UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um Microprocessador com Capacidades Analógicas

por

FLÁVIO LUIZ DE OLIVEIRA ZIMMERMANN

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Altamiro Amadeu Susin
Orientador

Porto Alegre, março de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Zimmermann, Flávio Luiz de Oliveira

Um Microprocessador com Capacidades Analógicas / por Flávio Luiz de Oliveira Zimmermann. – Porto Alegre: PPGC da UFRGS, 2002.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, RS - BR, 2002. Orientador: Susin, Altamiro A.

1. Sinais Mistos. 2. CORE. 3. Teste de Hardware 4. Geração de sinais 5. Risco 6. BIST 7. Sigma-delta I. Susin, Altamiro Amadeu. II. Título.

Dedico essa dissertação à memória de meu avô,
Prof. José Rodrigues de Oliveira.

Agradecimentos

Agradeço a meus pais e familiares que me deram suporte financeiro, moral e emocional para a execução desse trabalho.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro dado no último semestre de mestrado.

Agradeço a meu orientador, Altamiro Amadeu Susin, por todo o incentivo, ajuda prestados e por sua amizade durante momentos difíceis.

Agradeço aos outros professores do PPGC por seu apoio e por estarem sempre disponíveis para solucionar dúvidas ou mesmo para simplesmente conversar.

Agradeço ao professor Luigi Carro por, além de ser um grande amigo, também ser um grande professor e por todas as cobranças de trabalhos que ele manda fazer.

Agradeço aos funcionários do Instituto, em especial à Eliane (portaria dos laboratórios) por toda a ajuda, eficiência em resolver os problemas e amizade.

Sumário

Lista de Abreviaturas ou Siglas

ABILBO	Analog Built-In Logic Block Observer
A/D	Analógico / Digital
ADC	Conversor A/D
AMS	Austria Mikro Systeme International AG
BIST	Built-In Self Test
D/A	Digital / Analógico
DAC	Conversor D/A
DC	Corrente Direta
DDFS	Direct Digital Frequency Synthesis
DSP	Digital Signal Processing
fig.	Figura
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
LC	Indutância - Capacitância
LPM	Library of Parameterized Modules
LSB	Least Significant Bit
MADBIST	Mixed Analog-Digital BIST
MSB	Most Significant Bit
MSI	Medium Scale Integration
Op-amp	Operational Amplifier
PC	Parte de Controle
PLL	Phase-Locked Loop
PO	Parte Operativa
RAM	Random Access Memory
RC	Resistor-Capacitor
ROM	Read Only Memory
RMS	Root Mean Square
SC	Capacitor Chaveado
SNR	Signal to Noise Ratio
SRAM	Static RAM
UC	Unidade de Constante
UD	Unidade de Deslocamento
ULA	Unidade Lógica-Aritmética
UPC	Unidade Contador de Programa
VHDL	Very High Speed Integrated Circuit HDL
VLSI	Very Large Scale Integration

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta um estudo, implementação e simulação de geradores de sinais analógicos usando-se circuitos digitais, em forma de CORE, integrando-se este com o microprocessador Risco. As principais características procuradas no gerador de sinais são: facilidade de implementação em silício, programabilidade tanto em freqüência quanto em amplitude, qualidade do sinal e facilidade de integração com um microprocessador genérico.

Foi feito um estudo sobre a geração convencional de sinais analógicos, dando-se ênfase em alguns tipos específicos de circuitos como circuitos osciladores sintonizados, multivibradores, geradores de sinais triangulares e síntese de freqüência digital direta.

Foi feito também um estudo sobre conversão digital-analógica, onde foram mostrados alguns tipos básicos de conversores D/A. Além disso foram abordadas questões como a precisão desses conversores, tipos digitais de conversores digital-analógico, circuitos geradores de sinais e as fontes mais comuns de erros na conversão D/A.

Dando-se ênfase a um tipo específico de conversor D/A, o qual foi utilizado nesse trabalho, abordou-se a questão da conversão sigma-delta, concentrando-se principalmente no ciclo de formatação de ruído. Dentro desse assunto foram abordados o laço sigma-delta, as estruturas de realimentação do erro, estruturas em cascata, e também o laço quantizador.

Foram abordados vários circuitos digitais capazes de gerar sinais analógicos, principalmente senóides. Além de geradores de senóides simples, também se abordou a geração de sinais multi-tom, geração de outros tipos de sinais baseando-se no gerador de senóides e também foi apresentado um gerador de funções.

Foram mostradas implementações e resultados dessas. Iniciando-se pelo microprocessador Risco, depois o gerador de sinais, o teste deste, a integração do microprocessador com o gerador de sinais e finalmente a implementação standard-cell do leiaute desse sistema.

Por fim foram apresentadas conclusões, comentários e sugestões de trabalhos futuros baseando-se no que foi visto e implementado nesse trabalho.

Palavras-Chave: Sinais Mistos, CORE, Teste de Hardware, Geração de sinais, Risco, BIST, Sigma-delta.

TITLE: “A MICROPROCESSOR WITH ANALOG CAPABILITIES”

Abstract

This work presents a study, simulations and an implementation of analog signal generators through the use of digital circuits, as a CORE, integrating this with the Risco Microprocessor. The main characteristics wanted from the signal generator are: ease of implementation on silicon, programmability in frequency and amplitude, quality of the signal, ease of integration with a general purpose processor.

A review on conventional analog signal generation was made. Some emphasis was given to specific types of circuits such as tuned oscillator circuits, multi vibrators, triangular signal generator and direct digital frequency synthesis.

Also a study on digital to analog conversion was made, where some basic types of D/A converters were shown. Some aspects as the precision of these converters, digital D/A converters, signal generator circuits and the most common sources of errors in D/A conversion.

Giving emphasis on a specific D/A converter, which was used on this work, the sigma-delta conversion was looked upon, concentrating mainly on the noise-shaping loop. On this subject the sigma-delta loop, error feedback structures, cascaded structures and quantization loop were given special attention.

Several digital circuits capable of generating analog signal, especially sine wave, were studied. Also the generation of multi tone signals, generation of other types of signals based on the sine wave generator and a special function generator.

The implementation and results were shown, starting with the Risco microprocessor, then the signal generator, its test, and the integration with the microprocessor and finally a standard-cell implementation of this system.

At last some conclusions, comments and future work suggestions, based on the results of the experiments and implemented circuits on this work, were drawn.

Keywords: Mixed signal, CORE, Hardware test, Signal generation, Risco, BIST, Sigma-delta.

1 Introdução

A proposta desse trabalho é desenvolver um circuito digital, que possa ser usado como CORE e integrado a um processador, gerador de sinais analógicos. Com o objetivo de conseguir assim um sinal analógico confiável e programável quanto a freqüência e amplitude. Usando-se componentes digitais para facilitar, assim, a integração do mesmo no silício e o teste do mesmo através de técnicas de teste puramente digitais.

Hoje em dia, a maioria dos projetistas tende a colocar partes analógicas e digitais em um mesmo *chip*, além de ser muito comum o uso de componentes analógicos em placas junto com componentes digitais. Isso proporciona um melhor desempenho ao sistema eletrônico e ao produto final pelo seu custo reduzido e menor consumo de energia. Além de proporcionar melhor desempenho ao circuito, essa união de circuitos digitais com circuitos analógicos permite a implementação de novas funções no *chip* aumentando assim do valor agregado do circuito. Como novas funções são adicionadas no *chip*, menos componentes são necessários na placa. Considerando-se que o custo da placa é um fator que deve ser considerado quando se desenvolve circuitos, reduzindo a complexidade da placa, o custo desta fica reduzido aumentando-se assim a competitividade do circuito implementado num mercado altamente agressivo. Novos problemas surgem com essa integração, sendo um deles o teste de circuitos mistos. O custo de produção de tais sistemas pode ser altamente influenciado pelo custo do teste destes, especialmente de suas partes analógicas. [COT97]

Por isso, a maioria dos projetistas passaram a incorporar o teste no circuito criando o BIST [KIM88] [ROB96]. Essa técnica é muito usada para circuitos digitais. Para circuitos mistos são usadas vários métodos *ad-hoc* e técnicas conhecidas como MADBIST [ROB95] [HAU98] [TON95] [ROB96] e ABILBO [PAW96] [LUB96] [WES93]. Com a aprovação do padrão IEEE 1149.4 para barramentos de sinais mistos de teste, as estratégias de teste serão ainda mais difundidas. Através da geração de sinais analógicos na freqüência necessária e análise dos sinais recebidos [REN97] [PAP99], pode-se realizar o teste dos componentes analógicos que estejam na mesma placa ou no mesmo *chip*, dependendo do caso [COT97].

Para testar os componentes analógicos que estão em uma placa é necessário que se tenha um sinal analógico confiável e facilmente configurável quanto à freqüência e à amplitude. O uso de componentes analógicos para gerar esse sinal é muito complexo pois os componentes analógicos usados para gerar sinais de teste podem estar eles mesmos com falhas ou serem imprecisos.

Esse trabalho irá em primeiro lugar apresentar vários circuitos capazes de realizar a geração de formas de onda analógicas. Esses circuitos, amplamente usados e conhecidos são na sua maioria circuitos analógicos, o que acaba dispensando o uso de conversores digitais-analógicos. Como esses circuitos utilizam-se de vários componentes analógicos, inclusive alguns deles apresentam cristais, seus sinais não são programáveis e nem confiáveis, devido a natureza de seus componentes.

Após isso, serão apresentados circuitos conversores digitais-analógicos. Como o objetivo desse trabalho é implementar circuitos digitais para geração de sinais analógicos, esses serão necessários para realizar a conversão da saída digital em um sinal analógico. Alguns circuitos conversores são na verdade circuitos sinais mistos, ou seja, tem partes digitais e partes analógicas. Outros são circuitos onde as entradas digitais simplesmente controlam chaves nos circuitos analógicos. Os circuitos sinais

mistos são os que mais interessam nesse trabalho, pois quanto menor a parte analógica necessária mais facilmente programáveis eles podem ser e mais confiáveis serão os sinais.

No quarto capítulo, serão discutidos circuitos conversores D/A sigma-delta. Esses circuitos apresentam uma predominância de componentes digitais, sendo que, a única parte analógica necessária para essa classe de circuitos é um filtro passa-baixa. Assim sendo, esses circuitos não só são de fácil integração como podem ser testados através de técnicas digitais, tendo assim algumas características desejáveis nesse trabalho.

No próximo capítulo, serão discutidos alguns circuitos digitais que são capazes de gerar sinais, assim como os que foram discutidos no capítulo dois, só na sua forma digital. Ou seja, circuitos que geram formas de ondas digitais que podem ser convertidos obtendo-se assim o sinal analógico na forma desejada. Esses circuitos como são digitais atendem as características pedidas de serem programáveis e testáveis através de métodos digitais.

Na sexta parte desse trabalho, será apresentado o CORE processador que foi utilizado na integração e suas várias implementações. Além disso, serão apresentadas as implementações realizadas com alguns dos circuitos discutidos e os resultados obtidos. Essas implementações foram realizadas utilizando-se a linguagem VHDL para a descrição dos componentes digitais. A parte mista do circuito foi implementada, para efeitos de simulação na ferramenta Matlab. Isso porque as ferramentas de VHDL disponíveis não são capazes de realizar uma simulação de sinais mistos, enquanto que usando-se a linguagem própria do Matlab isso se torna possível, podendo-se enxergar a forma de onda resultante após a filtragem analógica.

No último capítulo serão apresentadas as conclusões do trabalho como um todo, apresentando também um relato de algumas dificuldades que surgiram ao executar esse trabalho. Além disso, serão apresentadas algumas sugestões para a resolução desses problemas além de algumas sugestões para o desenvolvimento continuado desse trabalho.

2 Geração convencional de sinais analógicos

Esse capítulo irá apresentar vários circuitos analógicos capazes de realizar a geração de formas de onda analógicas. Esses circuitos, por serem analógicos, dispensam o uso de conversores digitais-analógicos. Além dos circuitos analógicos, também serão apresentados alguns circuitos digitais que, utilizando-se de uma memória com valores pré gravados e um conversor digital-analógico são capazes de gerar formas de onda analógicas.

2.1 Circuitos Osciladores Sintonizados

Circuitos osciladores sintonizados ativos geram formas de onda senoidais estáveis seguindo o comportamento de um circuito passivo (LC, RC, RL ou RLC). Determinadas configurações de circuitos contendo indutores e capacitores tem tendência a oscilar. Um circuito contendo apenas componentes passivos, entretanto, não consegue iniciar nem manter a oscilação. Os circuitos que devem gerar uma oscilação sustentada contém componentes ativos.

Uma configuração popular, conhecida como oscilador Colpitts, é mostrado na fig. 2.1. Esse circuito tem uma freqüência de oscilação dada por:

$$\omega_0 = \left(\frac{1}{L_2} \left(\frac{1}{C_1} + \frac{1}{C_3} \right) \right)^{\frac{1}{2}}$$

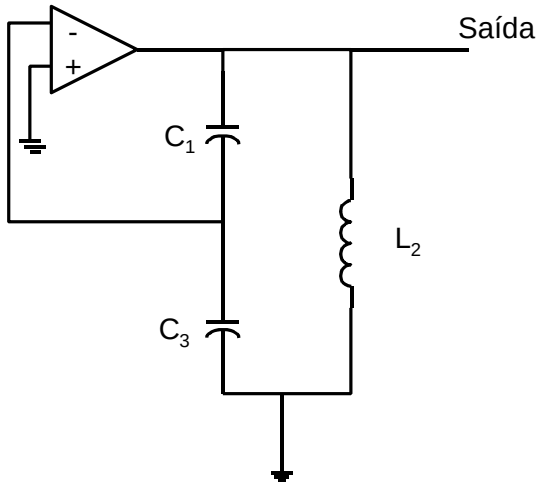


FIGURA 2.1 - Configuração básica de um oscilador Colpitts

Alternativamente, o circuito da fig. 2.1 pode ser modificado para conter um cristal piezelétrico, como na fig. 2.2. O circuito resultante tem uma freqüência muito próxima da freqüência de ressonância do cristal. Entretanto são incompatíveis com o processo de fabricação de circuitos integrados. Além disso, a freqüência de oscilação não é ajustada facilmente.

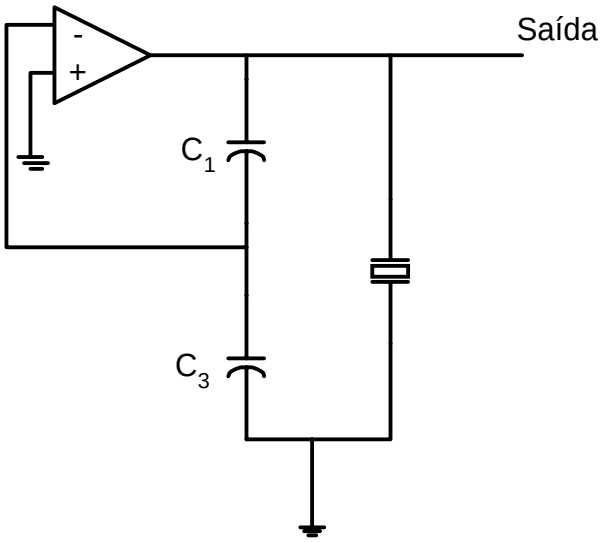


FIGURA 2.2 – Oscilador Colpitts com cristal [ROB95]

2.2 Multivibradores

Também conhecidos como osciladores de relaxação, os multivibradores são apropriados para a implantação de circuitos integrados porque não requerem indutores ou cristais. O espectro das formas de onda geradas por esses circuitos contém muito mais componentes do que outros métodos. Como multivibradores não geram senóides, técnicas para a formatação do sinal devem ser utilizadas para converter a saída do multivibrador (normalmente triangular ou quadrada) para um único tom. Esse processo de conversão envolve operações não lineares que provocam uma distorção harmônica total em torno de 40dB (1%). Esse nível de distorção não é aceitável para MADBIST [ROB95].

2.3 Osciladores a capacitores chaveados

Existem várias estratégias para gerar uma determinada freqüência dentro de circuitos MOS integrados. O método mais simples é usar divisores de freqüência para dividir a freqüência do relógio mestre. Isso resulta em uma boa estabilidade de freqüência, mas estando esta restrita a uma sub-harmônica da freqüência do relógio. Com isso consegue-se gerar um sinal numa certa freqüência, mas por depender de um sinal que vem do relógio com outra freqüência isso não serve como um oscilador.

Outra possibilidade é usar um multivibrador com um controle por voltagem como é um requisito de um *phased-locked loop* (PLL). A freqüência então pode ser controlada incorporando no circuito uma fonte controlada de voltagem. Infelizmente, a fabricação dessa fonte estável e precisa sem o uso de componentes fora do chip é muito difícil e dessa forma a freqüência de oscilação não é estável sendo muito sensível à temperatura e aos parâmetros do processo.

Também é possível simular um oscilador RC ativo, substituindo todos os resistores por caminhos SC (capacitores chaveados). Assim a freqüência pode ser independente da freqüência do relógio, mas mesmo assim o espectro de saída irá conter a freqüência do relógio e freqüências mais altas. Como a freqüência de oscilação depende apenas na razão de capacitâncias, ele pode ser previsível e estável. Um oscilador sensível a parasitas baseado em um oscilador “phase-shift” RC ativo é

ilustrado na fig. 2.3. Assumindo que a freqüência do relógio f_c seja muito maior que a freqüência de oscilação f_o, que o ganho de tensão dc circuito aberto $A = g_m r_d$ do MOSFET é muito maior que um, e que $\alpha << 1$, o valor crítico de g_m necessário para que se tenha oscilação passa a ser

$$g_{mcrit} \cong 12 f_c \alpha\ C\left(1+\frac{14}{A}\right)$$

e a freqüência de oscilação é

$$f_o \cong \frac{\sqrt{3}}{2\Pi}\alpha\ f_c\left(1+\frac{6}{A}\right).$$

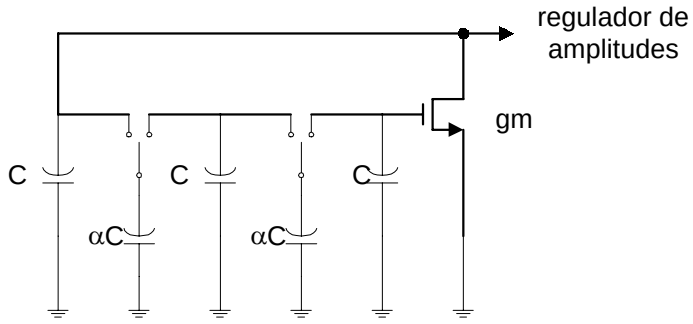


FIGURA 2.3 – Circuito baseado em um oscilador phase shift RC ativo

2.4 Oscilador "Schmitt Trigger"

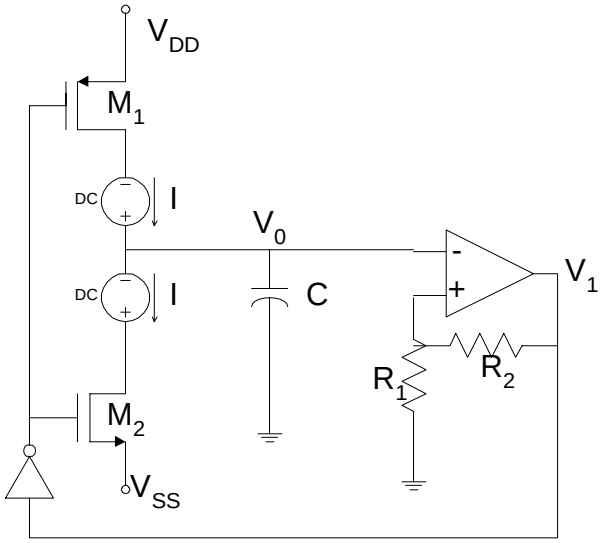


FIGURA 2.4 – Oscilador sem relógio mestre

Um oscilador que não requer um relógio é mostrado na fig. 2.4. O op-amp (usado como comparador) e seus resistores de realimentação R_1 e R_2 formam um "Schmitt Trigger", sua voltagem de saída V_1 controla as chaves M_1 e M_2. Se $V_1=V_{DD}$ então $V_0 < \frac{V_{DD}R_1}{(R_1+R_2)}$. Como V_1 é alto, M_2 está cortado, M_1 está ligado, carregando C e assim

aumentando V_0. Quando V_0 alcançar $\frac{V_{DD}R_1}{R_1+R_2}$ a saída V_1 muda para V_{SS}. Isso causa M_1 a desligar e ligar M_2, descarregando C e reduzindo V_0 até esse atingir $\frac{V_{SS}R_1}{R_1+R_2}$. Fazendo V_{SS}= -V_{DD} a freqüência de oscilação é $f_0 = \frac{\left(1+\frac{R_2}{R_1}\right)I}{4CV_{DD}}$. [GRE86]

2.5 Gerador de ondas triangulares

Um gerador de ondas triangulares com amplitude constante é mostrado na fig. 2.5. Esse circuito fornece uma onda triangular cuja amplitude independe da freqüência.

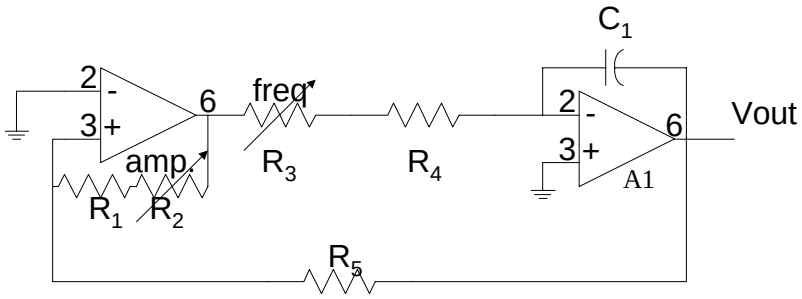


FIGURA 2.5 – Gerador de ondas triangulares com amplitude e freqüência ajustáveis

O gerador contém um integrador que funciona como um gerador de rampa e um detetor de nível. Esse é similar a um Schmitt Trigger por ser um circuito fechado com uma grande zona morta. Essa função é implementada através de uma realimentação positiva em torno de um amplificador operacional. Quando a saída do amplificador for um estado saturado, positivo ou negativo, a realimentação fornece uma voltagem na entrada não inversora que é determinada pela atenuação do laço e voltagem de saturação do amplificador. Para que esse mude de estado, a voltagem na entrada do op-amp deve mudar de polaridade numa amplitude superior à voltagem de offset da entrada do amplificador. [EIM70]

Quando isso ocorrer, o amplificador satura na direção oposta e permanece nesse estado até que a voltagem na sua entrada reverta novamente.

