja, só convertem sinais abaixo de uma determinada freqüência.

Recentemente, foi proposto um conversor A/D Σ-Δ passa-faixas que, diferentemente dos passa-baixas, converte uma faixa de freqüência com freqüência central elevada. Um exemplo desses conversores é mostrado na Figura A.13 [39]. Trata-se de um conversor passa-faixas de 8ª ordem para conversão de uma faixa de *200 kHz*, centrada na freqüência intermediária de *10.7 MHz*. Esse conversor possui uma faixa dinâmica de *67 dB* e foi implementado em tecnologia BiCMOS de *0.8 μm*.

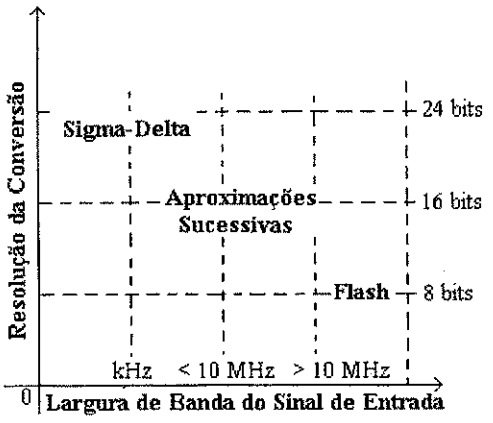


Figura A.12 – Gráfico que mostra o forte compromisso entre largura de banda e resolução para os conversores A/D convencionais e Sigma-Delta

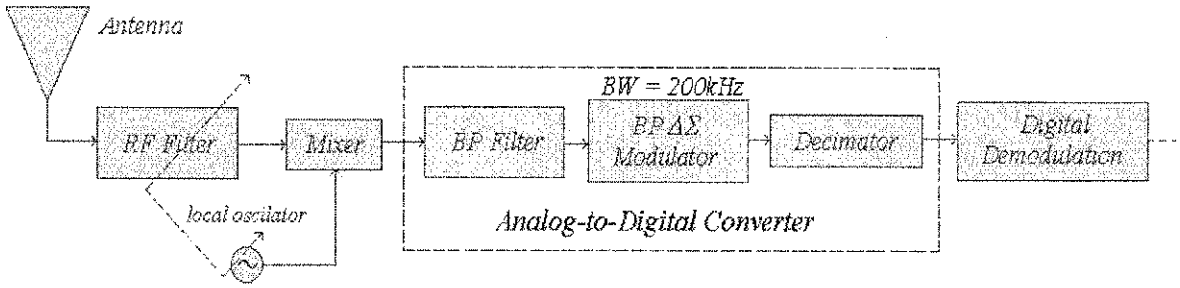


Figura A.13 – Conversor A/D Sigma-Delta passa-faixas de 8ª ordem [39]

A seguir, serão mostradas as características e a arquitetura de um conversor A/D Sigma-Delta monobit de 1ª ordem, que se constitui na estrutura básica dos conversores A/D Sigma-Delta. Além disso, serão descritas outras estruturas, algumas aplicações e realizações.

A.3.1. Conversores A/D Sigma-Delta de 1ª Ordem

A estrutura mais simples de um conversor A/D Sigma-Delta é àquela da Figura A.11, ou seja, uma estrutura monobit (o quantizador possui apenas um *bit*) e de 1ª ordem (o integrador é de 1ª ordem). Embora não seja mostrado, deve haver um bloco de processamento digital que realize o *down-sampling* e a filtragem do sinal de saída.

Nos itens a seguir, serão descritas características e estruturas de realização do bloco modulador e de processamento digital de um conversor A/D Σ-Δ monobit de 1ª ordem.

A.3.1.1. Modulador A/D Sigma-Delta Monobit de 1ª Ordem

Nesse bloco, o sinal de entrada analógico é convertido, usando sobreamostragem e uma estrutura de realimentação que produz *Noise-Shaping*, para um sinal digital que possui sua resolução relacionada ao fator de sobreamostragem *M*.

Na Figura A.14 são mostradas as estruturas de análise contínua e discreta de um conversor A/D Σ-Δ. O bloco **quantizador** é completamente analógico e pode ser realizado ou no modo contínuo (Figura A.14.a), utilizando componentes discretos, ou no modo discreto (Figura A.14.b), usando a técnica de capacitores chaveados ou correntes chaveadas. A última realização, por ser facilmente integrável, é preferida pelos projetistas.

Normalmente, a análise do funcionamento de tal conversor é realizada utilizando a estrutura discreta, por ser mais direta e simples. Desse modo, de acordo com a Figura A.14.b[10], pode-se retirar que *u[n]*, entrada do integrador, é dada por:

$$u[n] = x[n] - y_a[n] \qquad \text{(A.15)},$$

em que $y_a[n]$ representa o sinal convertido para analógico da saída digital do modulador e *x[n]* é o sinal de entrada.

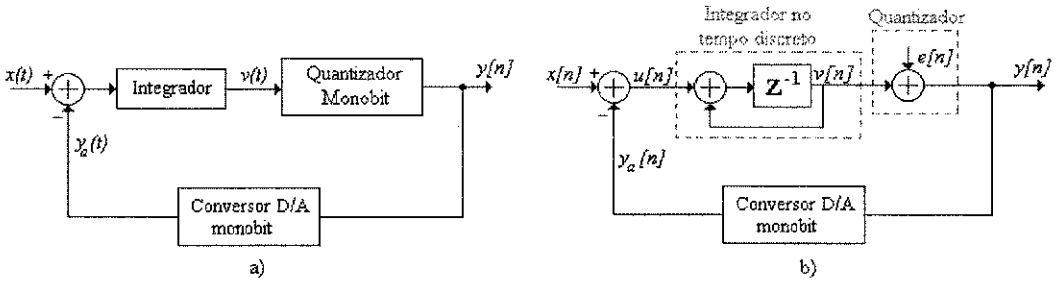


Figura A.14 – Estrutura de um conversor A/D Sigma-Delta de 1ª ordem: a) modulador realizado na forma contínua; b)modulador realizado em tempo discreto

O integrador é, assim, responsável em corrigir quaisquer erros devido a desigualdades entre esses dois sinais, levando o sinal *v[n]* (saída do integrador) a um valor com amplitude pico-a-pico muito baixa.