A freqüência triangular é determinada por R_3, R_4 e C_1 e pelas voltagens de saturação positiva e negativa do op-amp A_1. A amplitude é determinada pela razão de R_5 pela combinação de R_1, R_2 e voltagem de saturação do detetor de threshold.

2.6 Geração digital de sinais

Recentes melhoramentos na tecnologia de microeletrônica, como conversão digital-analógico ou analógico-digital, microprocessadores, etc. têm permitido a desenvolvedores de circuitos integrados realizar processamento de sinais analógicos usando técnicas de DSP devido a alta velocidade de processamento e também à crescente densidade dos componentes atuais. Essa mesma filosofia pode ser aplicada à geração de sinais analógicos com resultados excepcionais.

2.7 Síntese Digital Direta de Freqüência

Síntese digital direta de freqüência (DDFS) gera sinais senoidais analógicos usando uma combinação de lógica digital e conversão D/A. Uma arquitetura comum de DDFS é mostrada na fig. 2.6. Aqui, uma palavra de sintonia de freqüência e um acumulador de L bits são usados para gerar o argumento de fase teta-n (θn), onde n é a ordem do valor que está sendo usada na computação, como mostrado na fig. 2.6.

Enquanto circuitos DDFS oferecem uma boa velocidade de chaveamento e boa resolução de freqüência, a função seno, a qual é computacionalmente intensa de ser calculada, deve ser computada usando-se uma tabela *look-up* baseada em ROM. Apesar dessa ROM poder ser minimizada usando-se algoritmos de compressão, as arquiteturas DDFS resultantes ainda são muito grandes para serem consideradas para aplicações de teste, a menos que essa ROM venha a ter poucas palavras. Sendo que, quanto menos palavras existirem na ROM menos preciso é o seno. Assim sendo, se for necessário um sinal muito estável e sem distorção, os algoritmos de compressão não poderiam ser utilizados, sendo necessário assim, uma ROM muito grande.

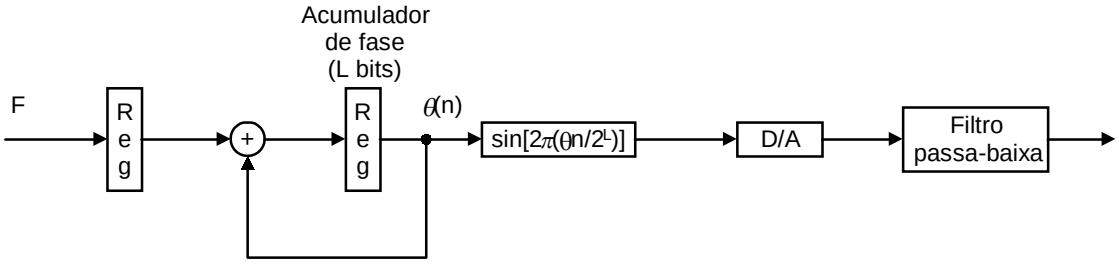


FIGURA 2.6 - Uma arquitetura DDFS comum

Esse capítulo apresentou várias arquiteturas clássicas para geração de sinais analógicos. As primeiras arquiteturas aqui mostradas são circuitos analógicos, sendo que algumas delas fazem uso de cristais. Infelizmente esses circuitos não atingem as metas do trabalho, pois além da integração de circuitos analógicos, com ou sem cristais, não ser tão simples quanto a integração de circuitos digitais ainda existe o problema desses não serem facilmente programáveis quanto a freqüência e amplitudes que serão geradas.

No final deste capítulo foi apresentado uma alternativa para o uso de circuitos analógicos que é a realização do oscilador através de circuitos digitais, passando o sinal gerado por um conversor D/A. Essa idéia será aproveitada, mas com uma arquitetura diferente da apresentada, isso porque esta gera o sinal através de valores gravados em uma memória. Esse tipo de circuito, apesar de ser de fácil integração apresenta uma área muito grande, quando comparado a circuitos como os sigma-deltas que serão apresentados no capítulo quarto.

3 Conversão Digital-Analógica

Apesar da quantidade de trabalhos publicados sobre conversão analógico-digital ser em torno de dez vezes maior que a quantidade de trabalhos sobre conversão digital-analógica, isso não significa que os DACs sejam menos importantes comercialmente do que os ADCs ou que eles sejam mais simples de serem projetados e construídos. A conversão de dados digitais para dados analógicos equivalentes é uma tarefa fundamental para que os resultados de computações digitais possam ser utilizados e facilmente compreendidos no mundo analógico. [HNA88]

Nesse capítulo serão apresentados circuitos conversores digitais-analógicos. Como o objetivo desse trabalho é implementar circuitos digitais para geração de sinais analógicos, esses serão necessários para realizar a conversão da saída digital em um sinal analógico. Alguns circuitos conversores são na verdade circuitos sinais mistos, ou seja, tem partes digitais e partes analógicas. Outros são circuitos onde as entradas digitais simplesmente controlam chaves nos circuitos analógicos. Os circuitos sinais mistos são os que mais interessam nesse trabalho, pois quanto menor a parte analógica necessária mais facilmente programáveis eles podem ser e mais confiáveis serão os sinais.

A fig. 3.1 mostra o diagrama de blocos de um conversor D/A onde a entrada é uma palavra de 3 bits com sinal e converte para uma voltagem analógica equivalente. As partes básicas desse conversor são: registrador de entrada, fonte de referência estável e o conversor D/A contendo uma chave analógica (ativada pelo bit de sinal), chaves controladas pelos bits de magnitude e uma rede de resistores.

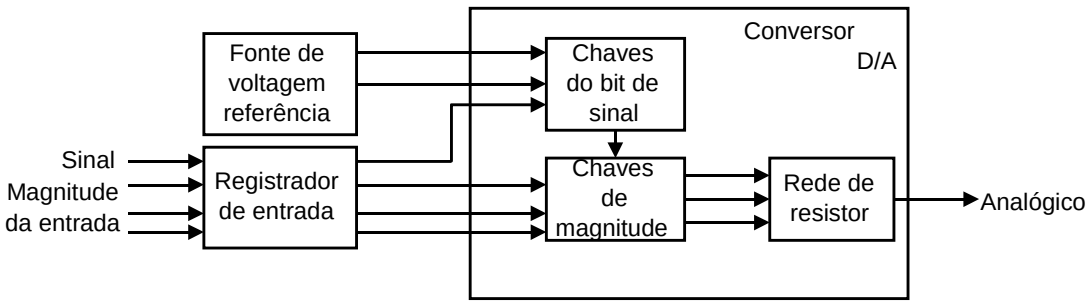


FIGURA 3.1 – Conversor D/A de 3 bits mais sinal [HNA88]

Dois esquemas dessa rede de conversão D/A são comumente usadas: a rede de resistores binarizados e a configuração R-2R. Ambos são mais convenientemente descritos como conversores D/A com voltagem de saída, isto é, onde uma entrada digital resulta num nível de voltagem discreto na saída.

3.1 Tipos básicos de conversores D/A

3.1.1 Rede de Resistores Binarizados

Conversor D/A de resistores binarizados, como o mostrado na fig. 3.2 incluem uma fonte de voltagem referência, um conjunto de chaves, um conjunto de resistores de precisão com binarização e um amplificador operacional. Cada bit da palavra de entrada

controla uma chave. Se o bit for 1 a chave fecha. Quando a chave fecha, a voltagem de referência passa pelo resistor em série com a chave, e uma corrente passa pelo barramento comum. A corrente de todos os ramos da rede de resistores são somadas no amplificador, a voltagem de saída é proporcional à corrente total e, assim, ao valor digital de entrada.

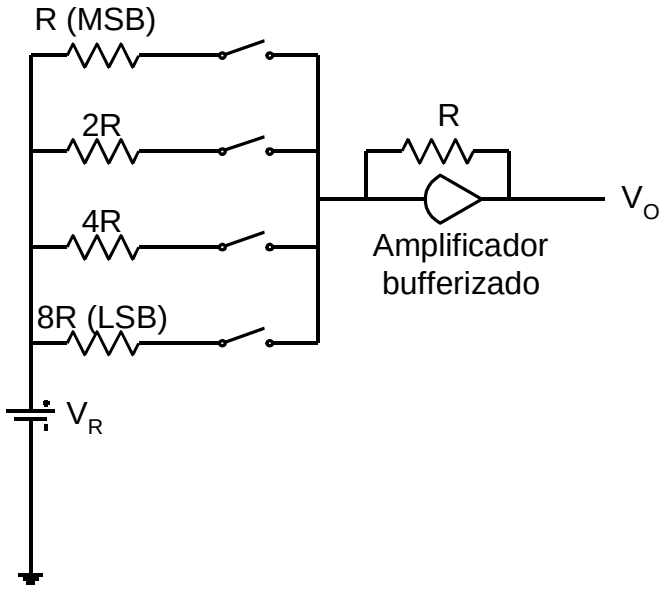


FIGURA 3.2 – Rede de resistores binarizados

3.1.2 Conversores D/A R-2R

Conversores D/A R-2R contém uma fonte de referência, um conjunto de chaves e um conjunto de resistores. Ao invés de um conjunto de resistores binarizados, os conversores D/A R-2R contém dois resistores por bit: um em série com a chave, e o outro, na linha comum, assim a combinação forma uma rede pi em conjunto com os estágios sucessores [HNA88]. As chaves são analógicas sendo que elas devem chavear uma voltagem DC com alto grau de precisão. O ideal seria que a chave tivesse resistência zero quando fechada e infinita quando aberta. A impedância de saída é constante independentemente da posição das chaves, de forma que uma carga fixa não venha a introduzir erros. Um exemplo desse tipo de conversor é mostrado na fig. 3.3.

Essa rede é relativamente simples porque apenas 2 valores de resistores são usados, R e 2R. Para um conversor discreto, os resistores podem ser selecionados de um grande grupo e propriamente escolhidos, porque apenas valores relativos são importantes. Ainda mais, como a combinação R-2R é mais crítica nos resistores dos bits mais significativos, os resistores dos bits menos significativos passam a ser menos importantes e assim sua precisão não é tão crítica e afetam menos a precisão do conversor do que nos bits mais significativos. Algumas vezes o resistor 2R é formado por dois resistores do tipo R em série, permitindo o uso de resistores do mesmo tipo em toda a rede. Isso pode ser importante pois garante um funcionamento adequado em circuitos integrados quando as variações de processo ou de ponto de operação do circuito, como temperatura, desviarem os valores dos resistores, isso ocorrera com todos para a mesma direção.

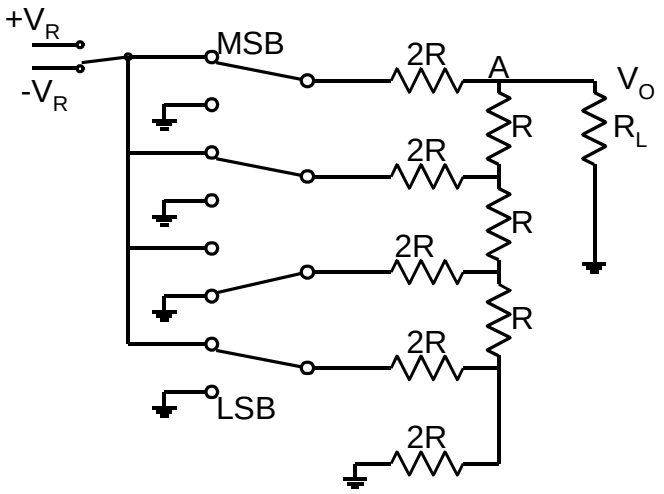


FIGURA 3.3 – Conversor D/A do tipo R-2R

As chaves no conversor D/A R-2R servem para chavear uma voltagem de referência DC com alto grau de precisão. Idealmente isso significa resistência zero quando fechado e resistência infinita quando a chave estiver aberta. A impedância de saída é constante independentemente da posição das chaves, de forma uma carga resistiva fixa não introduza erros de não linearidade. Para demonstrar isso, considere que as chaves estejam na posição da fig. 3.3 e omita a carga resistiva R_L. Os resistores em série com MSB, 2R, são desviados pela resistência equivalente do ponto A na rede de resistores para a terra. A resistência equivalente, determinada pela combinação de elementos em série e em paralelo, começando com LSB é 2R. Assim, a voltagem de saída no ponto A, sem a carga, é $\frac{V_R}{2}$; como a carga na saída é R, a voltagem de saída, com a carga, é $\frac{V_R}{2}\left[\frac{R_L}{R+R_L}\right]$. Similarmente, cada chave contribui com sua própria componente ajustada pelas cargas dos resistores para a voltagem de saída, resultando numa voltagem combinada final que é proporcional a palavra binária de entrada, que controla as chaves.

O conversor da fig. 3.2 é simples e de baixo custo, mas só pode ser usado para converter baixas ou médias resoluções (10 bits ou menos). Conversores práticos baseados na configuração da fig. 3.3 apresentam defeitos de performance que são mais ou menos característicos de todos os circuitos D/A. Além de erros causados por imprecisão dos resistores, e coeficientes de temperatura diferentes, capacitâncias parasitas introduzem atrasos na resposta, especialmente com resistores de alto valor nominal. Já resistores de baixo valor nominal acentuam a queda de voltagem percentual nas chaves, introduzindo mais erros. Além disso, capacitâncias parasitas e, também as chaves (tempo de chaveamento) inserem atrasos na resposta, fazendo com que a velocidade de conversão seja limitada. [HNA88]

3.1.3 Conversor D/A Multiplicativo

Um conversor D/A multiplicativo fornece como resposta uma voltagem analógica proporcional ao produto da entrada digital com a entrada de referência. Para

obter a saída multiplicada corretamente, um comparador é usado para sentir a polaridade da entrada referência e mudar de acordo com o bit sinal, isto é, alto para entradas positivas e baixo para entradas negativas.

Um conversor D/A multiplicativo de baixo custo pode ser construído a partir de um amplificador operacional e uma ponte resistiva, como na fig. 3.4. [JOH73] Normalmente, se as duas entradas do op-amp são ligadas a valores idênticos, a saída resultará em zero se R_1, R_2, R_3 e R_4 tiverem valores iguais. Adicionando-se a ponte resistiva, essa irá atenuar a entrada negativa do op-amp, assim tornando-as diferentes. A saída então será não inversora, e dada pela equação:

$$V_O \cong \left(\frac{R_1}{2} Y_N \right) V_{IN}$$

onde Y_N é o valor na entrada negativa do op-amp e a expressão entre parênteses é o ganho do circuito. [HNA88]

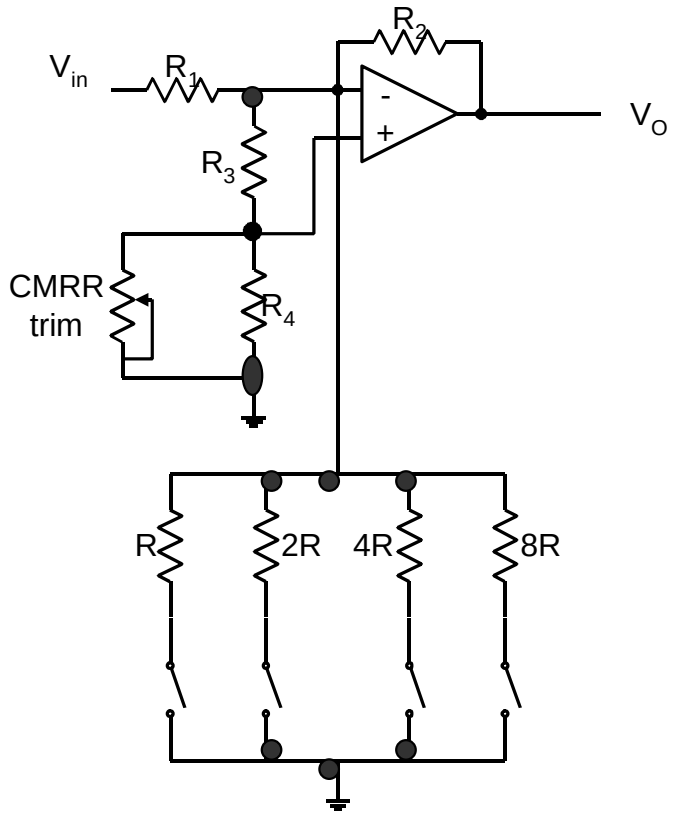


FIGURA 3.4 – Conversor D/A multiplicativo de baixo custo

3.1.4 Registradores em Conversores D/A

Os circuitos conversores D/A de entrada paralela considerados até agora tem como propriedade que a saída analógica continuamente reflete o estado das entradas. Se o circuito conversor for precedido por um registrador, como na fig. 3.5, o circuito irá responder apenas às entradas gravadas no registrador. Essa propriedade é útil em sistemas em que dados são continuamente modificados, mas deseja-se que o D/A converta apenas alguns desses valores e, depois, mantenha constante a saída analógica.

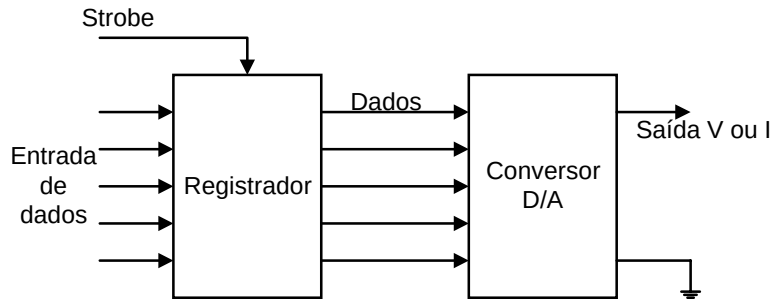


FIGURA 3.5 – Conversor D/A com registrador

O registrador é controlado por um sinal de “strobe”, o que permite sua atualização. A taxa de atualização do registrador é limitado por dois fatores: o tempo de conversão (não é desejável que as entradas digitais mudem antes que o conversor D/A acabe de converter o valor anterior) e o tempo de resposta da lógica digital (não adianta querer gravar um novo valor se a lógica digital não acabou de calcular o valor anterior).

3.2 Gerador de sinais analógicos

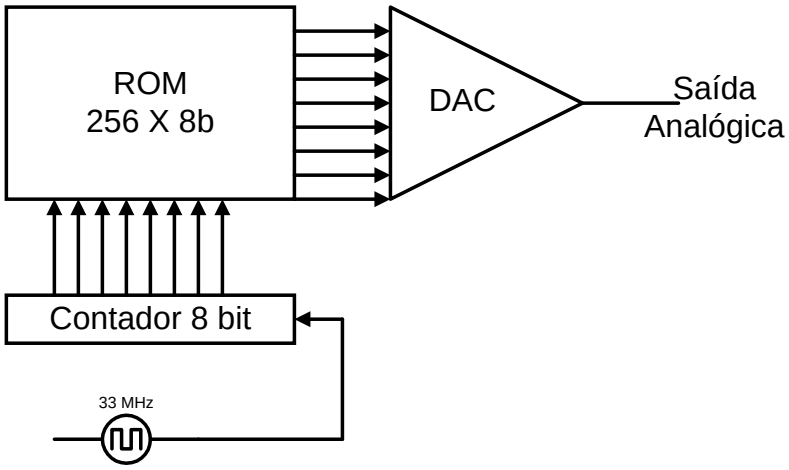


FIGURA 3.6 – Gerador de formas de onda controlado digitalmente

A fig. 3.6 mostra um circuito gerador de sinais usando uma ROM e um conversor D/A. Um contador de 8 bits controlado por um oscilador é o endereçador da ROM (gera os endereços de leitura). Os dados da ROM são convertidos para uma voltagem analógica através de um DAC. Realizando-se uma codificação apropriada da ROM, uma grande variedade de formas de onda podem ser geradas. Múltiplas combinações de ROM e DAC podem ser usadas para gerar diferentes formas simultaneamente, ou múltiplas fases do sinal.

3.3 Fontes de erros na conversão Digital Analógica

Vários fatores determinam a precisão de um conversor D/A, entre eles as propriedades térmicas dos resistores, a estabilidade da fonte de referência e as propriedades do amplificador. Um alto grau de precisão pode ser mantido se os coeficientes térmicos das chaves e dos resistores forem baixos e aproximadamente idênticos, pois assim, mesmo com uma variação na temperatura dos componentes, caso ocorra um desvio no valor convertido será igual em todo o circuito.

O tempo de resposta é dependente da taxa de *slew* e do tempo de estabilização do amplificador para a resolução desejada e na constante de tempo dos resistores com as capacitâncias parasitas associadas. As propriedades do op-amp normalmente são fatores dominantes no tempo de resposta.

Nas figs. 3.7, 3.8 e 3.9 podemos ver um DAC e algumas modificações possíveis para reduzir erros. Nesse caso, o DAC da fig. 3.7 tem uma saída como a mostrada no zoom da fig. 3.10. Agora, se adicionarmos um registrador (fig. 3.8) e um amplificador (fig. 3.9) obteremos uma saída sem os erros mostrados no zoom da fig 3.10.

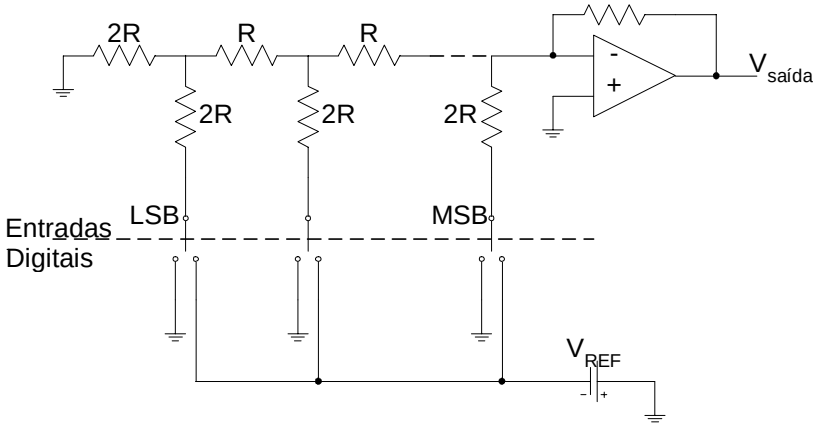


FIGURA 3.7 – DAC básico (com erros)

A função do registrador, que foi adicionado ao DAC na fig. 3.8, é a de reduzir os erros que podem acontecer quando acontece a mudança dos sinais digitais que controlam as chaves. Como normalmente as chaves levam algum tempo para serem movidas e os sinais também não são atualizados muito rápido, principalmente se forem provenientes de uma cadeia de carry por exemplo. Ao colocar o registrador, o tempo total para chaveamento que pode causar erros, passa a ser dependente apenas da velocidade das chaves, e não mais do circuito que gera os sinais digitais.

Conversores D/A com rápidos tendem a gerar os maiores erros. Para essa aplicação, a solução do registrador não é suficiente. Agora, se incorporarmos um janelamento em torno do erro, junto com o registrador, podemos conseguir uma ganho de vários ordens de magnitude no tamanho da falha. A escolha lógica para gerar o janelamento, é um amplificador do tipo *track-and-hold*. Seu design é relativamente simples e elimina a necessidade de outro amplificador de saída no DAC. A fig. 3.9 mostra um diagrama de blocos desse circuito.

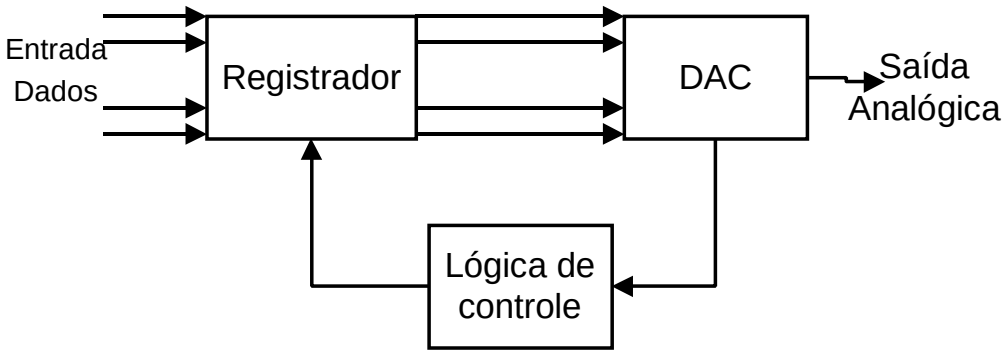


FIGURA 3.8 – DAC com registrador adicionado

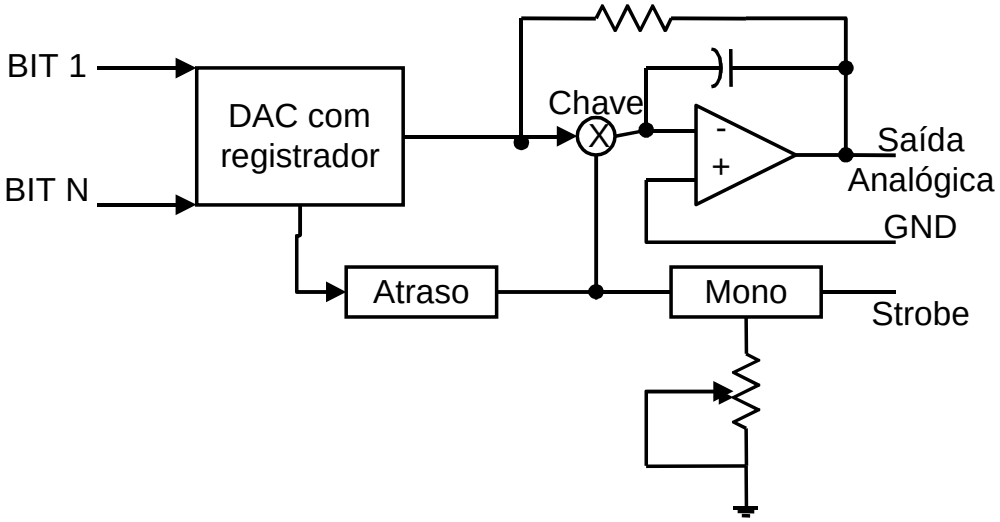


FIGURA 3.9 – DAC com registrador e um amplificador

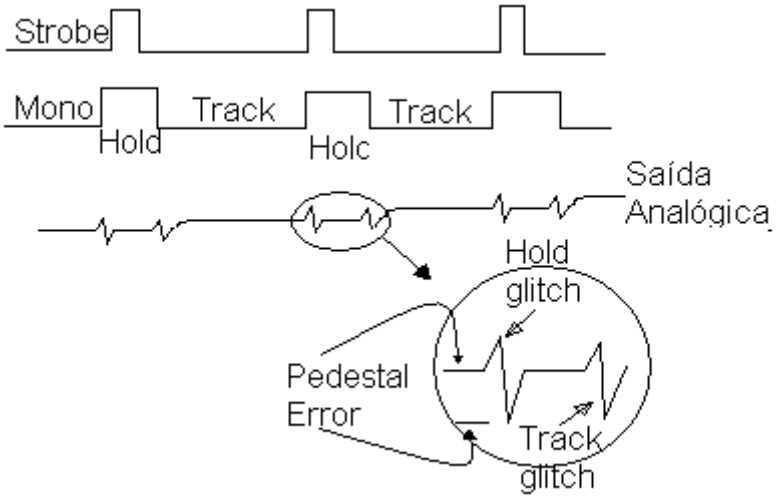


FIGURA 3.10 – Glitches que aparecem na saída do DAC

Esse capítulo apresentou circuitos conversores D/A, necessários para que se possa realizar a geração dos sinais analógicos através de circuitos digitais. Assim como no capítulo anterior os circuitos aqui apresentados são formados por componentes analógicos, algo que se quer evitar, pois estes não são facilmente integráveis e nem ajustáveis sendo difíceis de serem controlados.

No próximo serão apresentados os conversores D/A sigma-delta que, ocupando uma área pequena conseguem realizar a tarefa de conversão sendo quase totalmente digitais.

4 Conversor D/A Sigma-Delta

Nessa parte do trabalho serão discutidos circuitos conversores D/A sigma-delta. Esses circuitos apresentam uma predominância de componentes digitais, sendo que, a única parte analógica necessária para essa classe de circuitos é um filtro passa-baixa. Assim sendo, esses circuitos não só são de fácil integração como podem ser testados através de técnicas digitais, tendo assim algumas das características desejáveis.

A motivação para introdução do sistema sigma-delta é facilmente compreendida se analisarmos as condições necessárias para um DAC com precisão de 16 bits usando como referência uma voltagem de 3V. Esse DAC exige tal precisão que seria necessário procedimentos de calibragem e *trimming* muito caros. Usando-se técnicas de sobre amostragem, pode-se superar os problemas de precisão analógica, substituindo essa pela complexidade e velocidade digital para que o circuito não seja sensível às imperfeições analógicas.

O diagrama básico de um sistema de DAC com sobre amostragem é mostrado na fig. 4.1. A entrada X do conversor é um sinal digital multibit de tamanho b1 numa taxa fn. Normalmente, fn é um pouco acima da taxa Nyquist do sinal. Esse sinal é processado pelo filtro de interpolação (IF), que altera a taxa para Rfn (onde R é a taxa de sobre amostragem). A palavra sobre amostrada tem tamanho b2 a qual é ligeiramente menor ou igual a b1. O sinal então passa por um ciclo de formatação de ruído (NL), também chamado de estado modulador, o qual diminui drasticamente o tamanho da palavra, tipicamente um bit. Depois esse bit é convertido pelo DAC.

A saída analógica do DAC contém uma réplica linear da entrada digital X junto com uma grande quantidade de ruído devido a erros de quantização introduzidos no NL. Como esse ruído se encontra quase todo fora da banda do sinal, ele pode ser quase totalmente eliminado pelo filtro passa-baixa (LPF) que segue o DAC. O desempenho do conversor sigma-delta é determinado pelo ciclo de formatação do ruído.

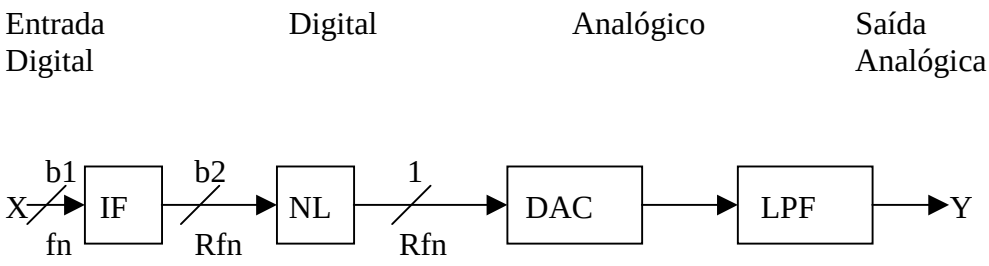


FIGURA 4.1 – Diagrama de blocos de um DAC sigma-delta

4.1 Arquiteturas para o ciclo de formatação do ruído

O propósito desse estágio é, tendo como entrado um sinal quantizado truncá-lo para um tamanho de palavra bem pequeno sem introduzir muito ruído.

4.1.1 Laço Sigma-Delta

A forma geral de um laço sigma-delta é mostrado na fig. 4.2. Na sua forma mais simples, o filtro passa-baixa pode ser simplesmente um acumulador, resultando num ciclo de primeira ordem, como na fig. 4.3. O truncador, nada mais é do que um circuito que deixa passar apenas o bit mais significativo da palavra, descartando todos os outros bits. O circuito da fig. 4.3 é o mesmo circuito da fig. 4.2, simplesmente substituindo o filtro passa-baixa por um acumulador realimentado. Nessa figura podemos ver que o valor da entrada é somado com o bit mais significativo anterior, ou saída anterior, o resultado é somado com o antigo valor de saída, antes do truncamento. Esse valor é então armazenado no registrador para futura soma, e o valor armazenado será também usado pelo truncador para gerar a nova saída do circuito.

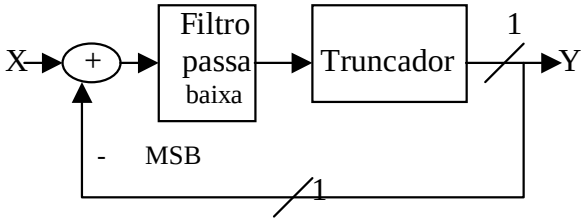


FIGURA 4.2 – Diagrama de blocos para um laço sigma-delta

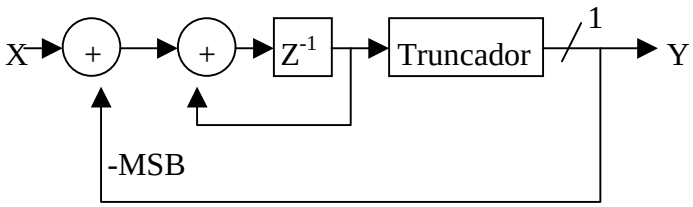


FIGURA 4.3 – Laço sigma-delta de primeira ordem

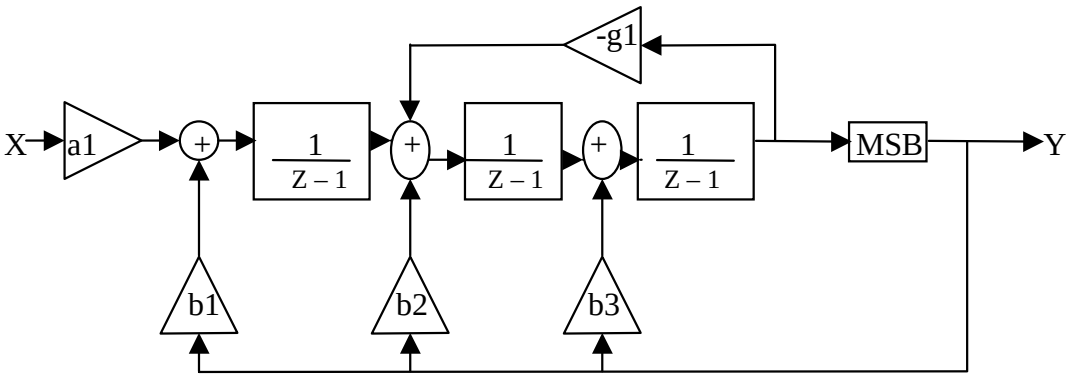


FIGURA 4.4 – Laço sigma-delta de terceira ordem usando apenas realimentação [NOR96a]

A fig. 4.4 mostra um sistema sigma-delta com ordem superior (terceira ordem) contendo uma cascata de um acumulador e um oscilador digital de segunda ordem, com a saída MSB sendo realimentada para cada estágio. Esse circuito é dito de terceira ordem por apresentar três registradores dividindo o circuito. O primeiro registrador,

junto com o somador formam um acumulador. Os outros dois registradores com os somadores e a realimentação do terceiro registrador para o somador, logo antes do segundo registrador formam um oscilador digital de segunda ordem. Cada um dos somadores recebe também, como uma das parcelas de soma o valor da saída do circuito multiplicado por um fator (b1, b2 e b3). Essa realimentação, multiplicada por um fator, serve para reduzir a taxa de erros na conversão D/A.

Um modulador similar, no qual a saída de cada estágio é propagada para a entrada do quantizador enquanto o MSB é realimentado apenas para o nodo inicial do laço, pode ser visto na fig. 4.5. O circuito mostrado é de quinta ordem. O resultado de cada estágio é somado e então passa pelo quantizador, ou truncador, que separa o bit mais significativo dos outros, tendo como saída esse bit, o qual também é realimentado para ser somado com uma entrada futura do circuito.

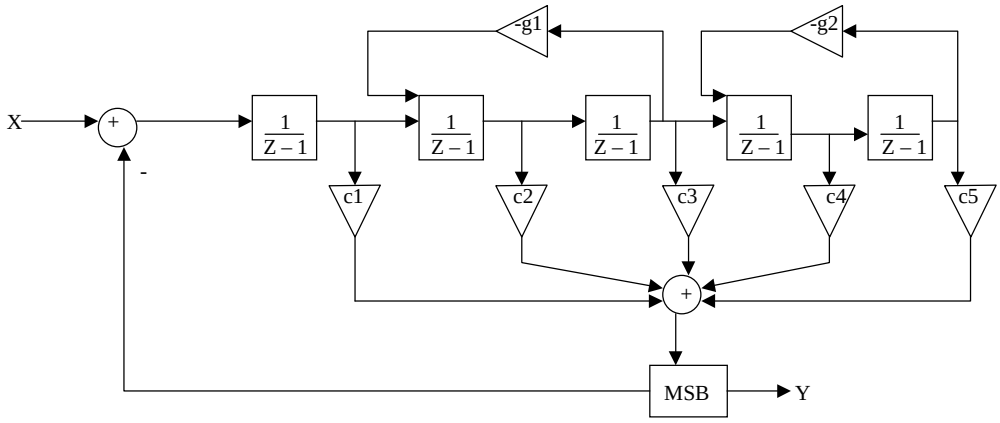


FIGURA 4.5 – Laço sigma-delta de quinta ordem com realimentação e propagação

Nessas estruturas pode-se economizar área de hardware se os fatores constantes a1, b1, b2, ..., gn, etc. forem escolhidos como somas de algumas potências de dois. Tal quantização dos coeficientes pode introduzir uma mudança de nível ou um pico num sinal que normalmente seria plano, mas essa distorção de amplitude pode ser compensada nas características do filtro. Outro lugar onde se pode economizar hardware é reduzindo o número de bits que precisam ser propagados no laço à medida em que o sinal se propaga da esquerda para a direita.

4.1.2 Estrutura de realimentação do erro

Uma estrutura alternativa útil para um modulador sigma-delta é ilustrado na fig. 4.6. Nessa estrutura, ao invés de realimentar com a saída MSB, usa-se a negação do erro de truncamento composto pelos bits neglicenciados LSB realimentados através de um filtro H(z). Uma análise simples mostra que, no domínio Z, a saída do laço é dada por:

$$Y(z) = X(z) + [1 - H1(z)]E(z)$$

onde Y, X e E são as transformadas Z da saída y, da entrada x e do erro de truncamento e do laço, respectivamente. Essa estrutura de realimentação do erro normalmente leva a uma realização mais simples (substituindo o filtro digital por um acumulador) do que a mostrada na fig. 4.2. e assim mais comumente utilizada para laços digitais.

No caso mais simples, pode-se fazer H1(z) = z^{-1} como função de transferência do filtro de realimentação. Assim, a função de transferencia do sistema se torna

$$H(z) = 1 - H1(z) = 1 - z^{-1}$$

obtendo-se uma formatação de ruído de primeira ordem. Para obter um de segunda ordem, podemos selecionar a função de transferencia do filtro como sendo

$$H1(z) = 1 - (1 - z^{-1})^2 = z^{-1}(2 - z^{-1}) .$$

Nenhuma dessas estruturas requer multiplicação de palavras multibit, apenas atrasos, adições e deslocamentos. Para evitar saturação do somador, um limitador pode ser adicionado ao circuito. Alternativamente, o circuito pode ser operado com uma saída de dois bits ao invés de apenas um, e um segundo laço de primeira ordem pode ser adicionado para gerar a saída final de um bit, como mostrado na fig. 4.7. A velocidade do sistema é um pouco menor do que se a entrada "x" fosse diretamente para a entrada do segundo laço, mas o custo da complexidade do hardware digital é grandemente reduzido através do uso de componentes mais lentos. O estágio mais rápido, ou seja, o segundo laço também pode ser realizado por uma memória pequena e um registrador de deslocamento.

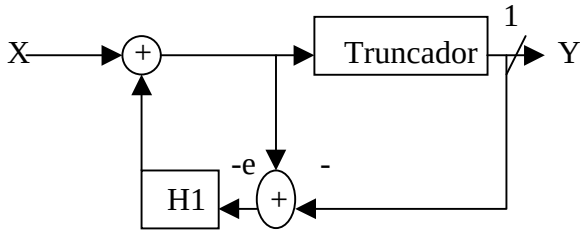


FIGURA 4.6 – Esquema de realimentação do erro

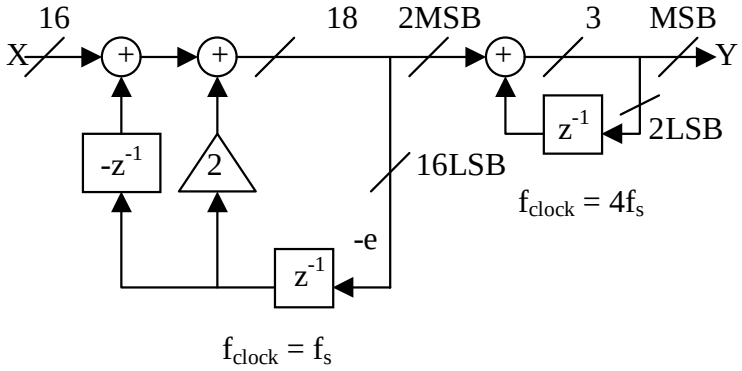


FIGURA 4.7 – Estrutura sigma-delta de primeira ordem onde o estágio rápido tem entrada com 2 bits

4.1.3 Estrutura em cascata

A realização em cascata (MASH) dos laços digitais é uma outra opção. A fig. 4.8 mostra um modulador de segunda ordem construído como uma cascata de dois moduladores de primeira ordem. Infelizmente, a saída do circuito é então a soma de duas palavras de um bit, assim é necessário um DAC de 2 bits com os mesmos requerimentos de linearidade do sistema. Para evitar esse problema, é possível realizar duas conversões individuais usando dois DACs de um bit e combinar seus resultados num estado somador. Alternativamente, um outro estágio sigma-delta simples com uma

freqüência de amostragem maior pode ser adicionada para reduzir o comprimento da palavra para um bit, como na fig. 4.7.

Uma estrutura na qual tanto o erro como a saída são realimentados, combinando assim a realimentação de um bit com a multibit, é mostrada na fig. 4.9. Uma analise dessa estrutura, dá ao sinal de saída no domínio Z a forma

$$Y(z) = z^{-2} X(z) + (1 - z^{-1})^2 E(z)$$

assim, o circuito funciona como um laço truncador com formatação de ruído de segunda ordem. O primeiro somador mostrado na figura 4.9 pode ser substituído por um somador de 18 bits, onde o sinal X é estendido e o sinal da realimentação de um bit é o sinal de carry.

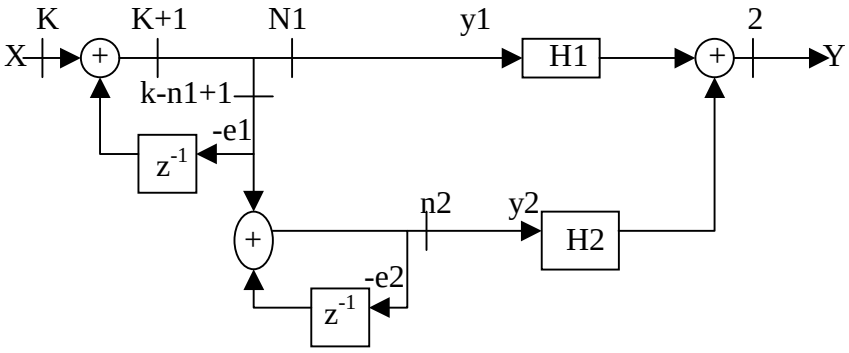


FIGURA 4.8 – Estrutura em cascata para modulador sigma-delta de segunda ordem [NOR96a]

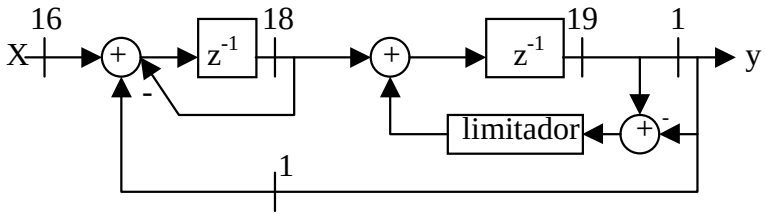


FIGURA 4.9–Laço de segunda ordem com realimentação multibit e bit único[NOR96a]

4.1.4 Laço quantizador multibit

O uso de quantizador multibit traz diversas vantagens, como maior estabilidade, menor requerimento de velocidade, entre outros. Além disso, é mais fácil evitar a geração de tons quando se use laço multibit. Infelizmente, para atingir esses objetivos, algum hardware extra é necessário para cancelar os efeitos de não linearidade inerentes. Existem diversas técnicas que podem ser usadas, por exemplo, randomização, *barrel-shifting*, ou nível médio individual.