Já o sinal de saída quantizado (*y[n]*) constitui-se numa seqüência de zeros e uns, cuja maior quantidade de zeros ou de uns depende do valor médio do sinal de entrada. Caso o valor médio do sinal de entrada seja zero, haverá a mesma quantidade de zeros e uns. Para valores médios positivos, a quantidade de uns será maior que a de zeros e para negativos, a quantidade de zeros é que será maior. Assim, o funcionamento do modulador A/D Sigma-Delta baseia-se em buscar a igualdade dos valores médios da saída e da entrada e, correspondentemente, levar a diferença mostrada em (A.15) a zero.

Devido a essa característica da densidade de uns e zeros do sinal de saída digital depender do valor médio do sinal de entrada analógico, um modulador A/D Σ-Δ é conhecido também como um modulador por densidade de pulso ou, simplesmente, um modulador PDM (*Pulse Density Modulator*).

Para se determinar a tensão de saída digital, deve-se notar primeiramente que essa tensão *y[n]* é igual a $y_a[n]$ e é regida pela seguinte equação:

[10] O bloco de conversão D/A, no caminho de realimentação, é modelado como um ganho unitário.

$$y[n] = e[n] + v[n] \tag{A.16}$$

A tensão de saída no integrador $v[n]$ é dada por:

$$v[n] = u\,[n-1] + v[n-1] \tag{A.17}$$

Multiplicando a equação de diferenças que define $u[n]$ (A.15) por z^{-1} e substituindo o valor obtido em (A.17), tem-se:

$$v[n] = v[n-1] + x[n-1] - y[n-1] \tag{A.18}$$

Multiplicando a equação (A.16) por z^{-1} e substituindo o valor obtido em (A.18), tem-se:

$$v[n] = v[n-1] + x[n-1] - e[n-1] - v[n-1] \tag{A.19.a}$$

$$v[n] = x[n-1] - e[n-1] \tag{A.19.b}$$

Por fim, substituindo a equação (A.19.b) na equação (A.16):

$$y[n] = e[n] + x[n-1] - e[n-1] \tag{A.20}$$

A diferença mostrada na equação (A.20) indica que o erro de quantização, devido à característica de realimentação negativa do modulador Sigma-Delta e à presença do integrador no caminho direto, é menor que no caso dos conversores A/D convencionais, cuja relação entrada-saída, no modo discreto, é dada por:

$$y[n] = x[n] + e[n] \tag{A.21}$$

Fazendo a transformada Z da equação A.20, obtém-se:

$$Y[z] = X[z].z^{-1} + E[z].(1 - z^{-1}) \tag{A.22}$$

De acordo com (A.22), observa-se que, em baixas freqüências, o sinal de erro $E[z]$ será aproximadamente nulo, levando o sinal de saída $Y[z]$ ser função apenas do sinal de entrada $X[z]$. Para altas freqüências, o sinal de erro será muito elevado. Portanto, pode-se dizer que o comportamento do sinal de erro $E[z]$ assemelha-se a de um filtro passa-altas (Figura A.10).

Em [36] é mostrado que a variança do ruído de quantização para um modulador A/D Σ-Δ de 1ª ordem é dada por [43]:

$$\sigma_{sd}^2 = \sigma_e^2 \frac{\pi^2}{3}\left(\frac{2f_B}{f_{os}}\right)^3 \tag{A.23}$$

Assim, a relação sinal ruído é:

$$SNR(dB) = 10log\left(\sigma_x^2/\sigma_{sd}^2\right) \tag{A.24}$$

$$SNR(dB) = 10log\left(\sigma_x^2\right) - 10log\left(\sigma_e^2\right) - 10log\left(\frac{\pi^2}{3}\right) + 30log\left(\frac{f_{os}}{2f_B}\right) \tag{A.25}$$

Chamando-se a relação $f_{os}/2f_B$ de 2^r, obtém-se:

$$SNR(dB) = 10log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) - 10log\left(\frac{\pi^2}{3}\right) + 9.03r \tag{A.26}$$

Assim, para um quantizador de $(n+1)$ bits, obtém-se:

$$SNR(dB) = 6{,}02 \cdot n + 10{,}8 - 20log\left(\frac{X_m}{\sigma_x}\right) - 10log\left(\frac{\pi^2}{3}\right) + 9.03r \tag{A.27}$$

Substituindo-se (A.9) em (A.27):

$$SNR(dB) = 5{,}63 - 20log\left(\frac{X_m}{\sigma_x}\right) + 6{,}02(n + 1{,}5r) \tag{A.28}$$

Através da equação (A.28), verifica-se que para cada duplicação do fator de sobreamostragem M ou incremento no fator r, há um correspondente aumento de *1,5 bits* na resolução do conversor ou um aumento de *9 dB* na SNR (A.27).