Outra técnica que pode ser usada é a correção digital. A fig. 4.10 mostra o diagrama de blocos de um DAC corrigido digitalmente. Nesse sistema, a RAM contém palavras multibit representando com bastante precisão os níveis reais das saídas do DAC de N bits. Como o DAC e a RAM tem as mesmas entradas, suas saídas (analógica e digital, respectivamente) também são valores correspondentes. Além disso, como os

componentes de baixa freqüência dos dados de saída da RAM seguem a entrada X com uma precisão muito grande devido ao laço de realimentação, a mesma conclusão serve para a saída do DAC. Ou seja, a saída y do truncador é distorcida pela inclusão da RAM no laço de realimentação de tal maneira que o espectro de baixa freqüência da saída do DAC é uma réplica analógica precisa da entrada digital X.

Os dados armazenados na RAM da fig. 4.10 podem ser obtidos utilizando-se o processo de calibragem. Para economizar algum hardware digital, pode-se também armazenar os equivalentes digitais dos erros nos níveis do DAC, ao invés do próprio nível, e combinar a entrada e saída da RAM através de um laço adicional.

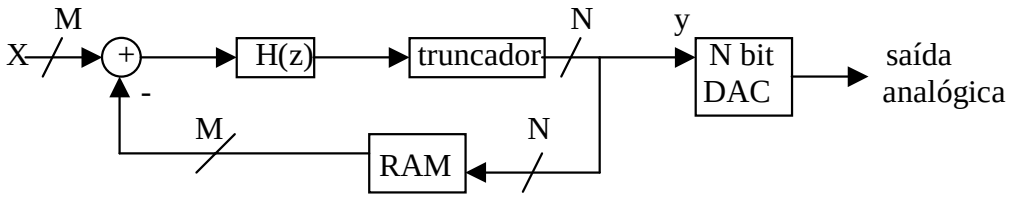


FIGURA 4.20 – Um laço de N bits corrigido digitalmente

Outra técnica muito utilizada é a da quantização dupla. A fig. 4.11 mostra uma estrutura de DAC com quantização dupla, muito parecida com a arquitetura Leslie-Singh [LES90], a qual não é alvo desse trabalho. Nesse sistema, o negativo do grande erro de quantização e1 devido ao laço de um bit é alimentado num caminho de correção, onde ele é truncado para M bits (M>1). O erro truncado é então convertido para a forma analógica, filtrado pelo filtro analógico passa-alta H2(z) e somado à saída analógica do DAC simples para reduzir o ruído e1 do sinal de saída y. Ele é substituído em y pelo erro de truncamento do truncador de M bits e a não linearidade do DAC de M bits filtrado pelo filtro H2(z).

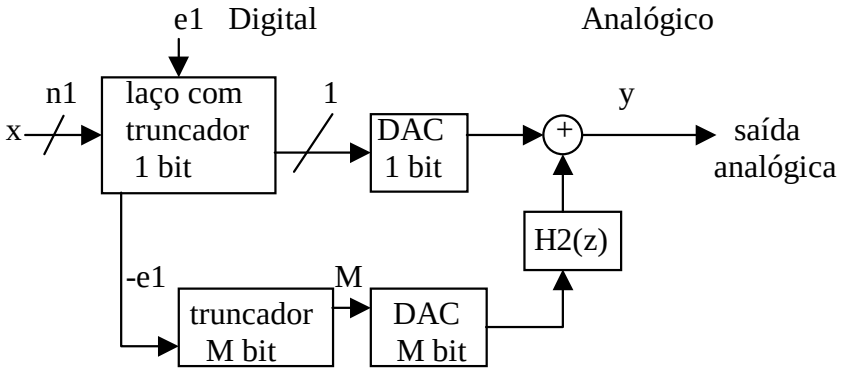


FIGURA 4.11 – Laço sigma-delta com truncamento duplo [LES90]

Sob condições ideais, a saída corrigida y do laço não contém mais o erro e1, mas sim o erro de truncamento e_M formatado da mesma maneira que e1 também era. Esse resultado é uma melhora de aproximadamente M bits na SNR. É normal que ocorra algum vazamento de e1 para o sinal y, mas devido aos zeros em dc da função de transferência do filtro, esse efeito é não significativo.

Outra opção, quando se usa truncamento duplo, é, ao invés de realizar simplesmente um segundo truncamento, usar um segundo laço de formatação de ruído no caminho de correção de erro. Como resultado, o erro devido a e_M fica mais reduzido.

Outra alternativa para laço com truncamento duplo é mostrado na fig. 4.12. Em ambos estágios são usados dois acumuladores, para aumentar a estabilidade foi usado um fator de escala 0,25. O truncador Q1 no primeiro laço tem três níveis possíveis de saída: -1, 0 e 1. Enquanto isso, Q2 tem dois níveis, -1 e 1. Como resultado, os caminhos de saída Ds1 e Ds2 tem, cada um, três possíveis níveis: $Ds1 = 0,\pm 1$ e $Ds2 = 0,\pm 2$. A saída geral Ds1+Ds2 pode ser 0, ± 1, ± 2, ou ± 3.

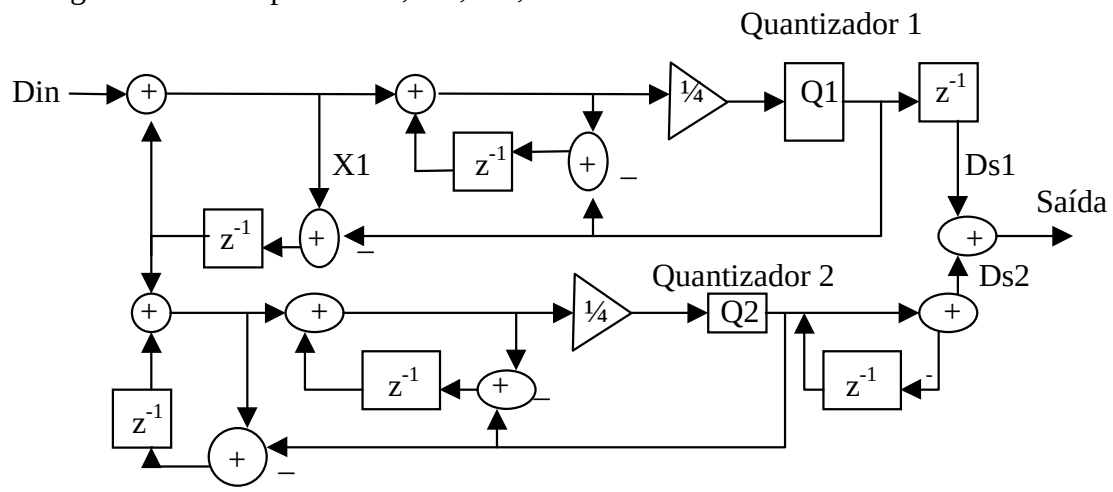


FIGURA 4.13 – MASH multibit de dois estágios [NOR96a]

A entrada do segundo estágio é $z^{-1}X1$ - Ds1, onde X1 é a saída do primeiro somador do primeiro estágio. Uma análise mais detalhada [UCH88] indica que o sistema quase completamente cancela o erro de truncamento de Q1 e fornece uma formação de terceira ordem para o ruído de truncamento gerado por Q2. Para evitar distorção harmônica, é preferível usar dois DACs separados para converter Ds1 e Ds2.

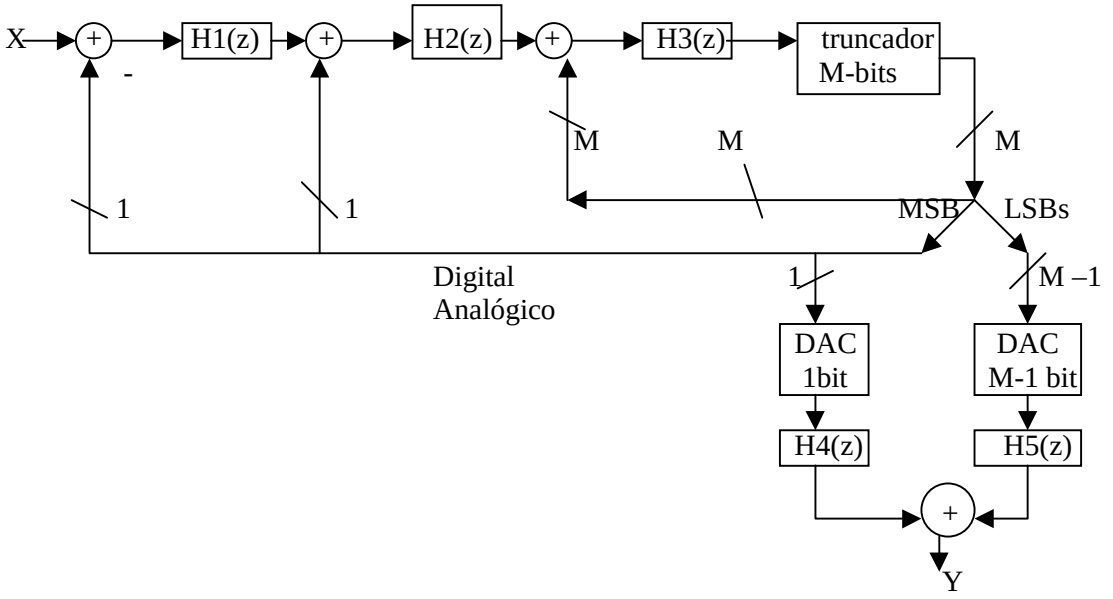


FIGURA 4.14 – Laço único com duplo truncamento [UCH88]

Uma arquitetura de truncamento duplo diferente, onde tanto a saída de um bit quanto a multibit são realimentadas dentro de laço único é mostrada na fig. 4.13. Aqui, o bloco H5(z) é um filtro passa-alta analógico, então o erro de não linearidade do DAC multibit é suprimido na banda base [UCH88].

Foi mostrado recentemente [NOR96] que se um DAC interno multibit linear pode ser realizado, então pode-se particionar as palavras de entrada que contêm N bits em M bits MSBs e (N-M) bits LSBs, usar um laço digital para truncar o LSB apenas, e recombinar a MSB com os LSB truncados. Os dados resultantes conterão pouco ruído adicional além do que é obtido diretamente de laço de N bits, e o hardware necessário é muito menos elaborado. [NOR96a]

Um fato comum a todos DACs sigma-delta é a geração de diferentes tons. Esse tendência é particularmente mais forte em DACs onde os sinais do modulador são totalmente digitais e a aritmética é puramente racional, sem nenhum tratamento em caso de estouro da capacidade dos somadores. Até pouco tempo atrás acreditava-se que sistemas com apenas um caminho de dados não seriam capazes de gerar múltiplos tons, mas recentemente foi demonstrado tanto através de simulações como na prática a geração desses múltiplos tons. Circuitos com vários caminhos de dados, como alguns mostrados aqui, tendem a reduzir esses múltiplos tons de acordo com a quantidade de níveis do DAC de M bits, ou seja, se esse tem 3 níveis, serão gerados três tons a menos.

Esse capítulo apresentou toda uma classe de conversores D/A que, atingem os requisitos do sistema ser implementado. Além de serem pequenos, ainda por cima são quase totalmente digitais, facilitando a integração em silício. Para esses circuitos, o único elemento analógico necessário é um filtro passa-baixa, o qual não precisa ser integrado, o que elimina a necessidade de incluir componentes analógicos no circuito impresso. Esses circuitos por serem seriais não apresentam uma grande velocidade impondo assim algumas restrições de desempenho ao sistema quanto um todo. No caso do sistema proposto isso não é um problema pois não foi colocado nenhum requisito de desempenho para este.

5 Circuitos Digitais Geradores de Sinais Analógicos

Nesse capítulo, serão discutidos alguns circuitos digitais que são capazes de gerar sinais, assim como os que foram discutidos no capítulo dois, só que em formato digital. Ou seja, circuitos que geram formas de ondas digitais que podem ser convertidos, através de circuitos conversores digitais-analógico obtendo-se assim o sinal analógico na forma desejada. Esses circuitos como são digitais atendem as características pedidas de serem programáveis e testáveis através de métodos digitais.

Existe um procedimento completo para síntese de filtros digitais baseados em protótipos de redes LC. A estrutura do filtro digital resultante é normalmente formado por um conjunto de osciladores, os quais são formados pelo cascateamento de dois integradores discretos da forma $z^{-1}/(1-z^{-1})$ e $1/(1-z^{-1})$, em um laço com o sinal de um integrador sendo positivo e o outro negativo. Essa forma é mostrada na fig. 5.1.

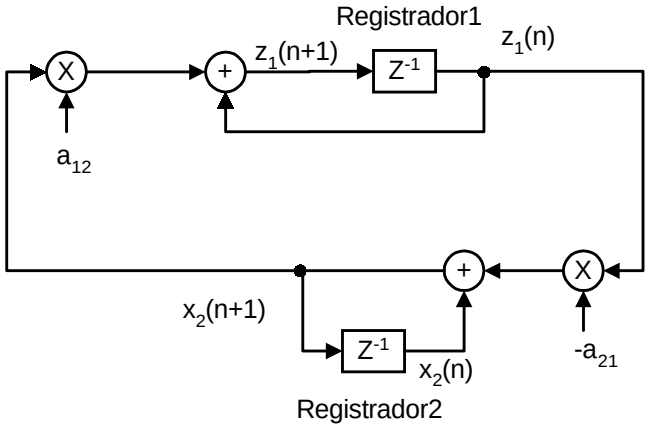


FIGURA 5.1 - Um circuito digital oscilador de segunda ordem consistindo de dois integradores cascateados em um laço [ROB95]

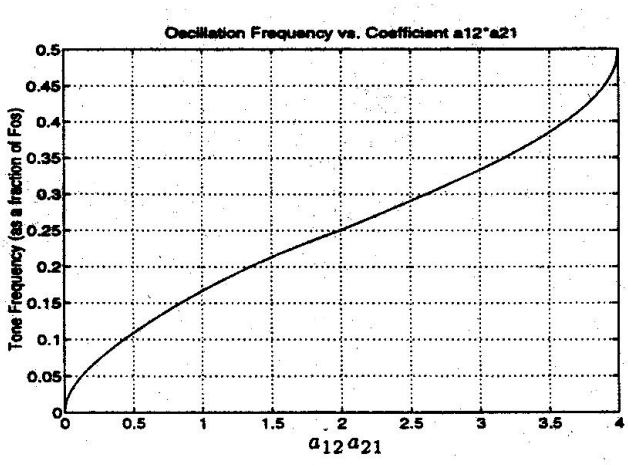


FIGURA 5.2 - Ilustrando a relação funcional entre a freqüência de oscilação e o produto $a_{12}a_{21}$ [ROB95]

Por esse circuito pode ser visto através da observação das raízes do sistema que variações nos coeficientes a_{21} e a_{12} podem causar deslocamentos na freqüência de oscilação. Ainda, a amplitude do tom oscilatório será uma função das condições iniciais

dos registradores 1 e 2. Para que a oscilação seja garantida, o produto $a_{12}a_{21}$ deve ficar entre 0 e 4, mantendo os sinais dentro do círculo unitário. [ROB95]

A fig. 5.2 ilustra a relação entre a freqüência de oscilação e o produto $a_{12}a_{21}$. Para valores de $a_{12}a_{21}$ entre 0 e 4, a freqüência de oscilação varia continuamente entre 0 e $f_{os}/2$. Se no entanto, os coeficientes são restritos a valores discretos as freqüências de oscilação selecionáveis também serão limitados a valores discretos, mas, mesmo com essa restrição, o circuito garantidamente oscilará.

As freqüências que podem ser gerados pelo circuito da fig. 5.1 e plotados na fig. 5.2 como uma função da freqüência de oscilação seguem o sistema de equações

$$\omega_0 = \begin{cases} f_{os} \cos^{-1}\left(1 - \frac{a_{21}a_{12}}{2}\right) & para\ 0 < a_{21}a_{12} \le 2 \\ f_{os}\pi - f_{os} \cos^{-1}\left(1 - \frac{a_{21}a_{12}}{2}\right) & para\ 2 < a_{21}a_{12} < 4 \end{cases}$$

Enquanto que as amplitudes seguem a formula

$$A = \frac{a_{12}x_2(0)}{sin(\omega_0 T)}$$

Nessa fórmula foi considerado que o registrador x_1 é iniciado como zero, dessa forma o desvio de fase também é zero e a fórmula da amplitude fica simplificada como mostrado.

A geração de um tom analógico pode ser facilmente alcançada passando-se a saída do oscilador digital através de um conversor D/A . Um conversor D/A por amostragem de um bit consistindo de um filtro de interpolação, um modulador sigma-delta, e um filtro analógico é ilustrado na fig. 5.3. Como mostrado, o filtro de interpolação recebe uma palavra de N bits na taxa f_n e a repassa numa taxa f_{os}. Esse sinal é então entregue a um modulador sigma delta onde ele é convertido a uma seqüência de bits na mesma taxa f_{os}, contendo a informação do sinal original mais um ruído. Como esse ruído reside em altas freqüências, este pode ser facilmente eliminado através de uma filtragem, obtendo-se assim um bom SNR.

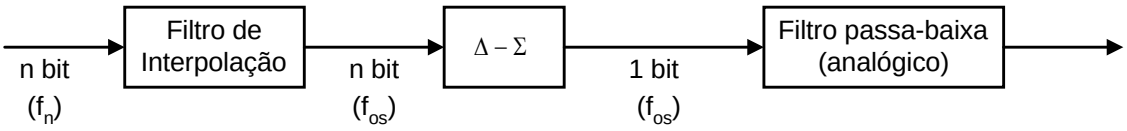


FIGURA 5.3 - Diagrama de bloco de um conversor D/A típico

Enquanto que cascatear um oscilador digital com um conversor D/A é um método simples e viável de geração de sinais analógicos, a grande quantidade de área necessária para o filtro de interpolação pode ser inaceitável em várias aplicações. O projeto alternativo descrito abaixo, ameniza esse problema operando o oscilador digital na freqüência f_{os}, eliminando a necessidade do filtro de interpolação. Além disso, se movermos o modulador sigma-delta para dentro do laço do oscilador, as duas multiplicações multibits podem ser simplificadas drasticamente numa implementação muito eficiente.

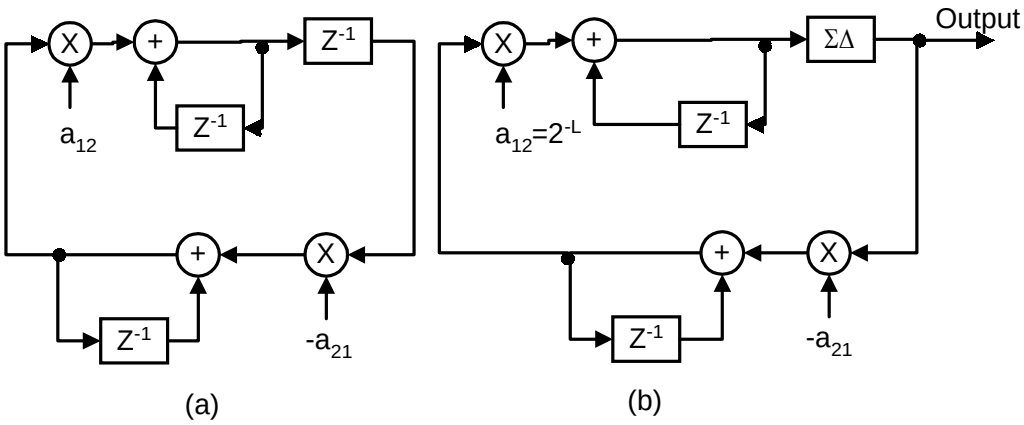


FIGURA 5.4 - Circuito oscilador com o atraso modificado (a), circuito (b) com o atraso substituído por um modulador sigma delta

O oscilador digital da fig. 5.1 pode ser re-arranjada usando uma manipulação do fluxo de dados. Referindo-se a oscilador da fig. 5.1, o atraso no caminho de propagação pode ser movido para dentro do caminho de realimentação se um outro atraso for inserido em série com a saída. A configuração resultante é mostrado na fig. 5.4(a). Nesse ponto, o atraso em série com o multiplicador n por n pode ser substituído por um modulador sigma-delta. O circuito resultante é mostrado na fig. 5.4(b) onde o modulador sigma-delta é assumido a ser um modulador de segunda ordem do tipo mostrado na fig. 5.5. Note que se a saída do oscilador vier diretamente da saída do modulador sigma-delta, a conversão D/A pode ser realizada através de uma filtragem passa-baixo da seqüência de bits. A complexidade do filtro analógico dependerá grandemente da taxa de amostragem e assim, será reduzido aumentando-se a freqüência de amostragem f_{os}.

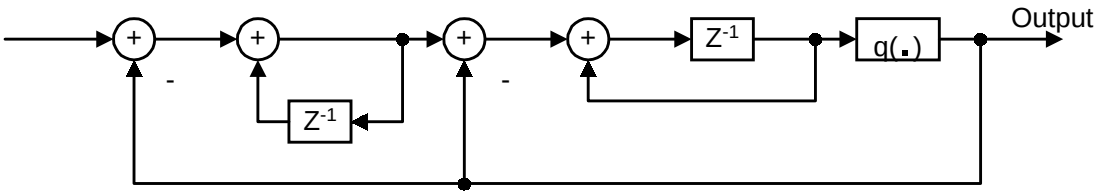


FIGURA 5.5 - Modulador sigma-delta de segunda ordem

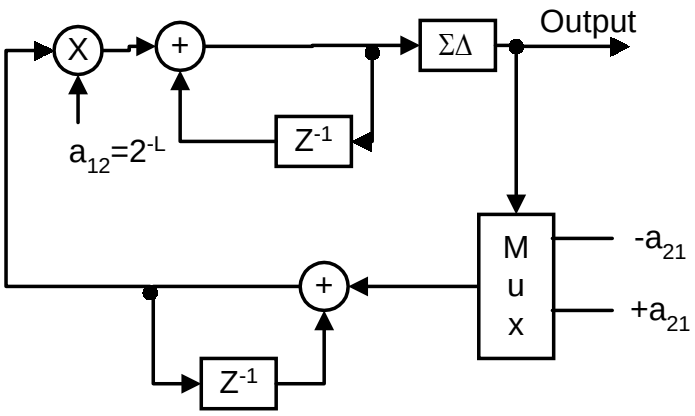


FIGURA 5.6 - Substituindo o multiplicador 1 por n do circuito na fig. 2.4(b) por um multiplexador 2 por 1.