Apesar da simplicidade de realização de um modulador A/D Σ-Δ de 1ª ordem, restrições devido às dificuldades de implementação de circuitos integrados com freqüências de amostragem elevadas (na faixa de GHz) impedem a sua maior difusão. Por exemplo, para um sinal de entrada com largura de banda de 20 MHz, a freqüência de amostragem necessária para se obter uma resolução de 10 bits deveria ser igual a 1,28 GHz. Por esse motivo, as aplicações para as quais esse modulador é destinado são, normalmente, as de freqüências baixas e médias.

Esse problema pode ser solucionado com a utilização de outras estruturas de moduladores Sigma-Delta, como o modulador Sigma-Delta de 2ª ordem, em que há dois integradores em série no caminho direto, como está mostrado na Figura A.15. Tal modulador é utilizado basicamente para aplicações de áudio e permite um aumento de 2,5 bits para cada duplicação do fator de sobreamostragem M [36, 37].

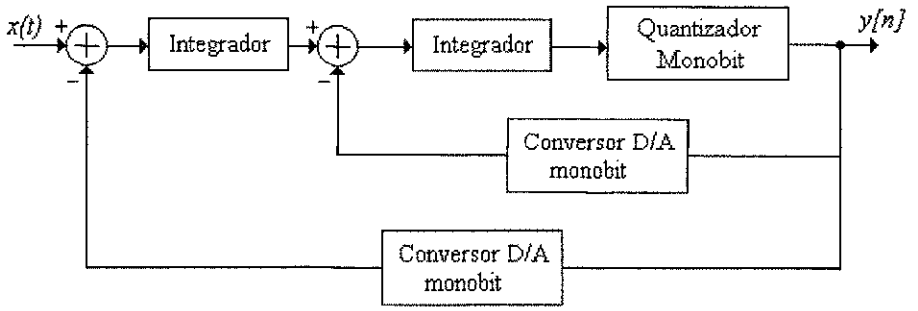


Figura A.15 – Diagrama de blocos de um modulador A/D Sigma-Delta de 2ª ordem

Generalizando-se, a equação que define a relação sinal ruído para um conversor de n-ésima ordem é dada por:

$$SNR(dB) = 6.02 \cdot [n + (l + 0.5)r] + K \qquad \text{(A.29)},$$

na qual l é a ordem do filtro e K é uma constante.

Uma outra alternativa para ampliar a faixa de aplicações de um modulador de 1ª ordem é utilizar, ao invés de um quantizador monobit, um quantizador multibit. Um exemplo de um modulador desse tipo é o modulador A/D incremental que possui estrutura semelhante a um modulador A/D Sigma-Delta de 1ª ordem, exceto por possuir um quantizador de *8 bits* [37]. Assim, sinais de entrada com largura de banda mais elevada podem ser convertidos. O maior problema dessas estruturas é a complexidade inerente de realização.

A.3.1.2. Bloco de Processamento Digital

O bloco de processamento digital, colocado na saída do modulador em um conversor A/D Σ-Δ monobit de 1ª ordem, deve ser capaz de realizar as seguintes funções:

- Retirar sinais espúrios presentes no sinal quantizado da saída do modulador. Esses sinais espúrios são formados basicamente pelo ruído de quantização, presente em altas freqüências, e por componentes fora da banda, presentes em freqüências um pouco superiores à largura de banda do sinal de entrada.
- Reamostrar o sinal da saída do modulador para a freqüência de Nyquist.

Sendo assim, dois sub-blocos compõe esse bloco de processamento digital: um filtro digital passa-baixas e um decimador (estrutura que permite reduzir a freqüência de sobrea-

mostragem ou que realiza o *down-sampling* do sinal digital da saída do modulador). Essa composição é mostrada na Figura A.16.

Nessa figura, um sinal de entrada digital *x[n]*, cujas amostras possuem uma freqüência de amostragem igual à freqüência de sobreamostragem (f_{os}), é submetido a um processo de filtragem passa-baixas (representado pelo sub-bloco *h[n]*), cuja freqüência de corte é a freqüência de Nyquist do sinal de entrada analógica (f_B). O sinal filtrado, *w[n]*, é então decimado (representado pelo sub-bloco $\downarrow N$), ou seja, a freqüência das amostras é reduzida por um fator N ($N = f_{os}/f_B$) para a freqüência de Nyquist do sinal de entrada analógica (f_B). O sinal resultante *y(m)* é um sinal digital, mas com uma freqüência de amostragem N vezes menor que a freqüência do sinal *x[n]*.

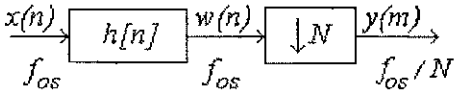


Figura A.16 – Blocos componentes de um decimador digital, com fator de decimação N

3.4. Conclusão

Nesse capítulo foram apresentadas definições, características, e estruturas de realização de conversores A/D convencionais, sobreamostrados e, especialmente, Sigma-Delta. Enfatizou-se a análise da estrutura de um conversor A/D Σ-Δ de 1ª ordem e seus blocos constituintes.

Atualmente, algumas empresas fornecem ASICs que realizam a conversão A/D Sigma-Delta e, na maioria dos casos, o estágio de decimação digital está presente. A Tabela A.1 apresenta exemplos desses ASICs, fabricados pela TEXAS INSTRUMENTS, ANALOG DEVICES e BURR-BROWN.