Na fig. 5.6 o valor de a_{12} foi ajustado para um valor que é uma potência negativa de dois (2^{-L}), e assim possibilitando-se que a multiplicação seja implementada usando um deslocamento fixo de L para a direita, ou seja, realizando-se uma divisão por 2^L. Como a saída do sigma delta pode assumir apenas valores 1 ou -1, a multiplicação por a_{21} pode ser implementado com um multiplexador 2 por 1. Resultado é um circuito muito eficiente requerendo apenas 6 somadores, 4 registradores e um multiplexador 2 por 1.

5.1 Geração de sinais analógicos multi-tons

Um excelente exemplo de uma aplicação multi-tom pode ser encontrada no teste de circuitos integrados. A freqüência de resposta de um dispositivo pode ser alcançado eficientemente e de forma acurada aplicando-se simultaneamente vários sinais de teste. Esse método é muitas vezes mais rápido do que percorrer toda a faixa de freqüência de interesse com um único sinal. Essa geração de sinais multi-tons pode ser obtida através da soma de vários sinais.

A soma de uma seqüência de bits, costuma resultar em dois bits. Para evitar isso, e manter o sinal gerado como uma seqüência de bits pode-se usar o circuito mostrado na fig. 5.7, onde o somador mostrado é um somador completo convencional com s, c_{out}, e c_{in} respectivamente a soma, o carry-out e o carry-in. Nesse circuito, a saída vem do carry-out do somador. Intuição sugere que a soma real irá diferir da saída pela valor de s do somador. Esse erro é corrigido através da realimentação de s ao circuito através do carry-in, passando por um registrador.

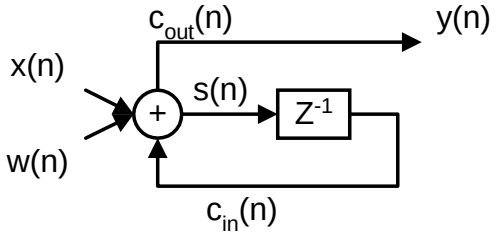


FIGURA 5.7 - Somador de um bit com realimentação

Existem dois métodos de se realizar a geração de sinais multi-tons. O primeiro método emprega múltiplos osciladores para gerar tons arbitrários que são subseqüentemente somados para obter um sinal multi-tom. Enquanto conceitualmente simples, esse método exige uma grande área de implementação, particularmente guando o número de tons aumenta. Dependendo da aplicação, isso pode ser inaceitável. Outro método é por multiplexação no tempo para reduzir o hardware necessário. Essa redução é conseguida através da redução da velocidade do circuito.

Como ambos os métodos tem vantagens e limitações, é difícil dizer que algum deles é superior para todas as aplicações. A escolha de qual arquitetura será usada dependerá de vários parâmetros, incluindo o número de tons, o SNR necessário e a área disponível no silício. Em alguns casos, uma combinação de ambos pode ser necessário.

5.1.1 Geração paralela e soma

O esquema de geração paralela e soma é ilustrado na fig. 5.8. Aqui, vários osciladores digitais do tipo discutido anteriormente são utilizados em paralelo para gerar N tons individuais. Essas seqüências de bits moduladas pelo sigma-delta são então somadas usando-se o somador mostrado na fig. 5.7, ou vários circuitos como o da fig.

5.7 para realizar as somas necessárias, e convertido para o analógico através do uso de um filtro passa-baixo. Como dito anteriormente, essa arquitetura pode ocupar uma grande área de silício porque cada tom adicional necessita da presença de um outro oscilador digital. Isso pode ser muito dispendioso em área para muitas aplicações, incluindo MADBIST. A alternativa desse método mostrado a seguir usa multiplexação no tempo numa troca entre área e velocidade.

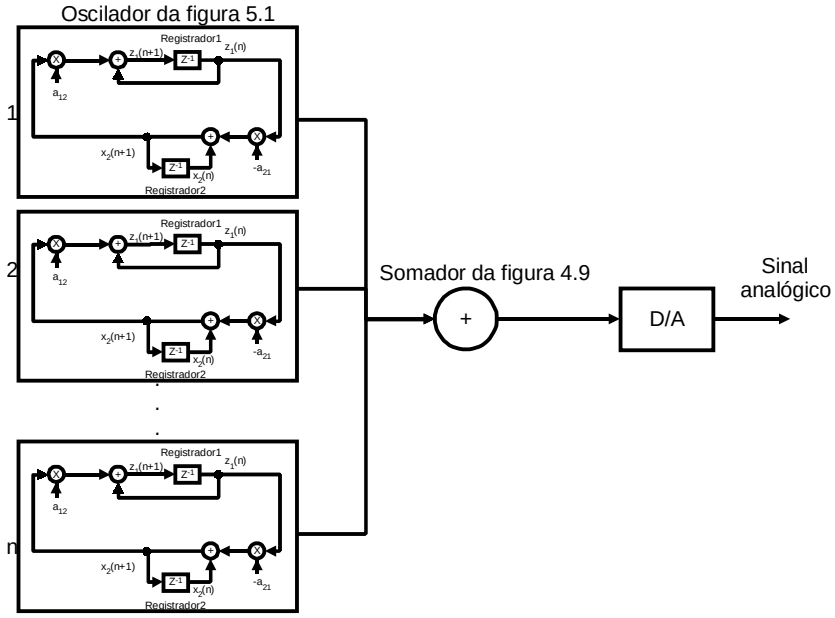


FIGURA 5.8 - Tons individuais são gerados em paralelo e somados. O resultado é então convertido para o analógico.

5.1.2 Multiplexação no tempo

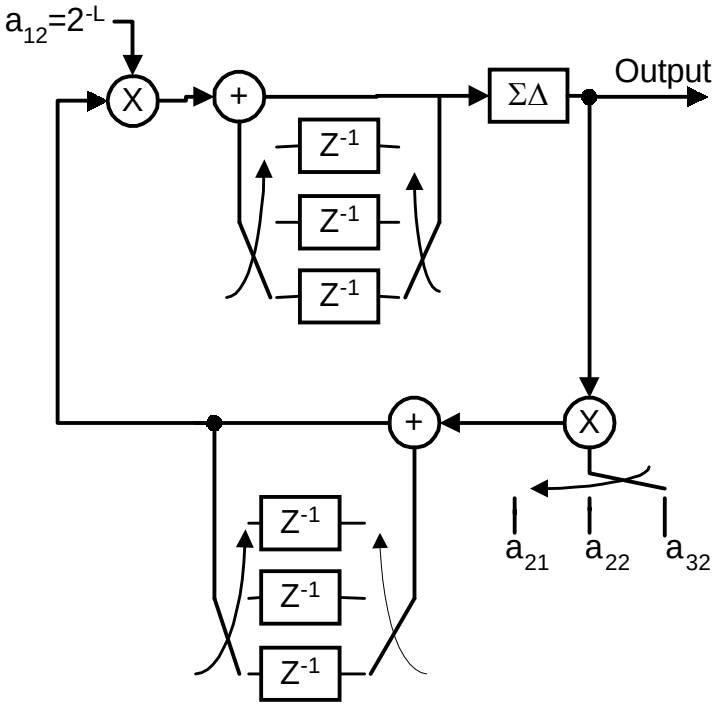


FIGURA 5.9 - Implementação multiplexada no tempo do oscilador digital

A multiplexação no tempo é uma técnica muito útil, que tem várias aplicações na área de telecomunicações. A velocidade desse circuito é N vezes menor do que a velocidade do circuito mostrado na fig. 5.6, onde N é a quantidade de tons a serem gerados.

A fig. 5.9 mostra uma versão multiplexada no tempo do circuito oscilador mostrado na fig. 5.6. Deve ser dito que o modulador sigma delta do diagrama foi modificado para processar sinais multiplexados no tempo. Essa modificação é essencialmente a adição de elementos de atraso extra para cada integrador dentro do modulador sigma delta. O número de atrasos é igual ao número de tons adicionais. Isso foi mostrado na fig. 5.10 para um exemplo de dois tons.

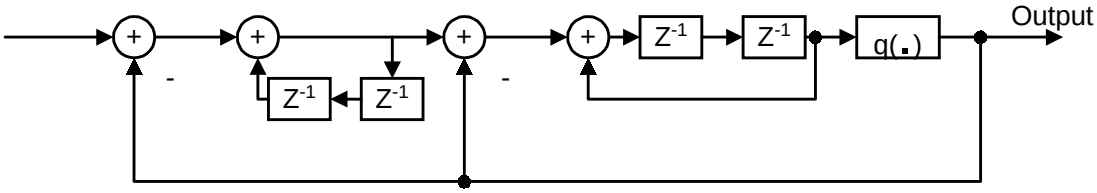


FIGURA 5.10 - Um modulador sigma delta modificado para aceitar uma multiplexação no tempo de dois tons

Como antes, a saída pode ser tomada diretamente da saída do modulador sigma delta. Nesse caso no entanto, a saída conterá N vetores multiplexados. Assim, se o método de soma for aplicado, a seqüência de bits pode ser convertida diretamente para o analógico sem a necessidade de demultiplexação e consequentemente sofrer uma filtragem passa-baixo para obter um sinal multi-tom. Se, no entanto, outro método de soma for preferido, os N vetores devem ser demultiplexados e somados usando um método alternativo antes da conversão D/A.

5.2 Gerador de funções baseado no gerador de senóides

Os circuitos mostrados anteriormente serão agora modificados para gerar ondas quadradas, triangulares ou dente de serra analógicos.

5.2.1 Onda quadrada

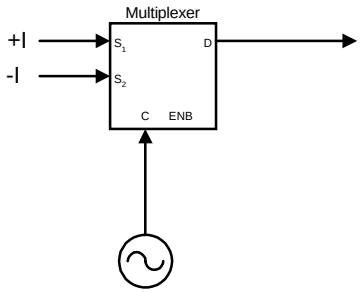


FIGURA 5.11 - Geração de uma onda quadrada digital usando um sinal senoidal de referência

Uma onda quadrada pode ser gerada facilmente, passando um tom de referência por um quantificador de 1 bit. O sinal de 1 bit resultante pode ser usado para controlar a entrada de um multiplexador 2 por 1. Esse esquema é mostrado na fig. 5.11 onde a saída

é tomado da saída do multiplexador. A onde quadrada resultante terá freqüência idêntica ao do sinal senoidal controlante. Ainda, a amplitude da onda pode ser programada ajustando-se os valores de I e -I nas entradas do multiplexador. Finalmente, a fase da onda quadrada é relacionada com a do tom de referência e assim, pode ser ajustada alterando-se a fase da senóide controlante.

5.2.2 Onda Triangular

Passando-se a onda quadrada descrita acima por um integrador digital irá resultar numa onda triangular digital. Um circuito realizando tal função é dado na fig. 5.12. A onda triangular na saída terá uma freqüência igual à do sinal senoidal de referência. Assim como a onda quadrada, a onda triangular tem a fase relacionada com a fase do sinal de referência, assim, qualquer alteração na fase do sinal de referência provoca uma alteração na fase da onda triangular. Além disso o valor inicial da onda triangular pode ser estabelecido ajustando-se o valor do registrador na fig. 5.12.

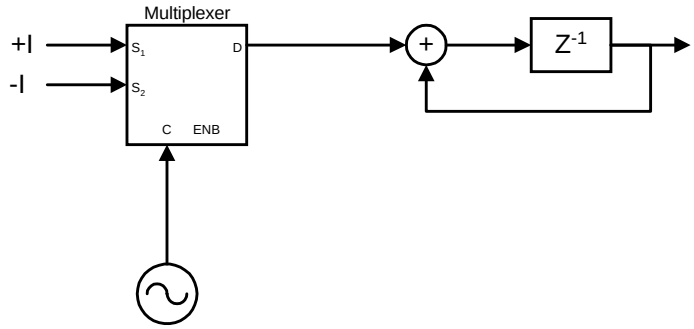


FIGURA 5.12 - Um gerador de onda triangular passando uma onda quadrada por um integrador

5.2.3 Onda dente de serra

A fig. 5.13 ilustra a maneira na qual uma onda dente de serra pode ser construída. A constante I é alimentada através de um integrador digital o qual é periodicamente resetado quando o sinal de referência passa por zero aumentando. Mais uma vez, a onda resultante terá uma freqüência idêntica ao do sinal de referência e fase similar aos sinais anteriores.

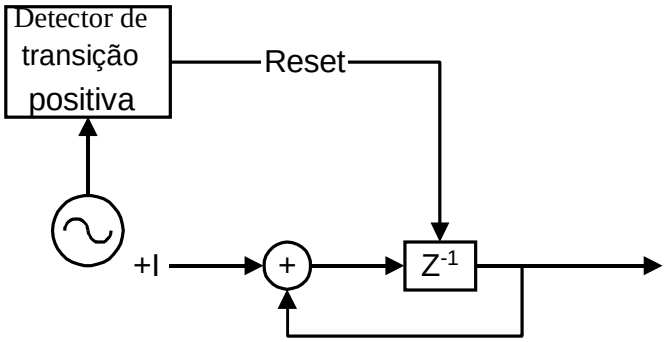


FIGURA 5.13 - Um gerador dente de serra

A fig. 5.14 mostra cada um dos três sinais plotados com sua respectiva senóide de referência. Apesar de todos os sinais terem sido gerados com amplitudes iguais ao sinal de referência, eles podem ser ajustados arbitrariamente e não dependem da amplitude do sinal de referência.

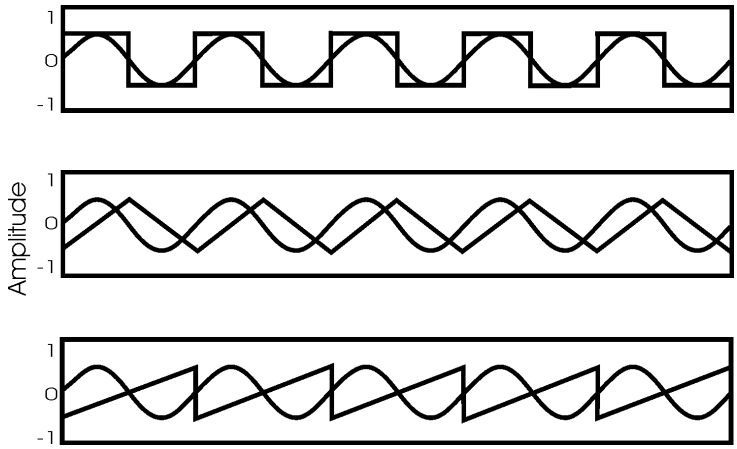


FIGURA 5.14 - Várias formas de onda geradas usando uma senóide de referência

5.3 Um gerador de funções eficiente

Como todos os blocos descritos anteriormente requerem um tom de referência, é apropriado começar com o oscilador digital apresentado inicialmente. Assim, a senóide multibit gerada pelo oscilador pode ser usado em conjunção com os blocos apresentados acima para sintetizar os sinais necessários. Esses sinais podem então ser modulados pelo sigma-delta e filtrados por um passa baixo para produzir os sinais analógicos. Essa configuração é mostrada na fig. 5.15.

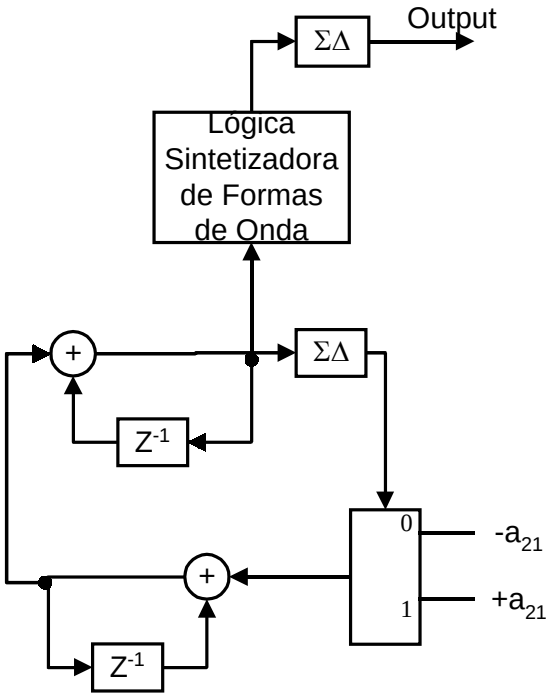
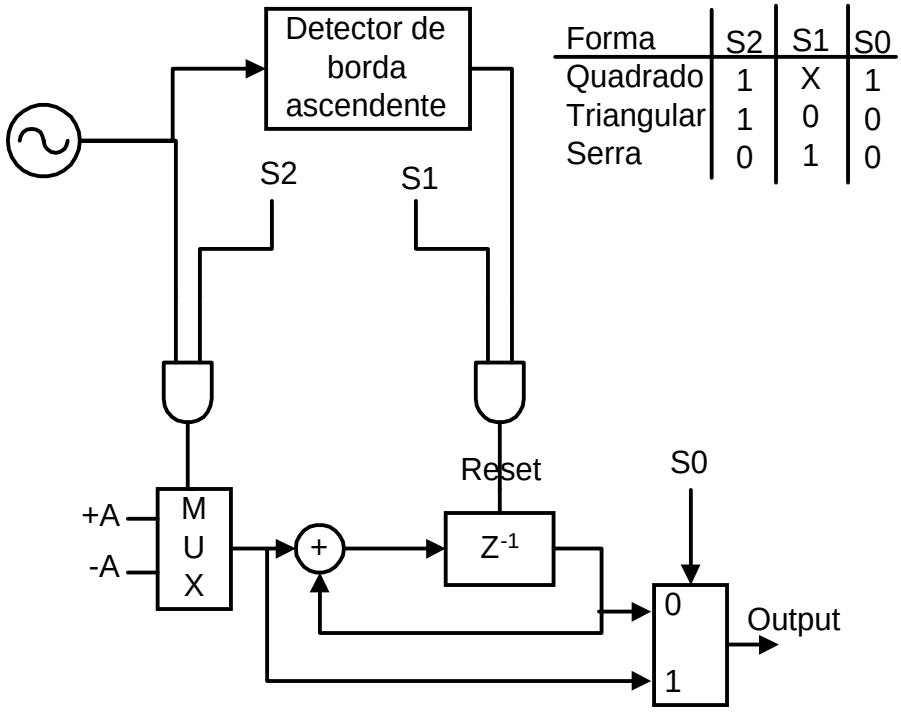


FIGURA 5.15 - Implementação de um gerador de funções baseado em delta-sigma. Uma senóide digital é processada para onda quadrada, triangular ou dente de serra. [ROB95]



Forma	S2	S1	S0
Quadrado	1	X	1
Triangular	1	0	0
Serra	0	1	0

FIGURA 5.16 - Lógica de Síntese de Ondas: a forma desejada pode ser selecionada através da correta seleção dos bits S2, S1 e S0. [ROB95]

Aqui o bloco Lógica Sintetizadora de Formas de Onda é simplesmente uma combinação dos blocos apresentados acima e está ilustrado na fig. 5.16. As várias formas de onda podem ser selecionados através dos bits de controle S2, S1 e S0, como mostrado na tabela da fig. 5.16. Esse bloco recebe um sinal senoidal digital, representado pela senóide dentro do círculo. Esse sinal pode ser modificado gerando sinais quadrados, triangulares ou dente de serra, pela seleção adequada dos bits de controle. Quando S2 e S0 forem um, onda quadrada, o sinal senoidal funcionará como o seletor do multiplexador, alternando entre A e –A, sendo que esse sinal será selecionado pelo segundo multiplexador, não interessando o funcionamento do resto do circuito. Se S2 for um com S1 e S0 sendo zero, a senóide continuará selecionando A ou –A no primeiro multiplexador, valor o qual será continuamente somado e acumulado no registrador, o valor sendo acumulado irá ter características triangulares, e esse sinal será selecionado como saída do circuito. Agora, se S2 e S0 forem zero com S1 sendo um, o valor A será acumulado continuamente até que o circuito detetor de borda ascendente reinicialize o acumulador, esse sinal será selecionado como saída do circuito formando uma onda dente de serra.

Esse capítulo apresenta uma série de circuitos osciladores digitais, capazes de gerar sinais digitais com características senoidais, dente de serra, triangular e quadradas. Esses sinais se forem passados para um circuito conversor D/A serão convertidos em sinais analógicos com as mesmas características dos sinais digitais. Esses circuitos são completamente digitais, o que facilita a sua integração e também a sua programação. Além disso, esses circuitos funcionam perfeitamente em conjunto com conversores D/A sigma-delta. Compostos dessa forma, o sistema inteiro passa a ser digital, necessitando apenas de um filtro passa baixa analógico, o qual pode ser externo. Dessa forma obteremos o circuito como desejado.

6 Implementações e Resultados

Nessa parte do trabalho, será apresentado o CORE processador que foi utilizado na integração e suas várias implementações. Além disso, serão apresentadas as implementações realizadas com alguns dos circuitos discutidos e os resultados obtidos. Essas implementações foram implementadas utilizando-se a linguagem VHDL para a descrição dos componentes digitais. A parte mista do circuito foi implementada, para efeito de simulação na ferramenta Matlab. Isso porque as ferramentas de VHDL disponíveis não eram capazes de realizar uma simulação de sinais mistos, enquanto que usando-se a linguagem própria do Matlab isso se torna possível, podendo-se enxergar a forma de onda resultante após a filtragem analógica.

6.1 Processador RISCO

O Risco originalmente foi desenvolvido como um microprocessador de 32 bits com arquitetura de estilo RISC (Reduced Instruction Set Computer). A seguir é apresentada uma visão geral do processador.

- Microprocessador RISC de 32 bits.
- Todos os dados, instruções e endereços são palavras de 32 bits. A unidade de endereçamento é a palavra, permitindo um acesso a 4 Giga palavras.
- Comunicação com a memória por barramento único de 32 bits, multiplexado de dados e endereços.
- Possui 32 registradores de 32 bits, incluídos nestes o PC (contador de programa), o SP (apontador de pilha), o PSW (palavra de estado do processador) e o ROO (constante 0).
- As instruções são executadas em um ciclo.
- Acessos à memória ocupam mais um ciclo de máquina. As instruções de salto são retardadas de uma instrução (“delayed branch”).
- Possui dois barramentos bifilares internos, permitindo duas leituras concorrentes no Banco de Registradores ou uma escrita. Executa quatro tipos básicos de instruções: Aritmético-Lógica, Salto, Acesso à memória e Sub-rotina. [JUN93]

Para questões de teste, simulação e síntese nos FPGAs disponíveis foram implementados 3 versões do Risco em VHDL. A primeira é a versão completa com todas as características descritas, ou seja, de 32 bits com banco de registradores dual-porta (32bdp), a qual devido ao tamanho final não pôde ser sintetizada nos FPGAs disponíveis nessa universidade. Por causa disso foram implementadas outras duas versões. Uma também de 32 bits, mas dessa vez com um banco de registradores single port (32bsp), simplificando a memória, mas sendo necessário um aumento na parte de controle. A última versão é de 16 bits com banco de registradores dual porta (16bits).

6.1.1 Formatos de Instruções

O Risco é uma máquina que para cada instrução utiliza três endereços (de operandos), usualmente com um operando destino e dois operandos-fonte (os operandos são sempre registradores da UCP ou uma constante presente na palavra de instrução). Existem também instruções com um destino, uma fonte e uma constante de 11 bits, bem

como um destino/fonte e uma constante de 17 bits. Não há palavra com valor imediato após as instruções.

Os três formatos possíveis de instrução estão esquematizados na figura 6.1.

31	30	29..25	24	23	22	21..17	16..12	11	10..6	5..0
T1	T0	C4..C0	APS	F1	F0	DST	FT1	SS2	FT2	...
T1	T0	C4..C0	APS	F1	F0	DST	FT1	SS2	Kp	
T1	T0	C4..C0	APS	F1	F0	DST	Kg			

FIGURA 6.1 – Formatos de instrução

Os campos da palavra de instrução e respectivas funções são descritos a seguir:

- T1, T0: T1 indica se a instrução terá um ciclo extra para acesso à memória (instruções de carga/armazenamento ou de sub-rotina) e T0 define se o campo C4..C0 contém um código de operação (aritmético-lógica ou de carga/armazenamento) ou o código de teste (salto ou sub-rotina).

⇨ 00 – Aritmético-lógica
⇨ 10 – Acesso à memória
⇨ 01 – Salto
⇨ 11 – Sub-rotina

- C4, C3, C2, C1, C0: determinam a operação a ser executada nas instruções aritmético-lógicas e de acesso à memória, ou a condição de teste nas instruções de salto e de sub-rotina.
- APS: indica se a PSW deve ser atualizada ao fim da instrução corrente, ou seja, se APS=1 então os bits negativo, overflow zero e carry da PSW serão atualizados de acordo com o resultado da operação.
- F1, F0, Kg: os bits F1 e F0 indicam se os bits de 16 a 0 contêm uma constante (Kg) de 17 bits como segundo operando da operação, ou o campo FT1/SS2/[FT2 ou Kp], F1, F0 e SS2 indicam também os operandos da instrução.
- DST, FT1: são os endereços de dois dos 32 registradores para destino (DST) e primeiro operando (FT1) da operação.
- SS2, FT2, Kp: SS2 indica se os bits de 10 a 0 significam uma constante (Kp) de 11 bits, ou se os bits de 10 a 6 significam o endereço de 1 dos 32 registradores (FT2) para segundo operando da operação.
- Constantes: de acordo com F1 e F0, a constante presente na palavra de instrução é carregada em um registrador interno temporário para ser o segundo operando. A constante Kp, de 11 bits tem seu bit mais significativo estendido. E a constante Kg de 17 bits pode ser estendido de duas maneiras (Kgh e Kgl). Em Kgh os 16 bits de Kg são copiados nos 16 bits mais significativos de uma palavra, e o bit 17e é copiado nos outros bits. Em Kgl o bit 17 é estendido.

O Formato da instrução no Risco procurou separar a codificação da operação a ser realizada da codificação dos operandos a serem utilizados, permitindo assim facilitar a interpretação das instruções.

No Risco, os bits de 31 a 24 da instrução vão direto para a parte de controle, os bits 23, 22 e 11 vão para uma lógica separada para determinar os operandos da instrução, e os bits de 21 a 0 vão direto para a parte operativa.

Os bits F1 e F0 indicam o tipo do formato dos operandos que a palavra de instrução possui, como mostrado na tab. 6.1.

TABELA 6.1 – Bits de seleção e operandos representados

F1	F0	SS2	Operandos Fonte
0	0	0	FT1 e FT2
0	0	1	FT1 e Kp
0	1	X	R00 e Kgl
1	0	X	DST e Kgh
1	1	X	DST e Kgl

FT1: Contém o número do registrador do primeiro operando fonte
FT2: Contém o número do registrador do segundo operando fonte
SS2: Bit utilizado para definir o formato dos bits de 0 a 10.
Kp: Constante de 11 bits, estende sinal.
Kgl: Constante de 17 bits, estende sinal.
Kgh: Constante de 17 bits, ocupa a parte mais significativa e estende o sinal para os bits menos significativos.

6.1.2 Instruções aritmético-lógicas

O Risco implementa 25 instruções, listadas na tab. 6.2.

TABELA 6.2 – Instruções Aritmético-Lógicas

Instrução	Operação	Comentário
ADD	dest ← fa + fb	Soma
ADDC	dest ← fa + fb + c	Soma Carry
SUB	dest ← fa – fb	Subtração
SUBC	dest ← fa – fb – c	Subtração Carry
SUBR	dest ← fb – fa	Subtração Reversa
SUBRC	dest ← fb – fa – c	Subtração Reversa Carry
AND	dest ← fa & fb	E lógico
OR	dest ← fa | fb	Ou lógico
XOR	dest ← fa ^ fb	Ou exclusivo lógico
RRL	dest ← fa rot fb	Rot. dir . log.
RRLC	dest ← fa rot fb c/ C	Rot. dir . log. carry
RRA	dest ← fa rot fb	Rot. dir . arit.
RRAC	dest ← fa rot fb c/ C	Rot. dir . arit. carry
RLL	dest ← fa rot fb	Rot. esq. log.
RLLC	dest ← fa rot fb c/ C	Rot. esq. log. carry
RLA	dest ← fa rot fb	Rot. esq. arit.
RLAC	dest ← fa rot fb c/ C	Rot. esq. Arit. carry
SRL	dest ← fa dlc fb	Desl. dir. log.
SRLC	dest ← fa dlc fb c/ C	Desl. dir. log. carry
SRA	dest ← fa dlc fb	Desl. dir. arit.
SRAC	dest ← fa dlc fb c/ C	Desl. dir. arit. carry
SLL	dest ← fa dlc fb	Desl. esq. log.
SLLC	dest ← fa dlc fb c/ C	Desl. esq. log. carry
SLA	dest ← fa dlc fb	Desl. esq. arit.
SLAC	dest ← fa dlc fb c/ Carry	Desl. esq. arit. Carry

6.1.3 Parte Operativa

A PO tem número de bits igual a palavra de dados e de endereço da arquitetura, e seus elementos estão distribuídos ao longo dos barramentos. A PO foi projetada e implementada na forma de bit-slice, ou seja, desenvolve-se os componentes para um bit e esses são instanciados várias vezes até formar elementos de 32 bits.

Na tab. 6.3 pode-se ver o resultado da síntese lógica da Parte Operativa.

TABELA 6.3 – Síntese da Parte Operativa

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	41	4*	9216	56	1199	52 (10K40)
32bsp	74	42	1024	8	1086	94 (10K20)
16bits	57	31	1024	8	611	53 (10K20)

* Apenas 4 pinos de saída foram utilizados, isso porque os dados só são acessados por componentes externos através da memória de dados, a qual nessa implementação está dentro do FPGA para possibilitar a simulação do mesmo. Quando a memória for externa, serão necessários ainda 32 pinos para dados e endereço de memória.

6.1.4 Banco de Registradores

O banco de registradores no Risco é formado por 32 registradores de 32 bits cada um. Este é na verdade uma memória SRAM dual-port. Para a implementação foi usado um componente da LPM conhecido como CSDPRAM (Cycle-Shared Dual-Port RAM), ou seja, uma memória RAM que realiza operações de dual-port através do compartilhamento do ciclo de clock.

Na tab. 6.4 pode-se ver o resultado da síntese lógica do Banco de Registradores da versão 32bdp.

TABELA 6.4 – Síntese do Banco de Registradores

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	86	64	1024	8	115	9 (10K20)

6.1.5 Unidade de Constante

A unidade de constante é composta por dois registradores, “const” e “constext”, e uma série de seletores. Essa unidade é responsável pela expansão das constantes Kp e Kg retornando Kpe, Kgl ou Kgh.

Se os bits de seleção (F1, F0 e SS2) forem na forma “001” então a saída será Kpe, quando estes forem “10x” a saída será Kgh, se “x1x” então a saída é Kgl senão a operação é realizada usando dados vindos do banco de registradores.

Na tab. 6.5 pode-se ver o resultado da síntese lógica da Unidade de Constante.

TABELA 6.5 – Síntese da Unidade de Constante

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	22	32	---------	----------	72	12 (10K10)
32bsp	20	32	---------	----------	55	09 (10K10)
16bits	19	16	---------	----------	17	02 (10K10)

6.1.6 Unidade Lógico-Aritmética (ULA)

A unidade lógico-aritmética (ULA) realiza as operações básicas de soma, e-lógico, ou-lógico e ou-exclusivo-lógico, bem como gera os bits negativos, overflow, zero e carry. Controlando-se as entradas A, B (normal ou complementadas de 1) e Carry (0,1, C) da ULA, obtém-se no Risco 15 operações, as quais estão listadas na tab. 6.6

TABELA 6.6 - Operações da ULA

Função	Código	Operação
And	01111	A & B
Or	01110	A | B
Xor	01101	A ^ B
Subrc	01011	~A + B + C
Subr	01001	~A + B + 1
Subc	00111	A + ~B + C
Sub	00101	A + ~B + 1
Add	00000	A + B
Addc	00010	A + B + C

Na tab. 6.7 pode-se ver o resultado da síntese lógica da Unidade Lógico-Aritmética.

TABELA 6.7 – Síntese da Unidade Lógico-Aritmética

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	68	36	---------	----------	319	55 (10K10)
32bsp	68	36	---------	----------	319	55 (10K10)
16bits	36	20	---------	----------	169	29 (10K10)

6.1.7 Unidade do contador de programa

A unidade do contador de programa é composta de dois registradores, PC e PCINC, e do incrementador INC. A função desta unidade é realizar o incremento do registrador PC e a busca de nova instrução.

Essa unidade usa o valor de PC para buscar a nova instrução e realiza o seu incremento no incrementador e o valor é guardado temporariamente em PCINC.

TABELA 6.8 – Síntese da Unidade de Contador de Programa

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	32	32	---------	----------	42	07 (10K10)
32bsp	32	32	---------	----------	42	07 (10K10)
16bits	16	16	---------	----------	20	03 (10K10)

Na tab. 6.8 pode-se ver o resultado da síntese lógica da Unidade de Contador de Programa.

6.1.8 Unidade de Deslocamento

O Risco possui uma unidade de deslocamento composta de uma máquina de estados finitos e de um deslocador, onde a máquina de estados controla a quantidade de vezes que os dados passarão pelo deslocador de forma a realizar as operações necessárias ao que foi pedido pela instrução.

O dado a ser deslocado é sempre o operando fonte 1, que é transferido para o registrador RDA (e RDB recebe 0). O valor do deslocamento ou rotação é fornecido pelos cinco bits menos significativos do operando fonte 2, sendo transferido para o decodificador que seleciona uma das linhas de comando.

Nas instruções de deslocamento e rotação, as operações executadas seguem a seguinte norma geral:

- Deslocamento/rotação aritmético: S (sinal = bit 31) e carry permanecem inalterados; exceto em SRA onde S é copiado para os bits menos significativos.
- Deslocamento/rotação lógico: Carry mantém-se inalterado, S entra na operação.
- Deslocamento/rotação lógico com carry: S e C entram na operação.
- Deslocamento/rotação aritmético com carry: S fica inalterado, C entra na operação.

As instruções da UD estão descritas na tab. 6.9.

TABELA 6.9 – Operações da unidade de deslocamento

Operação	Código
Rot. Dir . log.	10100
Rot. Dir . log. Carry	10101
Rot. Dir . arit.	10110
Rot. Dir . arit. Carry	10111
Rot. Esq. log.	10000
Rot. Esq. Log. Carry	10001
Rot. Esq. arit.	10010
Rot. Esq. Arit. Carry	10011
Desl. Dir. log.	11100
Desl. dir. log. Carry	11101
Desl. Dir. arit.	11110
Desl. dir. arit. Carry	11111
Desl. Esq. log.	11000
Desl. esq. log. Carry	11001
Desl. Esq. arit.	11010
Desl. esq. arit. Carry	11011

Na tab. 6.10 pode-se ver o resultado da síntese lógica da Unidade de Deslocamento.

TABELA 6.10 – Síntese da Unidade de Deslocamento

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	45	33	---------	----------	305	52 (10K10)
32bsp	44	33	---------	----------	330	57 (10K10)
16bits	27	17	---------	----------	203	35 (10K10)

6.1.9 RAM

Para poder testar na implementação do RISCO as instruções de “load” e de “store”, era necessário a existência de uma memória. Por causa disso, foi implementada uma pequena memória RAM utilizando-se o componente LPM_DQ_RAM, que nada mais é do que uma memória “single-port”.

Na tab. 6.11 pode-se ver o resultado da síntese lógica da RAM.

TABELA 6.11 – Síntese da RAM

Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
42	32	8192	66	0	0

6.1.10 Parte Controle

No Risco os elementos da parte operativa (armazenamento, transformação e de comunicação), são basicamente os registradores, as unidades funcionais (ULA, UD e incrementador do PC).

A função do controle é ordenar no tempo a atuação da parte operativa em função das instruções a serem executadas.

A parte de controle do Risco é composta de uma máquina com oito estados somente, sendo assim uma das máquinas mais simples usadas em processadores. Onde cada estado é responsável pela atuação de uma unidade funcional, ou da decodificação da instrução quando necessário.

Na tab. 6.12 pode-se ver o resultado da síntese lógica da Parte de Controle.

TABELA 6.12 – Síntese da Parte de Controle

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	13	18	----------	----------	40	06 (10K10)
32bsp	13	19	----------	----------	51	08 (10K10)
16bits	27	55	----------	----------	173	30 (10K10)

Na tab. 6.13 pode-se ver o resultado da síntese lógica do Risco.

TABELA 6.13 – Síntese do Risco

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bdp	2	8	9472	38	1259	25 (10K100)
32bsp	66	49	1024	8	1142	99 (10K20)
16bits	18	30	5120	41	785	68 (10K20)

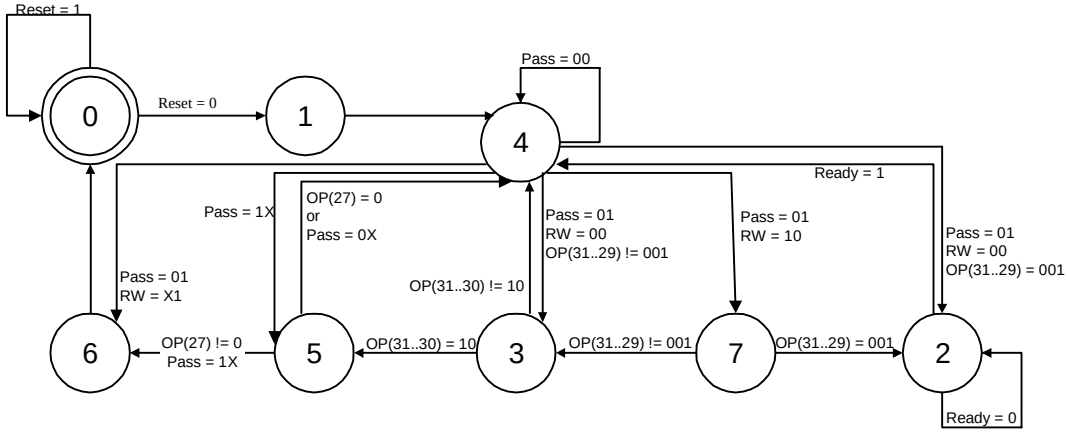


FIGURA 6.2 – Máquina de estados da parte de controle do microprocessador Risco

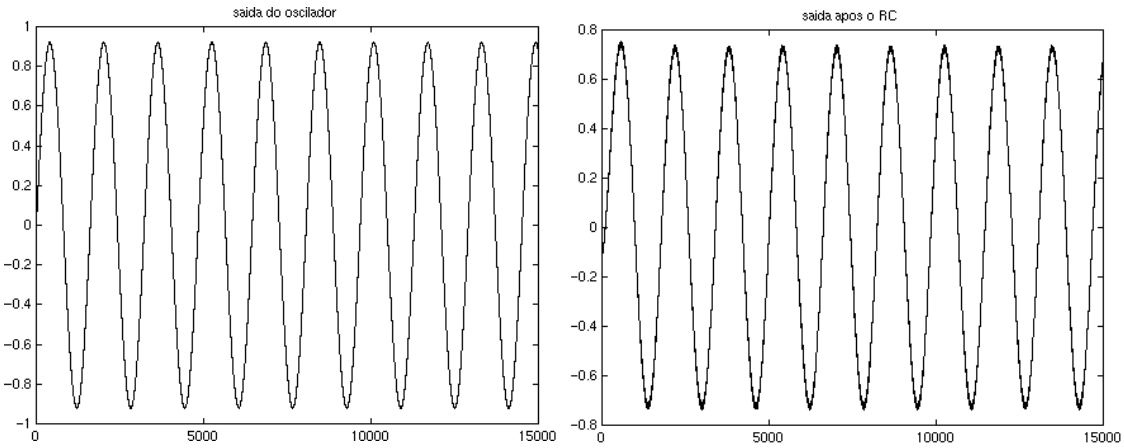
A máquina de estados que compõe a parte de controle do microprocessador Risco, mostrada na fig. 6.2, quando este não tem pipeline, apresenta sete estados. O estado inicial é sempre o estado zero, o qual também é o estado assumido quando acontece uma reinicialização. As transições mostradas na fig. 6.2 são todas as transições possíveis nessa máquina. Quando nada estiver escrito próximo à transição, isso significa que uma vez alcançado o estado onde a seta se origina, a transição ocorrerá independente de qualquer condição. Sempre que uma transição depender de uma condição esta estará escrita da mesma. Por exemplo, a máquina se inicializa no estado ‘0’ e permanecerá neste enquanto o “reset” for um. Quando “reset” for zero, a máquina irá para o estado ‘1’ e no próximo ciclo irá para o estado ‘4’ independente de qualquer outra condição.

6.2 Gerador Digital de Senóides

Foram implementadas várias versões do sistema oscilador, sendo que inicialmente foi criado uma versão 16 bits em VHDL, em seguida foram criadas algumas versões em Matlab e finalmente foi criada uma versão 32 bits também em VHDL. Como será visto na continuação desse trabalho, ao limitar o gerador a 16 bits, restringe-se a resolução da senóide a ser gerada e as freqüências que essa pode ter, decidiu-se usar, na implementação final desse trabalho a versão de 32 bits do gerador digital de senóides, apesar das versões de 16 bits ou mesmo a do Roberts ocuparem menor área.

A versão 16 bits em VHDL foi criada para que fosse possível realizar um teste do sistema. Ao utilizar essa versão notou-se que a quantidade de freqüências que poderiam ser geradas eram poucas e ainda por cima freqüências muito altas e de difícil separação em relação ao sinal portador.

Para determinar qual seria o tamanho ideal para o oscilador, de forma a conseguir gerar várias freqüências e especialmente freqüências mais facilmente separáveis da portadora resolveu-se utilizar o Matlab e criar varias descrições, dessa forma era mais rápida a tarefa de desenvolver e testar os circuitos criados. Além disso, utilizando-se o Matlab é possível realizar simulações nas quais não apenas se simula o oscilador digital mas também o filtro analógico. A descrição Matlab é mostrada no Anexo A.



(a) (b)

FIGURA 6.3 – Plotagens realizadas através do Matlab: (a) valores com 32 bits gerados pelo oscilador digital (b) onda após passagem por circuito sigma-delta e filtro passa-baixo

Na fig. 6.3 são mostradas duas plotagens de valores gerados na ferramenta Matlab. Na primeira (6.3 (a)) os valores são pontos gerados pelo oscilador digital de 32 bits com uma amplitude de 0,9. Esses pontos são então passados por um circuito sigma-delta de segunda ordem e por um filtro passa-baixa, realizando-se assim uma conversão digital-analógica. A onda resultante dessa conversão é mostrada na fig. 6.3 (b). Ao passar pelo circuito sigma-delta e pelo filtro a onda foi atenuada e defasada, sendo que sua amplitude ficou reduzida em 22%. Essa atenuação será notada ao compararmos a escala, no lado esquerdo das imagens.

Utilizando-se a fórmula apresentada no capítulo anterior, considerando que o circuito a ser utilizado trabalha com 32 bits e que a freqüência de oscilação da onda portadora seja de 1,5734375 MHz, foram calculadas todas as freqüências que seria possível de se gerar, mostradas na tab. 6.14. Caso a quantidade de bits utilizada seja

menor deve-se lembrar que o tamanho máximo do fator $a_{12}a_{21}$ será diferente e proporcional à quantidade de bits. Aumentar a quantidade de bits não irá causar um aumento nas freqüências geradas pois quando o fator $a_{12}a_{21}$ passar de –52 as freqüências que podem ser geradas são tão baixas que se tornam desprezíveis e dificilmente calculáveis. Ao incrementar o valor de um dos parâmetros, o outro deve ser decrementado de fator igual para manter a mesma freqüência de geração. Não se esquecendo que isso afeta a amplitude do sinal gerado, de acordo com a fórmula mostrada anteriormente no capítulo 5 desse texto.