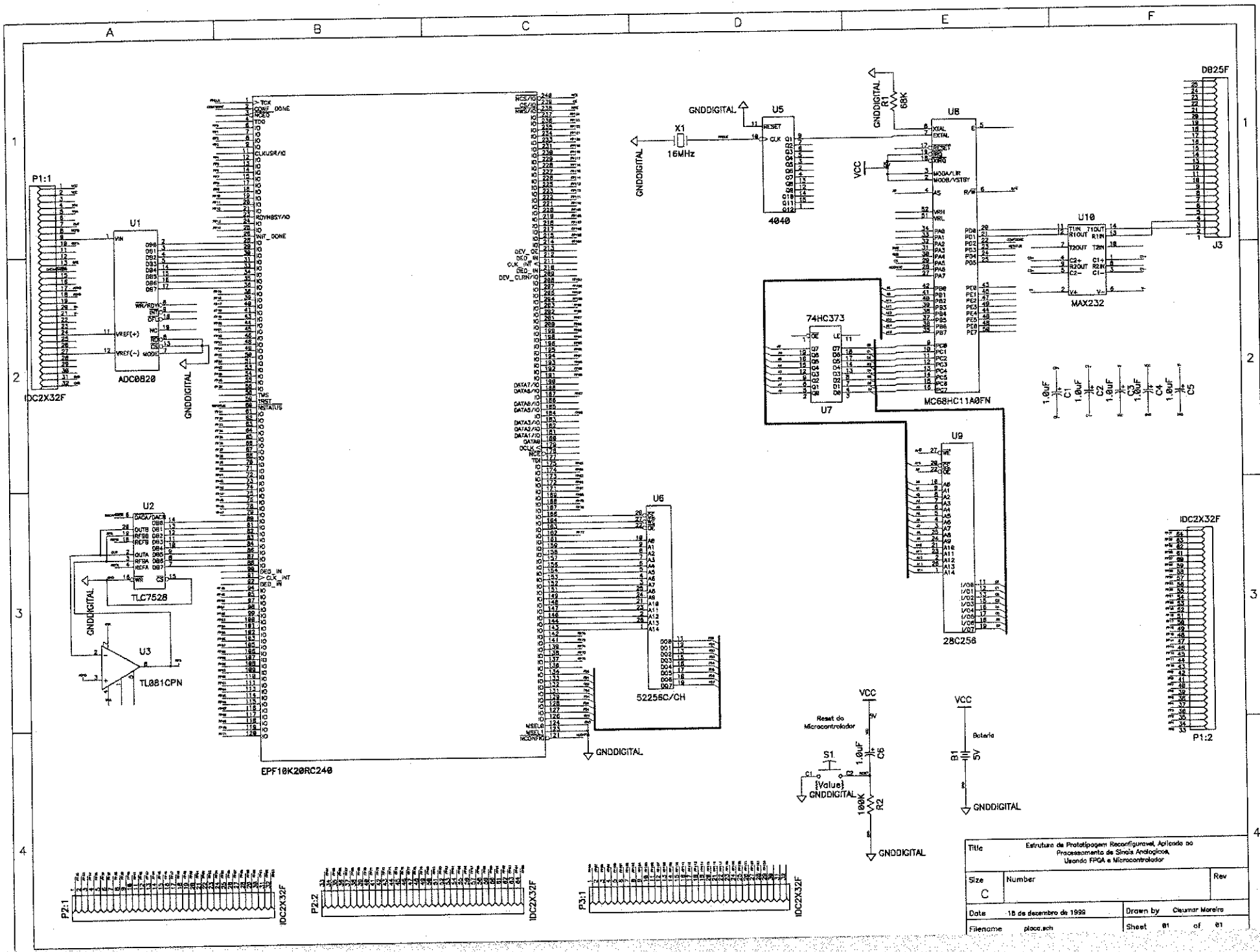
O campo de aplicações dos conversores A/D Sigma-Delta vai, a cada dia, sendo expandido, devido às vantagens oferecidas em termos de simplicidade do ponto de vista da seção analógica e obtenção de altas resoluções com quantizadores de poucos *bits*. Um problema está na relação entre a largura de banda do sinal de entrada e a resolução desejada para o con-

versor, levando esses conversores a serem aplicados em freqüências baixas e médias, embora esse cenário esteja se modificando hoje em dia.

Empresa	Características
TEXAS INSTRUMENTS (TLC320AD58C)	Conversor A/D Sigma-Delta *Stereo*, com resolução de *18 bits*, relação sinal-ruído de *100 dB* e faixa dinâmica de *95 dB*
ANALOG DEVICES (AD7701)	Conversor A/D Sigma-Delta de 2ª ordem, com resolução de *16 bits* e consumo de potência de *10 μW*
BURR-BROWN (ADS1201)	Modulador A/D Sigma-Delta de 2ª ordem, com 130 dB de faixa dinâmica e pode atingir resolução de 24 bits.