TABELA 6.24 – Freqüências que podem ser geradas pelo circuito implementado

Fator X+Y ($a_{12}a_{21}=2^{x+y}$)	X($a_{12}=2^{x}$)*	Y($a_{21}=2^{y}$)*	Freqüência (% da Fos.)	Freqüência (HZ)**
-52	-31..-21	-21..-31	0,0000002%	0,004
-51	-31..-20	-20..-31	0,0000003%	0,005
-50	-31..-19	-19..-31	0,0000005%	0,007
-49	-31..-18	-18..-31	0,0000007%	0,011
-48	-31..-17	-17..-31	0,0000009%	0,015
-47	-31..-16	-16..-31	0,0000013%	0,021
-46	-31..-15	-15..-31	0,000002%	0,030
-45	-31..-14	-14..-31	0,000003%	0,042
-44	-31..-13	-13..-31	0,000004%	0,060
-43	-31..-12	-12..-31	0,000005%	0,084
-42	-31..-11	-11..-31	0,000008%	0,119
-41	-31..-10	-10..-31	0,000011%	0,169
-40	-31..-9	-9..-31	0,000015%	0,239
-39	-31..-8	-8..-31	0,000021%	0,338
-38	-31..-7	-7..-31	0,00003%	0,478
-37	-31..-6	-6..-31	0,00004%	0,675
-36	-31..-5	-5..-31	0,00006%	0,955
-35	-31..-4	-4..-31	0,00009%	1,351
-34	-31..-3	-3..-31	0,00012%	1,911
-33	-31..-2	-2..-31	0,00017%	2,702
-32	-31..-1	-31..-1	0,00024%	3,821
-31	-31..0	0..-31	0,0003%	5,404
-30	-31..1	1..-31	0,0005%	7,642
-29	-31..2	2..-31	0,0007%	10,808
-28	-31..3	3..-31	0,0010%	15,284
-27	-31..4	4..-31	0,0014%	21,615
-26	-31..5	5..-31	0,002%	30,569
-25	-31..6	6..-31	0,003%	43,231
-24	-31..7	7..-31	0,004%	61,138
-23	-31..8	8..-31	0,005%	86,462
-22	-31..9	9..-31	0,008%	122,276
-21	-31..10	10..-31	0,011%	172,924
-20	-31..11	11..-31	0,016%	244,551
-19	-31..12	12..-31	0,022%	345,848
-18	-31..13	13..-31	0,031%	489,102
-17	-31..14	14..-31	0,044%	691,695
-16	-31..15	15..-31	0,062%	978,205
-15	-31..16	16..-31	0,088%	1383,392
-14	-31..17	17..-31	0,124%	1956,414
-13	-31..18	18..-31	0,176%	2766,794
-12	-31..19	19..-31	0,249%	3912,858
-11	-31..20	20..-31	0,352%	5533,673
-10	-31..21	21..-31	0,497%	7825,955
-9	-31..22	22..-31	0,703%	11068,022
-8	-31..23	23..-31	0,995%	15653,821
-7	-31..24	24..-31	1,407%	22141,453
-6	-31..25	25..-31	1,991%	31322,960
-5	-31..26	26..-31	2,817%	44326,328
-4	-31..27	27..-31	3,989%	62769,280
-3	-31..28	28..-31	5,657%	89004,704
-2	-31..29	29..-31	8,043%	126552,559
-1	-31..30	30..-31	11,503%	180987,367
0	-31..31	31..-31	16,667%	262239,583
1	-30..31	31..-30	25%	393359,375

Ao implementar as diversas versões do circuito sigma-delta (16 bits, 32 bits, e a proposta diretamente por Roberts [ROB95]) foi-se notando algumas diferenças. Por exemplo os dois primeiros trabalham sempre com a mesma quantidade de bits, ou seja, na versão 16 bits todos os circuitos trabalham diretamente com 16 bits, a de 32 bits sempre tem 32 bits. Já no circuito proposto por Roberts, ele utiliza vários tamanhos de somadores indo desde 4 a até 20 bits. Por causa dessa diferença pode-se notar, na tab. 6.15, que os circuitos apresentam discrepância de tamanho. Tanto o primeiro quanto o último deveriam ser de 16 bits, mas por causa dessas particularidades, o primeiro ocupa uma área um pouco maior.

No caso dos circuitos osciladores digitais implementados, cujos resultados são mostrados na tab. 6.16, pode-se notar que existe uma diferença muito pequena entre as versões 16 e 32 bits, isso se deve à utilização de multiplicadores no primeiro enquanto que no segundo eles foram substituídos por deslocadores que além de serem menores são mais rápidos. No caso do circuito do Roberts, que também é de 16 bits, deve-se lembrar que já no sigma-delta houve uma diferença significativa. Como o sigma-delta faz parte desse outro circuito essa diferença é repassada, além disso, qualquer bit que se reduza em algum componente conta como vários a menos no circuito final, pois não apenas se ganha aquele que se reduziu como ainda a quantidade de células lógicas utilizadas para conexão passa a ser menor. [ZIM00a]

TABELA 6.15 – Tabela comparativa dos circuitos sigma-delta implementados

Implementação	Células Lógicas	Células Utilizadas (%)
Sigma-delta 16 bits	121	10
Sigma-delta 32 bits	213	18
Sigma-delta (Roberts)	93	8

TABELA 6.16 – Tabela comparativa dos circuitos osciladores , com os sigma-deltas, digitais implementados

Implementação	Células Lógicas	Células Utilizadas (%)
Gerador 16 bits	343	29 %
Gerador 32 bits	360	31 %
Versão do Roberts	174	15 %

Olhando-se os resultados apresentados nas tabs. 6.15 e 6.16, o leitor pode ser levado a crer que o circuito implementado pelo Roberts, que por ter toda a variação de tamanho já descrita, indo de 4 a 20 bits no mesmo circuito, ocupando áreas muito menores do que os outros, que esse seja melhor que os outros. Isso pode ser considerado verdade se esquecermos que estamos fazendo um circuito para ser utilizado em conjunto com um microprocessador de 32 bits. Levando isso em conta, e ainda pensando nas vantagens quanto à facilidade da testabilidade que será abordado no item 6.3, quando se tem vários componentes equivalentes e de mesmo tamanho, foi decidido utilizar o circuito sigma-delta e o circuito oscilador de 32 bits. O mesmo circuito que foi implementado em VHDL e Matlab e ainda usado na geração do layout final do circuito.

6.3 BIST do Gerador Digital de Senóides

Para que esse sistema, que gera sinais de teste para componentes analógicos, possa ficar completo é necessário que ele também seja testável. Por isso, foi inserido nele um sistema de BIST.

De acordo com [McC85], uma das formas mais econômicas de se fazer o teste de um sistema, quando este apresenta hardware replicado, é colocar entradas idênticas em cada parte replicada e comparar as saídas. Como o gerador apresenta vários somadores, todos de 32 bits, resolveu-se aplicar esse método.

Pelas tabs. 6.17, 6.18 e 6.19, referentes à comparação dos circuitos com e sem teste foi notado que todos eles apresentaram um acréscimo no número de pinos tanto de entrada com de saída. Isso se deve ao fato do teste ser realizado off-line, ou seja, fora do funcionamento normal do circuito. Assim é necessário que se avise o circuito quando começar a realização do teste e que esse informe ao usuário se o teste está bom ou não.

Pela tab. 6.17, onde é realizada a comparação dos circuitos geradores de sinais senoidais com e sem teste nota-se que houve um ligeiro incremento de área e consequentemente um decremento na freqüência de trabalho do circuito. Isso já era de se esperar pois qualquer circuito BIST acarreta um acréscimo de componentes e consequentemente um aumento no caminho crítico do circuito fazendo com que tenha que trabalhar mais lentamente. [ZIM01]

TABELA 6.37 – Comparação de tamanho entre o gerador, sem o sigma-delta, sem teste e com teste

Versão	Pinos de entrada	Pinos de saída	Células Lógicas	Células Utilizadas (%)	Freqüência (MHz)
Sem teste	2	32	147	25	9,48
Com teste	3	33	224	38	8,91

Pode-se notar também que, o circuito sigma-delta sofreu um acréscimo maior de área, tab. 6.18, do que o gerador de senóides, tab. 6.17, isso se deve ao fato do modulador sigma delta ter em sua estrutura 4 somadores e o gerador apresentar apenas dois. Como os vetores de teste são os mesmos, independentemente da quantidade de somadores, o que faz diferença nesse tamanho é a quantidade de conexões necessárias para levar os vetores aos respectivos circuitos e ainda os comparadores que irão dizer se o circuito está funcionando direito. No caso do gerador de senóides apenas o resultado das duas somas é comparado, já no caso do modulador sigma-delta o resultado de cada uma das quatro somas é comparado entre si e se um deles for divergente o circuito deve indicar o problema.

TABELA 6.18 – Comparação de tamanho entre o modulador sigma-delta sem teste e com teste

Versão	Pinos de entrada	Pinos de saída	Células Lógicas	Células Utilizadas (%)	Freqüência (MHz)
Sem teste	34	1	213	18	8,56
Com teste	35	2	383	33	6,73

No sistema completo, formado pela união do gerador de senóides com o modulador sigma-delta, nota-se que, apesar do acréscimo da área pela inserção do teste,

tab. 6.19, e também na contagem de pinos, não houve alteração na freqüência de operação do circuito como um todo. Isso se deve porque, apesar de cada parte do circuito ter sua freqüência reduzida, nenhum deles ficou abaixo da freqüência original, sendo que essa já era mais baixa, pois o ponto crítico desse circuito acabou sendo as conexões entre um componente e outro, não sendo alterado pela inclusão do teste.

TABELA 6.19 – Comparação de tamanho entre o sistema completo, gerador e sigma-delta, sem teste e com teste

Versão	Pinos de entrada	Pinos de saída	Células Lógicas	Células Utilizadas (%)	Freqüência (MHz)
Sem teste	2	1	360	31	6,66
Com teste	3	2	611	53	6,66

O circuito de BIST utilizado não é capaz de dizer em qual componente o erro se encontra, se é no gerador ou no modulador, e menos ainda em qual somador de cada um dos componentes. Ele apenas avisa que foi encontrado algum erro no circuito. Caso o circuito não apresenta-se nenhum erro o sinal de erro permanece em zero. Como a idéia do BIST não é a identificação de qual o erro e muito menos o seu isolamento, bastando realizar uma detecção, esse circuito é considerado satisfatório. Caso seja necessário identificar ou mesmo isolar o componente com erro existem diversos outros tipos de circuitos que poderiam ser usados, sendo estes mais complexos.

6.4 Sistema completo (Risco e Gerador Digital)

Para realizar a integração do Processador Risco com o gerador, algumas pequenas alterações tiveram que ser efetivadas. No caso do Risco foi necessário a colocação de algumas portas de comunicação, sendo uma para receber valores do registrador do gerador de sinais, uma para gravar valores nesse registrador e outro para programar a quantidade de deslocamentos que serão realizados no gerador. Além disso, dois registradores do banco de registradores do Risco foram retirados desse banco, sendo que agora o endereço R29 e R30 são ambos pertencentes ao gerador. Sendo, respectivamente, a programação dos deslocadores e do registrador temporário do gerador.

O efeito dessas modificações na descrição do Risco ficam mais claras quando se olha os resultados das sínteses apresentadas na tab. 6.20. Nota-se que houve um acréscimo na quantidade de pinos de entrada e de saída, sendo que para entrada foram acrescidos 32 pinos (valor de um registrador de 32 bits) e para a saída foram acrescidos 64 pinos (dois registradores de 32 bits). Não houve alteração na quantidade de bits de memória utilizados, mas a quantidade de células lógicas também sofreu um acréscimo devido a lógica adicional necessária para o endereçamento do banco de registradores.

No caso do gerador de sinais foram incluídas algumas portas também para permitir a comunicação já descrita, e exemplificada na fig. 6.4. Além disso também foi adicionado ao gerador a instrução de carga. Quando nessa instrução o registrador receberá o valor que vem da porta de programação e também os deslocadores serão programados com valores que vem do Risco. No caso de uma instrução de reset, o Risco irá retornar ao estado inicial e o gerador retorna à programação inicial tanto no registrador quanto nos deslocadores, reiniciando todo o processo.

O efeito das modificações na descrição do gerador de sinais pode ser notada quando se olha a tab. 6.21. Nota-se que houve um acréscimo na quantidade de pinos de entrada e de saída, sendo que para entrada foram acrescidos 43 pinos (valor de um

registrador de 32 bits, 5 bits de controle para cada deslocador e mais um para o sinal de load) e para a saída foram acrescidos 32 pinos (um registrador de 32 bits). A quantidade de células lógicas sofreu um acréscimo devido a lógica adicional necessária para a leitura e gravação do valor do registrador, e para a gravação dos deslocadores. Além disso, o circuito original usava deslocamento fixo, já essa versão programável usa deslocadores programáveis e por isso mais complexos e maiores. Também devido a esse aumento na complexidade e no tamanho, houve uma perda no desempenho do circuito, pois aumentou-se o caminho crítico desse.

TABELA 6.20 – Síntese do Risco simples e modificado para receber o gerador

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)
32bsp	66	49	1024	8	1142	99 (10K20)
32bsp	98	113	1024	8	1340	77 (10K30)

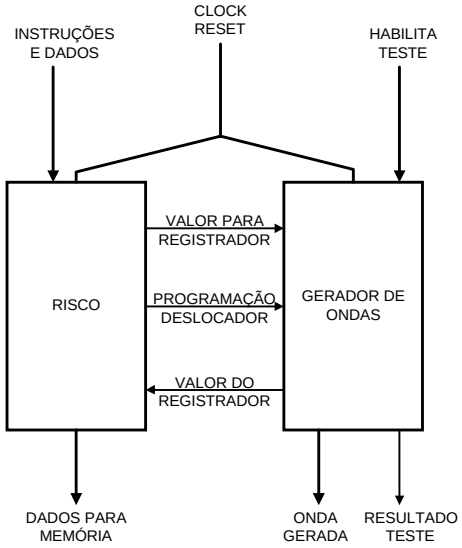


FIGURA 6.4 – Comunicação entre o processador Risco e o gerador de sinais

TABELA 6.21 – Síntese do gerador de sinais com teste e preparado para ser incluído no Risco

Versão	Pinos de entrada	Pinos de saída	Células Lógicas	Células Utilizadas (%)	Freqüência (MHz)
Com teste	3	2	611	53	6,66
Com teste e preparado	46	34	960	83	4,71

Como o sistema completo é formado pela união do Risco com o gerador de sinais, foi feita uma estimativa do seu tamanho e da sua freqüência de operação em um FPGA da Altera. Essa estimativa foi feita realizando-se a soma dos dados adquiridos para cada parte do circuito, cujo resultado são 2300 células lógicas, como mostrado na primeira linha da tab. 6.22. E a freqüência estimada era de menos de 5 MHz, isso por que se for olhar só o gerador ele roda a 4,7 MHz, ou seja, era esperado que, no máximo, as freqüências se igualassem.

Ao realizar a síntese completa do circuito, notou-se, como mostrado na segunda linha da tab. 6.22 que a área ocupada foi menor (1991 células lógicas) e que seu desempenho também foi melhor (6,7 MHz). A diferença na área se explica pelo fato de que o Altera utiliza suas células para realizar a comunicação entre os módulos de desses

com as portas de entrada e saída. Como o número de portas de entrada e saída foi reduzido (não sendo a soma simples das portas de cada componente) e que os componentes podem ser colocados de forma diferente dentro das pastilhas de silício economizando-se área de interconexão, fica explicado a diferença encontrada na área. Também pelas portas de entrada e saída se explica a diferença na freqüência, pois como essa costuma ter alta impedância é normal que seu funcionamento seja um pouco mais lento que do resto do circuito, dessa forma, quanto menos se tiver que utilizar as portas, ou seja, mais coisas dentro da pastilha, menor é o atraso da comunicação.

TABELA 6.22 – Síntese do projeto completo (Risco e gerador)

Versão	Pinos de entrada	Pinos de saída	Bits de memória	Memória utilizada (%)	Células lógicas	Células utilizadas(%)	Freqüência (MHz)
Estimativa	67	51	1024	----------	2300	----------	<5,00
Sistema completo	67	51	1024	6	1991	86 (10K40)	6,73

6.5 Implementação Standard-cell do sistema

A implementação do leiaute foi realizada utilizando-se uma biblioteca de células fornecida pela fábrica AMS (Austria Mikro Systeme International AG), com tecnologia 0,8 μm, e através da ferramenta Mentor Graphics, a partir do VHDL. A implementação em VHDL foi verificada através de simulações no ambiente Altera, através das formas de onda, tanto para o processador Risco, quanto o sistema gerador de sinais (apenas parte digital). Além disso, o sistema gerador de sinais também foi testado, tanto sua parte digital quanto a parte analógica, numa implementação em Matlab do circuito que já tinha sido descrito e testado no ambiente Altera. Como, para o leiaute, foi utilizado uma biblioteca de células fornecida pela fábrica AMS, já plenamente testada, não foi feita uma verificação do resultado da síntese. Apenas foi realizado um DRC, para verificar que nenhuma regra de design tenha sido violada.

Para realizar a geração foi necessário fazer uma alteração no código, ou seja, transformar o banco de registradores que inicialmente tinha sido implementado usando-se uma memória de FPGA através de uma biblioteca LPM para um banco de registradores formados apenas por *flip-flops*, uma mudança puramente na forma como foi implementado, mas que não altera o funcionamento. Isso foi necessário porque a biblioteca de células não é capaz de gerar uma memória. Justamente por usar apenas *flip-flops*, o banco de registradores acabou sendo o maior componente sintetizado, tanto em área como em número de transistores.

Os leiautes gerados podem ser vistos no Anexo B. A tab. 6.23 mostra o resultado da geração automática de leiautes para o microprocessador Risco, considerando cada parte em separado, suas dimensões (e área), número de transistores e a densidade do circuito. Já a tab. 6.24 mostra o resultado da geração automática de leiautes para o gerador de sinais, também considerando cada parte dele e apresentando as mesmas informações da tab. 6.23. Além disso, a tab. 6.24 também mostra os dados referentes à implementação do Risco junto com o gerador de sinais.

Como pode ser visto por comparação das duas tabelas, tanto o somador de 1 bit quanto o somador de 32 bits estão presentes em ambas, isso porque esse são circuitos básicos que fazem parte de componentes de ambos os circuitos.

Na tab. 6.23 são apresentados os componentes do Risco, em primeiro lugar aparecem os componentes da parte operativa, ou seja, a unidade lógica-aritmética, unidade contador de programa, unidade de deslocamento, unidade de constantes e banco de registradores. Desses o componente que ocupou maior área, teve o maior número de transistores e também a maior densidade foi justamente o banco de registradores por ser

feito com flip-flops ao invés de se usar uma memória comum. Logo a seguir são apresentados os macro-blocos PC (parte de controle) e PO (parte operativa), além do circuito completo do Risco. Pode-se notar que a densidade média dos componentes da parte operativa e do microprocessador Risco ficou basicamente a mesma, sendo muito influenciado pelo banco de registradores, o qual teve a maior densidade, sendo responsável por 86% do número total de transistores e por 55% da área ocupada. A parte de controle, a qual é extremamente simples foi responsável por aproximadamente 1% da área e do número de transistores final do Risco.

TABELA 6.23 – Dados da geração standard-cell do Microprocessador Risco

Componente	Dimensões(0.8 μm) Largura	Altura	Área (mm^2)	Número de transistores	Densidade (transistores/mm^2)
Soma 1 bit	187,60	136,00	0,026	30	1153,846
Soma 32 bits	577,60	505,70	0,292	960	3287,671
ULA 32 bits	1075,60	1211,20	1,303	5635	4324,635
UPC	562,60	479,60	0,270	960	3555,556
U. Desloc.	778,60	849,90	0,662	2890	4365,559
U. Const.	424,60	271,70	0,115	310	2695,652
Registradores	2920,60	3246,50	9,482	77068	8127,821
P.O.	3721,60	4541,30	16,901	88431	5232,294
P.C.	472,60	450,90	0,213	805	3779,343
Risco	3850,60	4416,60	17,007	89236	5247,016

Na tab. 6.24 são apresentados os componentes do gerador de sinais. Além dos componentes comuns com o microprocessador Risco (somador de 1 e 32 bits) também aparece um outro componente básico para esse circuito, um deslocador (shift) o qual irá nesse circuito realizar as operações de multiplicação necessárias para a geração da senóide. Esse circuito foi composto por uma sucessão de multiplexadores resultando em 1860 transistores ocupando aproximadamente meio milímetro quadrado.

TABELA 6.24 – Dados da geração standard-cell do gerador de sinais analógicos

Componente	Dimensões(0.8 μm) Largura	Altura	Área (mm^2)	Número de transistores	Densidade (transistores/mm^2)
Soma 1 bit	187,60	136,00	0,026	30	1153,846
Soma 32 bits	577,60	505,70	0,292	960	3287,671
Shift	667,60	633,00	0,423	1860	4397,163
Gerador Sem.	1330,60	1567,00	2,085	9590	4599,520
Mod. Σ-Δ	1309,60	1416,20	1,855	8832	4761,186
Sistema	1843,60	2197,40	4,051	18422	4547,519
Risco+Sist.	4129,60	4963,20	20,496	107658	5252,635

O gerador de sinais foi composto por dois macro-blocos, o gerador de senóides e o modulador sigma-delta sendo que o primeiro teve uma densidade de aproximadamente 4600 transistores por milímetro quadrado, representando 52% do circuito final, e o segundo teve mais de 4700 transistores por unidade de área, ou 48% do circuito final. O sistema gerador de sinais ficou com uma densidade menor do que cada um dos componentes que o compunha devido a uma perda de área para a interconexão dos componentes, mesmo assim ele ficou com mais de 4500 transistores por milímetro quadrado.

A união do microprocessador Risco, o qual teve uma área de 17 milímetros quadrados e do sistema com 4 milímetros quadrados, acabou apresentando uma otimização quanto à área, desperdiçando-se quase que nenhum espaço em conexão entre os dois componentes. Dessa forma esse circuito teve uma área final de 20,5 milímetros quadrados com uma densidade de 5252 transistores por milímetro quadrado com 107658 transistores ao todo. Deve-se lembrar que todos esses dados foram conseguidos através da geração automática do leiaute à partir de VHDL. Isso significa que caso seja realizado um leiaute "*full-custom*" a área pode ser reduzida devido às interconexões e também as portas podem ser mais otimizadas (quanto a área). Além disso, caso seja feito um leiaute "*full-custom*" poderia ser feito as memórias, usando-se SRAM ao invés de *flip-flops* e os deslocamentos poderiam ser realizados através de um *barrel-shifter*. Essas duas mudanças causariam um redução significativa na área final do circuito, possivelmente um aumento de desempenho e principalmente um aumento na densidade do circuito.