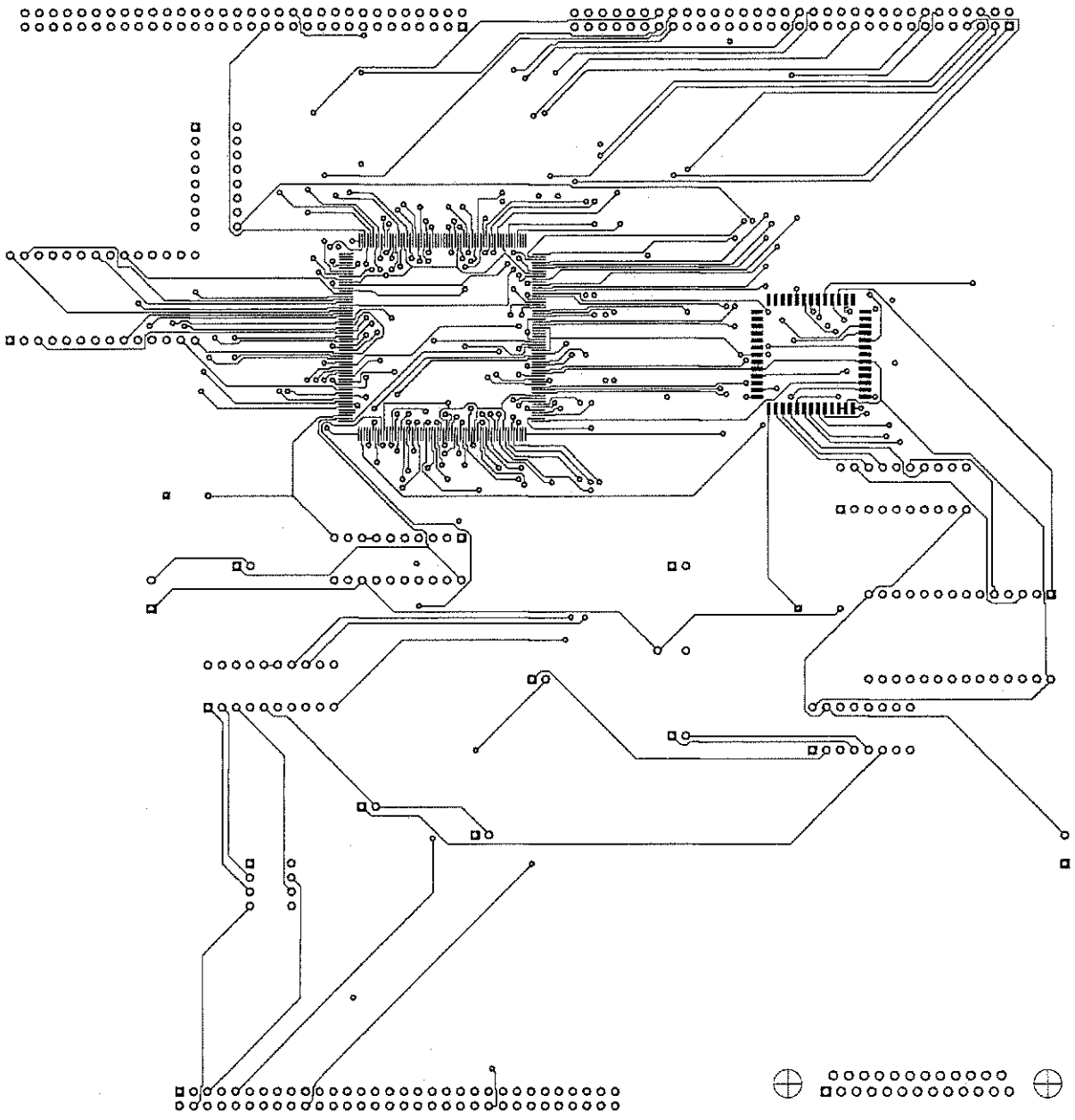
Tabela A.1 – Exemplos comerciais de conversores e modulador A/D Sigma-Delta

Apêndice B1: Esquemático da Placa Reconfigurável


A
B
C
D
E
F
1
2
3
4
P1:1
IDC2X32F
U1
ADC0820
GNDDIGITAL
EPF10K20RC240
U2
TLC7528
U3
TL081CPN
U6
52256C/CH
X1
16MHz
U5
4040
R1
68K
VCC
U8
MC68HC11A0FN
74HC373
U7
U10
MAX232
DB25F
J3
U9
28C256
C1
C2
C3
C4
C5
1.0uF
Reset do Microcontrolador
S1
C6
R2
100K
Bateria
B1
5V
P1:2
P2:1
P2:2
P3:1
Title
Estrutura de Prototipagem Reconfiguravel, Aplicada ao Processamento de Sinais Analogicos Usando FPGA e Microcontrolador
Size
C
Number
Rev
Date
18 de dezembro de 1999
Drawn by
Cleumar Moreira
Filename
placa.sch
Sheet
01
of
01

Apêndice B2: Leiaute da Placa Reconfigurável (Camada Superior – *Top Layer*)

Camada Superior ("Top Layer")



Apêndice B3: Leiaute da Placa Reconfigurável (Camada Inferior – *Bottom Layer*)

Apêndice C

Programas em Verilog HDL

1 – Filtro FIR de 8 *taps*

```
// Programx: FILTRO FIR CAUSAL SIMETRICO de 8 taps
// Estrutura na forma direta sem mecanismo de otimizacao
// Autor: Cleumar da Silva Moreira

module filter1(clk, dado, PCA, PCRD, PCWR, SBHE, ENF1, PCD, M16, PA, J2, J4);

// Declaracao das portas

input clk, PCRD, PCWR, SBHE, ENF1;
input [14:0]PCA;
input [7:0]PA;
output M16;
output [34:3]J2;
output [26:15]J4;
inout [15:0]PCD;
reg [15:0]mem;
tri [15:0]PCD;
input [7:0]dado;

// Lista de variaveis temporarias a serem utilizadas no sistema FIR projetado
reg [7:0]saida;
reg [7:0]s1;
reg [7:0]s2;
reg [7:0]s3;
reg [7:0]s4;
reg [7:0]s5;
reg [7:0]s6;
reg [7:0]s7;

wire ena = (!PCWR) && (!ENF1) && (PCA==0);
wire M16 = SBHE|ENF1;

wire [15:0] q;

lpm_counter contador (.clock(clk), .q(q));
defparam contador.lpm_width = 16;
defparam contador.lpm_modulus = 65536;

wire clk_a = q[10]; // Definicao da frequencia de amostragem

// Implementacao dos "taps" do filtro
always @(posedge clk_a)
begin
saida <= dado;
s1 <= saida;
s2 <= s1;
s3 <= s2;
s4 <= s3;
s5 <= s4;
s6 <= s5;
s7 <= s6;
end

// Obtencao dos termos-soma na resposta ao impulso do filtro FIR
wire [8:0]soma1 = saida + s7;
wire [8:0]soma2 = s1 + s6;
wire [8:0]soma3 = s2 + s5;
wire [8:0]soma4 = s3 + s4;

`include "fir.dat" // Coeficientes do Filtro FIR
// Cálculo da saida y do filtro FIR

wire [16:0]y = -(h1*soma1) - (h2*soma2) + (h3*soma3) + (h4*soma4);
```

```
// Truncamento da variavel de saida para 8 bits
wire [7:0]yt = (y>>7) + (y[6]);

always @(posedge clk_a)
begin
if (ena)
mem <= PCD;
end

wire [26:15]J4 = {mem[0], mem[1], yt, 1'b0, 1'b0};
wire [34:3]J2 = {PCWR, 1'b0, !SBHE, PCRD, (PCA==0), ENF1, 1'b0, ((!PCRD) && (ENF1)),
16'b0, PA};

lpm_bustri tribus (.data(mem), .enabledt((!PCRD) && (!ENF1)), .tridata(PCD));
defparam tribus.lpm_width = 16;

endmodule
```

2 – Filtro IIR de 2ª Ordem

2.1 – Sub-módulo

```
// Sub-modulo do programa do filtro IIR de 2a. ordem
// Autor: Cleumar da Silva Moreira

module iir2t (clk,clk_a,x,saida,n01,n11,d01,d11);

input clk;
input [7:0] x;
input [7:0]saida;
output clk_a;
output [7:0] n01;
output [7:0] n11;
output [7:0] d01;
output [7:0] d11;

// Lista de variaveis temporarias a serem utilizadas no sistema IIR
reg [7:0]n01;
reg [7:0]n11;
reg [7:0]d01;
reg [7:0]d11;

/*reg [7:0]n02;
reg [7:0]n12;
reg [7:0]d02;
reg [7:0]d12;

reg [7:0]n03;
reg [7:0]n13;
reg [7:0]d03;
reg [7:0]d13;

reg [7:0]n04;
reg [7:0]n14;
reg [7:0]d04;
reg [7:0]d14;*/

wire [15:0] q;

lpm_counter contador (.clock(clk), .q(q));
        defparam contador.lpm_width = 16;
        defparam contador.lpm_modulus = 65536;

wire clk_a = q[10]; // Definicao da frequencia de amostragem

// Coeficientes do filtro

// Implementacao dos "taps" do filtro
7always @(posedge clk_a)
begin
        n01 <= x;           // Variavel de entrada
        n11 <= n01;

        d01 <= saida;       // Variavel de saida
        d11 <= d01;
end

endmodule
```

2.2 – Módulo Principal

```
// Modulo principal: FILTRO IIR - Realizacao na forma direta
// Autor: Cleumar da Silva Moreira

`include "iir2t.v"

module iir2(clk, dado, PCA, PCRD, PCWR, SBHE, ENF1, PCD, M16, PA, J2, J4);

// Declaracao das portas

input clk, PCRD, PCWR, SBHE, ENF1;
input [14:0]PCA;
input [7:0]PA;
output M16;
output [34:3]J2;
output [26:15]J4;
inout [15:0]PCD;
reg [15:0]mem;
tri [15:0]PCD;
input [7:0]dado;

wire ena = (!PCWR) && (!ENF1) && (PCA==0);
wire M16 = SBHE|ENF1;

`include "iir.dat"

iir2t i1 (.clk(clk),.clk_a(clk_a),.x(dado),.saida(saida),.n01(n01)
        ,.n11(n11),.d01(d01),.d11(d11));

// Geracao da saida
wire [16:0] h = a0*dado + (a1*n01) - (b1*d01) + (a2*n11) - (b2*d11);

// Truncamento e "escalamento" da saida
wire [7:0] y = (h>>6) + h[5];

always @(posedge clk_a)
            begin
            if (ena)
                        mem <= PCD;
            end

wire [26:15]J4 = {mem[0], mem[1], y, 1'b0, 1'b0};
wire [34:3]J2 = {PCWR, 1'b0, !SBHE, PCRD, (PCA==0), ENF1, 1'b0, ((!PCRD) && (ENF1)),
                                                        16'b0, PA};

lpm_bustri tribus (.data(mem), .enabledt((!PCRD) && (!ENF1)), .tridata(PCD));
                        defparam tribus.lpm_width = 16;

endmodule
```

3 – Filtro Decimador

3.1 – Sub-módulo

```
//                      ******************************************
//                      * Sub-modulo: Conversor Serie-Paralelo    *
//                      * Autor: Cleumar da Silva Moreira        *
//                      * Trabalho de Dissertacao                *
//                      * Data: 05/05/1999                       *
//                      * UFPB - Campina Grande - PB             *
//                      ******************************************

module shifreg (clksa,entrada,dado);

input clksa;
input entrada; // entrada serial
output [7:0] dado; // saida paralela

reg e0,e1,e2,e3,e4,e5,e6,e7; // variaveis temporarias contendo
                                                // os valores dos bits da conversao serie-paralelo

// Descricao RTL do Conversor Serie-Paralelo

always @(posedge clksa)
begin
            e0 <= entrada;
            e1 <= e0;
            e2 <= e1;
```

```
            e3 <= e2;
            e4 <= e3;
            e5 <= e4;
            e6 <= e5;
            e7 <= e6;
end
wire [7:0] dado = {e7,e6,e5,e4,e3,e2,e1,e0};  // variavel de saida
endmodule
```

3.2 – Módulo Principal

```
//                  ***************************************************************
//                  * Programa: Filtros Passa-baixa e DOWN-SAMPLE de              *
//                  *              um Conversor A/D Sigma-Delta de 1a. ordem      *
//                  * Trabalho de Dissertacao                                     *
//                  * Autor: Cleumar da Silva Moreira                             *
//                  * UFPB - Campina Grande - PB                                 *
//                  * Data: 05/05/1999                                           *
//                  ***************************************************************

`include "shifreg.v"    // Chamada do sub-modulo registrador de deslocamento
                        // que converte o dado serial da saida do
                        // modulador A/D Sigma-Delta de 1a. ordem

module decim (clksa,clkdec,entrada,PCA,PCRD,PCWR,SBHE,ENF1,PCD,M16,PA,J2,J4);

// Declaracao das portas

input clksa,clkdec,PCRD,PCWR,SBHE,ENF1;
input [14:0]PCA;
input [7:0]PA;
output M16;
output [34:3]J2;
output [26:15]J4;
inout [15:0]PCD;
reg [15:0]mem;
tri [15:0]PCD;
input entrada;

// Lista de variaveis temporarias a serem utilizadas no sistema FIR projetado
reg [7:0]s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,
      s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31;

// Variavel de saida do decimador
reg [7:0]saida;

// Chamada do sub-modulo que realiza o conversor serie-paralelo
shifreg c1 (.clksa(clksa), .entrada(entrada), .dado(dado));

wire ena = (!PCWR) && (!ENF1) && (PCA==0);
wire M16 = SBHE|ENF1;

// Implementacao dos 32 "taps" do filtro passa-baixa tipo FIR linear e simetrico
always @(posedge clksa)
begin
            s0 <= dado;
            s1 <= s0;
            s2 <= s1;
            s3 <= s2;
            s4 <= s3;
            s5 <= s4;
            s6 <= s5;
            s7 <= s6;
            s8 <= s7;
            s9 <= s8;
            s10 <= s9;
            s11 <= s10;
            s12 <= s11;
            s13 <= s12;
            s14 <= s13;
            s15 <= s14;
            s16 <= s15;
            s17 <= s16;
            s18 <= s17;
            s19 <= s18;
            s20 <= s19;
            s21 <= s20;
            s22 <= s21;
            s23 <= s22;
            s24 <= s23;
            s25 <= s24;
            s26 <= s25;
            s27 <= s26;
            s28 <= s27;
            s29 <= s28;
            s30 <= s29;
            s31 <= s30;
```

```
// Obtencao dos termos-soma na resposta ao impulso do filtro FIR
wire [8:0]soma1 = s0 + s31;
wire [8:0]soma2 = s1 + s30;
wire [8:0]soma3 = s2 + s29;
wire [8:0]soma4 = s3 + s28;
wire [8:0]soma5 = s4 + s27;
wire [8:0]soma6 = s5 + s26;
wire [8:0]soma7 = s6 + s25;
wire [8:0]soma8 = s7 + s24;
wire [8:0]soma9 = s8 + s23;
wire [8:0]soma10 = s9 + s22;
wire [8:0]soma11 = s10 + s21;
wire [8:0]soma12 = s11 + s20;
wire [8:0]soma13 = s12 + s19;
wire [8:0]soma14 = s13 + s18;
wire [8:0]soma15 = s14 + s17;
wire [8:0]soma16 = s15 + s16;

`include "decim.dat"  // Coeficientes do Filtro FIR

// Calculo da saida y do filtro FIR

wire [16:0]y = +(h1*soma1)+(h2*soma2)+(h3*soma3)+(h4*soma4)+(h5*soma5)
          +(h6*soma6)+(h7*soma7)+(h8*soma8)+(h9*soma9)+(h10*soma10)+(h11*soma11)
          +(h12*soma12)+(h13*soma13)+(h14*soma14)+(h15*soma15)+(h16*soma16);

wire [7:0]yt = (y); // Truncamento para 8 bits e divisao do valor da variavel de saida

// Decimacao do sinal filtrado, no qual clkdec=2*fb e um sinal externo
always @(posedge clkdec)
          begin
                    saida <= yt;
                    if (ena)
                              mem <= PCD;
          end

// Define os sinais a serem postos no conector da placa
wire [26:15]J4 = {mem[0], mem[1], 1'b0, 1'b0, saida};
wire [34:3]J2 = {PCWR, 1'b0, !SBHE, PCRD, (PCA==0), ENF1, 1'b0, ((!PCRD) && (ENF1)),
                                        16'b0, PA};

lpm_bustri tribus (.data(mem), .enabledt((!PCRD) && (!ENF1)), .tridata(PCD));
                    defparam tribus.lpm_width = 16;

endmodule
```

Referências Bibliográficas

[1] Geppert, L. *The 100-Million Transistor IC*. IEEE Spectrum, pp. 22-24, July 1999.

[2] Chappell; B. *The Fine Art of IC Design*. IEEE Spectrum, pp. 30-34, July 1999.

[3] Faggin, f. The Future of the Microprocessor. 1999.

[4] Cardoso, J, M. P. and Vestias, M. P. *Architectures and Compilers to Support Reconfigurable Computing*. http://www.acm.org/crossroads/xrds5-3/reconcept.html. 1999.

[5] http://www.xess.com/FPGA/ho05001.html. XS95 Board. December 1998.

[6] Lima, M. E., Cavalcante, S. V. *CHAMELEON: A Prototyping Platform for Digital System Design*. ICMP98 – International Conference on Microelectronics and Packaging. 1998.

[7] http://www.brightstareng.com/ine1.htm. *ipEngine1*. Bright Star Engineering. 1998.

[8] http://www.sci.fi/~cubase/board.html. *Hedgehog, a General Purpose Computer Board*. 1997.

[9] Morais, M. R. A. *Sistema de Prototipagem Rápida Dirigido ao Processamento de Imagens*. Dissertação de Mestrado. Universidade Federal da Paraíba. 1998.

[10] Weste, N. H. E. and Eshraghian, K. *Principles of CMOS VLSI Design: A System Perspective*. Addison-Wesley Publishing Company. 2nd Edition. 1993.

[11] Smith, M. J. S. *Application Specific Integrated Circuits. Addison-Wesley Publishing Company*. 6th edition, April 1999

[12] Dewey, A. *Analysis and Design of Digital Systems with VHDL*. International Thomson Publishing. 1997.

[13] Rose, J., El Gamal, A., and Sangiovanni-Vicentelli, A. *Architecture of Field-Programmable Gate Arrays*. Proceedings of the IEEE, vol. 81, no. 7, July 1993.

[14] Chan, P. K. and Mourad, S. *Digital Design Using Field Programmable Gate Arrays*. Prentice Hall, Inc. Englewood Cliffs, New Jersey. 1994.

[15] Wilton, S. *Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*. Doctor's Thesis, University of Toronto, 1997.

[16] Kaviani, A., Brown, S. (1999). *The Hybrid Field-Programmable Architecture*. IEEE Design & Test of Computers. 1999.

[17] Knapp, S. K. *Using Programmable Logic to Accelerate DSP Functions*. Application Note. XILINX, Inc. 1995.

[18] Chapman, K. *Dynamic Microcontroller in an XC4000 FPGA*. Application Note. XILINX, Inc. 1994.

[19] Altera Corporation. *Implementing fft with On-Chip RAM in FLEX 10K Devices*. Application Note 84.

[20] Tzou, Y., Hsu, H. *FPGA Realization of Space-Vector PWM Control IC for Three-Phase PWM Inverters*. IEEE Transactions on Power Electronics, vol. 12, no.6, 1997.

[21] Moreira, C. S., Freire, R. C. S., Melcher, E. U. K., Deep, G. S., Catunda, S. Y., and Alves, R. N. C. *FPGA-Based SVPWM Trigger Generator for a 3ϕ Voltage Source Inverter*. Accepted for publication in the Instrumentation Measurement Technology Conference 2000 (IMTC2000).

[22] De Micheli, G. and Gupta, R. K. *Hardware/Software Co-Design*. Proceedings of the IEEE, vol.85, no. 3, march 1997.

[23] Wolf, W. H. *Hardware-Software Co-Design of Embedded Systems*. Proceedings of the IEEE, vol. 82, no. 87. July 1994.

[24] Lima, J. A. G., Melcher, E. U. K. e Cavalcanti, A. C. (1999) MCA: A Single Chip one Port Scalable ATM Controller. *In: Simpósio Brasileiro de Concepção de Circuitos Integrados*, 1999, Natal. Anais do XII Simpósio Brasileiro de Concepção de Circuitos Integrados.

[25] http://www.eecg.toronto.edu/EECG/RESEARCH/FPGA.html. *FPGA Research at University of Toronto*. 1996.

[26] http://www.sidsa.es/fipsoc.htm. *FIPSOC: Field Programmable System on a Chip Family*. June 1998.

[27] http://www.xilinx.co.jp/prs_rs/0_18process.htm. *Xilinx, UMC Group Set New 1 GHz Perfomance Milestone with 0.18 Micron, 1.8 Volt Prototype FPGA*. 1999.

[28] http:\\www.altera.com: *Homepage* da Altera:

[29] *HC11 – M68HC11 Reference Manual*. MOTOROLA Inc. 1990.

[30] ADC0820 8-Bit High Speed µP Compatible A/D Converter with Track/Hold Function (1997). *National Data Acquisition Databook*, National Semiconductors.

[31] AD7528 Dual 8-bit Multiplying Digital-to-Analog Converters (1995). Analog Devices.

[32] Ifeachor, E. C. and Jervis, A. W. *Digital Signal Processing: A Practical Approach*. Addison-Wesley Publishing Company, 1993.

[33] Rabiner, L. R. and Gold, B. *Theory and Application of Digital Signal Processing*. Prentice-Hall, Inc, 1975.

[34] Graeme, J. G., Tobey, G. E. and Huelsman, L. P. *Operational Amplifiers: Design and Applications*. McGraw-Hill International Editions, 1986.

[35] Ismail, M and Fiez, T. *Analog VLSI: Signal and Information Processing*. McGraw-Hill, Inc., 1994.

[36] Aziz, P. M., Sorensen, H. V. e Van Der Spiegel, J. An Overview of Sigma-Delta Converters: How a 1-bit ADC achieves more than 16-bit resolution. IEEE Signal Processing Magazine, pp. 61-84, 1996.

[37] Candy, J. C. e Temes, G. C. *Oversampling Delta-Sigma Data Converters: Theory, Design and Simulation*. IEEE Press. 1992.

[38] Thompson, R. S. *Digital audio Technology, A Primer: Part II.Center for Audio Recording Arts*. 1999.

[39] Louis, J, Abcarius, J. e Roberts, G. W. *An Eighth-Order Bandpass ΔΣ Modulator for A/D Conversion in Digital Radio*. IEEE Journal of Solid-State Circuits, vol. 34, no. 4, April 1999.