Nesse capítulo foi apresentado o Core utilizado na integração do sistema, com suas várias versões, de acordo com as necessidades de projeto, variando desde 16 bits a até 32 bits com banco de registradores porta dupla (permite dois acessos simultâneos). A versão 16 bits foi feita na tentativa de implementar um sistema em um dos FPGAs disponíveis nessa universidade (Flex 10K20). As versões de 32 bits, sendo uma com banco de registradores porta dupla e a outra com banco de registradores porta simples, foram implementadas seguindo a estrutura apresentada na especificação original do Risco realizada por [JUN93], sem o uso de pipeline. A versão com o banco de registradores porta simples se decorre da tentativa de colocar o processador em um FPGA como o citado acima.

Além disso foram apresentadas nesse capítulo as implementações realizadas do sistema gerador de sinais analógicos, tanto no ambiente MaxPlusII da Altera quanto em Matlab, para validar a integração do circuito digital com o filtro analógico. Também foi mostrado a integração do sistema gerador de sinais com o respectivo Core, transformando alguns registradores de uso geral do Core em registradores específicos para a programação do sistema gerador de sinais. Nas simulações, em especial a simulação em Matlab pois nessa considera-se também os componentes analógicos, pode-se perceber que o sistema realmente é capaz de gerar os sinais desejados. Pela falta de uma ferramenta que fizesse a checagem para equivalência lógica e pela falta de tempo disponível para realizar uma verificação do resultado da síntese em leiaute, nenhuma técnica foi utilizada para verificar a correção do circuito gerado nesse nível, a não ser uma varredura de DRC para garantir que nenhuma regra de leiaute tenha sido violada, como dito anteriormente.

7 Conclusão

Este trabalho se propôs a implementar um processador risc 32 bits chamado de Risco, que tinha uma arquitetura previamente definida em outra dissertação, junto com um gerador de sinais de testes analógicos. Ambos deveriam ser implementados como Cores sendo dessa forma facilmente integrados entre si, e com outros componentes que fossem necessários, tendo o mínimo possível de componentes analógicos.

No segundo capítulo foram apresentadas várias arquiteturas clássicas para geração de sinais analógicos. As primeiras arquiteturas mostradas são circuitos analógicos, sendo que algumas delas fazem uso de cristais. Infelizmente esses circuitos não atingem as metas do trabalho, pois além da integração de circuitos analógicos, com ou sem cristais, não ser tão simples quanto a integração de circuitos digitais ainda existe o problema desses não serem facilmente programáveis quanto a freqüência e amplitudes que serão geradas. No final foi apresentada uma alternativa para o uso de circuitos analógicos que é a realização do oscilador através de circuitos digitais, passando o sinal gerado por um conversor D/A. Essa idéia foi aproveitada, mas com uma arquitetura diferente da apresentada, isso porque esta gera o sinal através de valores gravados em uma memória. Esse tipo de circuito, apesar de ser de fácil integração apresenta uma área muito grande, quando comparado a circuitos como os sigma-deltas.

No terceiro capítulo foram apresentados alguns circuitos conversores D/A, necessários para que se possa realizar a geração dos sinais analógicos através de circuitos digitais. Esses circuitos são formados por componentes analógicos, algo que se quer evitar, pois estes não são facilmente integráveis e nem ajustáveis sendo difíceis de serem controlados.

No capítulo seguinte foi apresentada toda uma classe de conversores D/A que, atingem os requisitos do sistema que seria implementado. Além de serem pequenos, ainda por cima são quase totalmente digitais, facilitando a integração em silício. Para esses circuitos o único bloco analógico necessário é um filtro passa-baixa, o qual não precisa se encontrar integrado, isso elimina a necessidade de realizar uma integração com componentes analógicos. Esses circuitos por serem seriais não apresentam uma grande velocidade impondo assim algumas restrições de desempenho ao sistema quanto um todo. No caso do sistema proposto isso não é um problema pois não foi colocado nenhum requisito de desempenho para este.

O quinto capítulo apresenta uma série de circuitos osciladores digitais, capazes de gerar sinais digitais com características senoidais, dente de serra, triangular e quadradas. Esses sinais se forem passados para um circuito conversor D/A serão convertidos em sinais analógicos com as mesmas características dos sinais digitais. Esses circuitos são completamente digitais, o que facilita a sua integração e também a sua programação. Além disso, esses circuitos funcionam perfeitamente em conjunto com conversores D/A sigma-delta. Compostos dessa forma, o sistema inteiro passa a ser digital, necessitando apenas de um filtro passa baixa analógico, o qual pode ser externo. Dessa forma obteremos o circuito como desejado.

Por fim foi apresentado o Core utilizado na integração do sistema, com suas várias versões, de acordo com as necessidades de projeto, variando desde 16 bits a até 32 bits com banco de registradores porta dupla (permite dois acessos simultâneos). A versão 16 bits foi feita na tentativa de implementar um sistema em um dos FPGAs disponíveis (Flex 10K20). As versões de 32 bits, sendo uma com banco de registradores porta dupla e a outra com banco de registradores porta simples, foram implementadas seguindo a

estrutura apresentada na especificação original do Risco realizada por [JUN93], sem o uso de pipeline. A versão com o banco de registradores porta simples decorre da tentativa de colocar o processador em um FPGA como o citado acima.

Foram apresentadas também as implementações realizadas do sistema gerador de sinais analógicos, com suas respectivas simulações, tanto no ambiente MaxPlusII da Altera quanto em Matlab, para validar a integração do circuito digital com o filtro analógico. Também foi mostrado a integração do sistema gerador de sinais com o respectivo Core, transformando alguns registradores de uso geral do Core em registradores específicos para a programação do sistema gerador de sinais. Nas simulações, em especial a simulação em Matlab pois nessa considera-se também os componentes analógicos, pode-se perceber que o sistema realmente é capaz de gerar os sinais desejados.

O microprocessador Risco foi escolhido por ser um microprocessador desenvolvido anteriormente na própria UFRGS, existindo uma especificação completa e uma descrição em HDC, e também, por já existir inclusive compiladores C e Assembler para esse processador. Outro motivo que levou à escolha dele foi o fato dele ser um microprocessador RISC muito simples, e de não haver ainda uma implementação deste em linguagem HDL, tipo Verilog ou VHDL.

No desenvolvimento desse trabalho foi utilizada uma metodologia para verificar o correto funcionamento das partes. Inicialmente o microprocessador foi implementado em VHDL e simulado para verificar seu correto funcionamento. Após isso o gerador de sinais também foi implementado em VHDL, apenas a parte digital, e feita sua simulação. Como a simulação só era capaz de cobrir a parte digital, esse circuito foi também implementado utilizando-se a ferramenta MATLAB, através da qual foi possível realizar não apenas a simulação digital como também a simulação dos componentes analógicos. Uma vez que os componentes haviam sido validados, foi gerado seu layout através de uma biblioteca Standard-cell na ferramenta Mentor Graphics, possibilitando a sua fabricação.

Com relação aos circuitos criados, com seus respectivos leiautes, as próximas etapas são a simulação elétrica dos leiautes e a implementação em silício das partes do projeto, Risco e gerador de sinais, além da criação de uma placa para acomodar o sistema e colocá-lo em funcionamento, e testá-lo num ambiente real. O uso desses COREs em outros sistemas também é uma realidade muito próxima. Pode-se usar o Risco para outras finalidades, inclusive fazendo-se versões menores (16 ou 8 bits, que não foram cobertas na especificação original), e também poderia usar um outro processador junto com o gerador de sinais, como por exemplo o 8051 ou o MCORE.

Anexo 1 Descrição Matlab do Gerador de Sinais

Código Matlab implementado para realizar testes do sistema proposto.

Código de teste do sistema que chama outros códigos, foi nomeado de geral.m

```
close all;
clear all;
sf1=sistema(1000,.9);
figure;
plot(sf1);
title('saida apos o RC');
zoom;
```

Código do sistema, modela o sistema como ele é. Nome usado foi sistema.m, que é o nome da função implementada.

```
%frequencia em hertz, amplitude maxima=1
function sf1=sistema(frequencia,amplitude);
%close all;
%clear all;
%frequencia=244;
%amplitude=0.9;
simulationlength=15000;
%algumas condicoes iniciais
fos=25175000/16;
dados=0;
reg1ant=0;  %sem deslocamento de fase
%a12=-5;
a21=sqrt((2-2*cos(2*pi*frequencia/fos)));
a21=round(log2(a21));
reg2ant=floor((amplitude*sin((2*pi*frequencia)/fos)/2^a21)*(2^32))/(2^
32);
reg2ant=floor(reg2ant*(2^16));
reg2ant
a21
for contador = 1:simulationlength,
   reg1shift=bitshift(reg1ant,a21);
   reg2=reg2ant-reg1shift;
%  reg2=floor((reg2ant-reg1ant*2^a21)*(2^32))/(2^32);
   reg2shift=bitshift(reg2,a21);
   reg1=reg1ant+reg2shift;
%  reg1=floor((reg1ant+reg2*2^a21)*(2^32))/(2^32);
   reg2ant=reg2;
   reg1ant=reg1;
   x(contador)=reg1;
end
figure;plot(x);title('saida do oscilador');zoom;
figure;plot(fftshift(abs(fft(x))));title('fft do oscilador');zoom;
%for cont=1:simulationlength,
%   for i=1:4,
%      dados=[dados x(cont)];
%   end
% end
dados=x/(2^16);
%saida=sdelta(1,dados);
saida=sdelta(2,dados);
```

```
%figure; plot(saida); title('saida do sigma delta');zoom;
%[B,A]=butter(6,0.08);saidaf1=filter(B,A,saida);
%figure;freqz(B,A);
%figure;plot(saidaf1);title('Saida do Sigma-delta apos o filtro passa
baixo de 6a.ordem');
%figure;plot(fftshift(abs(fft(saidaf1))));title('Fft do sinal
filtrado');
%Definicao independente do filtro de laco
R= 100000;
C= 1e-6;
NS = [0 1/(R*C)];
DS = [1 1/(R*C)];
stran = tf(NS, DS);
ztran = c2d(stran, 1/simulationlength*2, 'tustin');
%[NZ, DZ] = tfdata(ztran);
NZ = [0.002494 0.002494]
DZ = [1       -0.995   ]
saida2=0;
for i=1:length(saida),
   for j=1:10,
      saida2=[saida2 saida(i)];
   end
end
figure;plot(saida2);title('saida antes do RC');zoom;
figure;plot(fftshift(abs(fft(saida2))));title('fft da saida do sigma
delta');zoom;
sf1= filter(NZ, DZ, saida2);
%figure;plot(sf1);title('saida apos o RC');zoom;
%figure;plot(fftshift(abs(fft(sf1))));title('fft do sinal apos o
filtro');
```

Código do circuito sigma-delta, também como função, a qual é chamada pela função sistema.

```
% Modulador sigma delta
function [y]=sdelta(order,dados);
%tamanho da simulacao equivale a quantidade de dados
simulationlength=length(dados);
%algumas condicoes iniciais
a=[1 1];
b=[0 4];
for k=1:order,
   v (k)= 0;
end
%comeca a execucao do algoritmo Sigma-delta
for time= 1:simulationlength,
   x= dados(time);
   if v(order) >= 0
       y(time)=1;
    else
       y(time)=-1;
    end
    for k=order:-1:2 % order to 2 step -1
        v(k) = v(k) + a(k)* (v(k-1)-b(k)*y(time));
    end
    v(1)=v(1) +a(1)*(x-y(time));
end
```

Anexo 2 Leiaute dos Componentes

Layout do Microprocessador Risco e do gerador de sinais analógicos e suas partes.

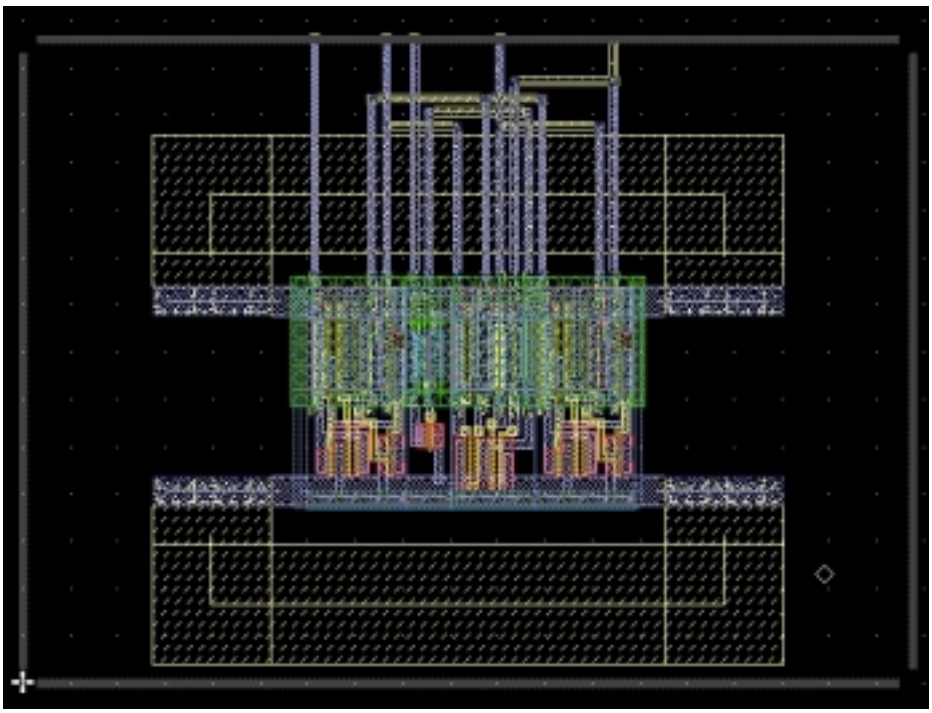


FIGURA B.1 – Layout de um somador de 1 bit

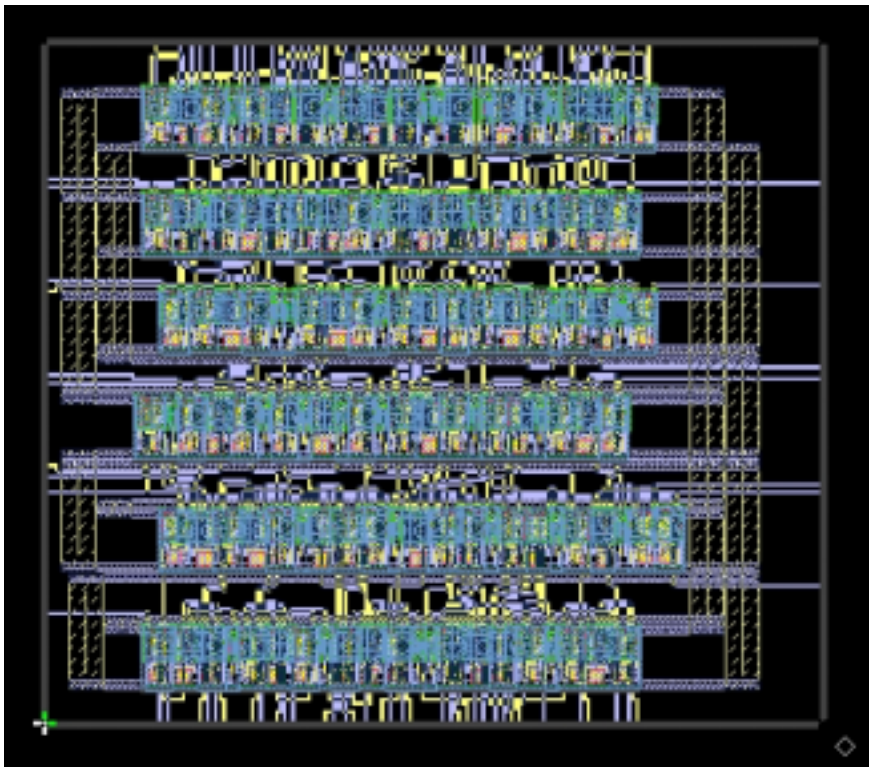


FIGURA B.2 – Layout de um somador de 32 bits

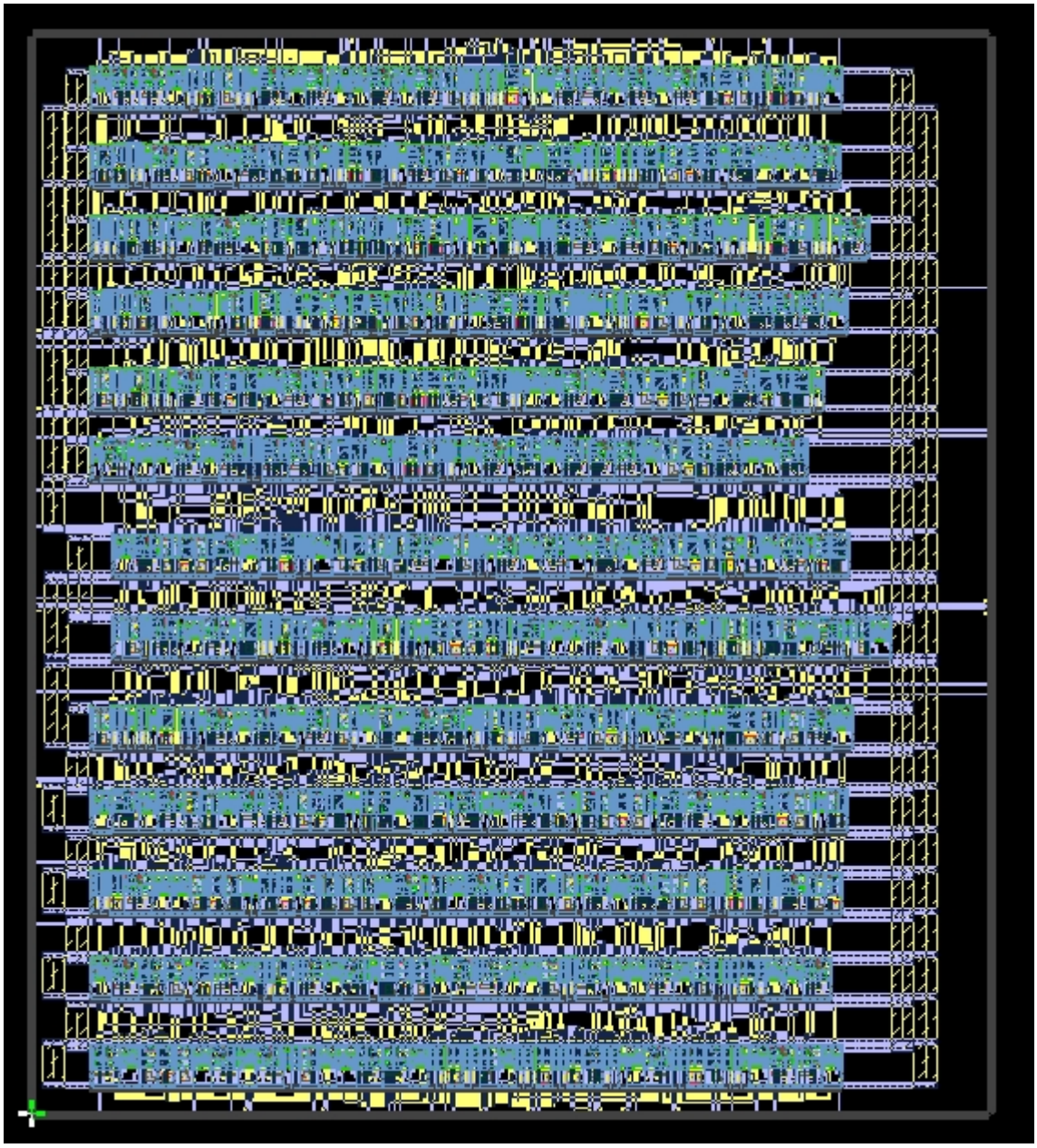


FIGURA B.3 – Layout da Unidade Lógica-Aritmética do Risco

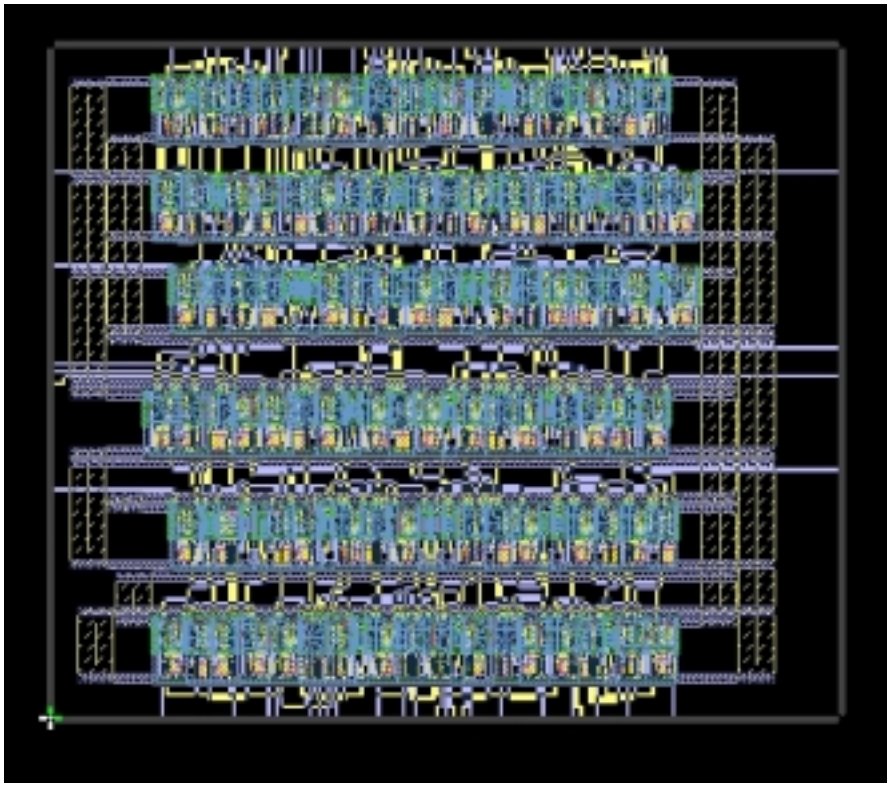


FIGURA B.4 – Layout da Unidade de Contador de Programa (Risco)

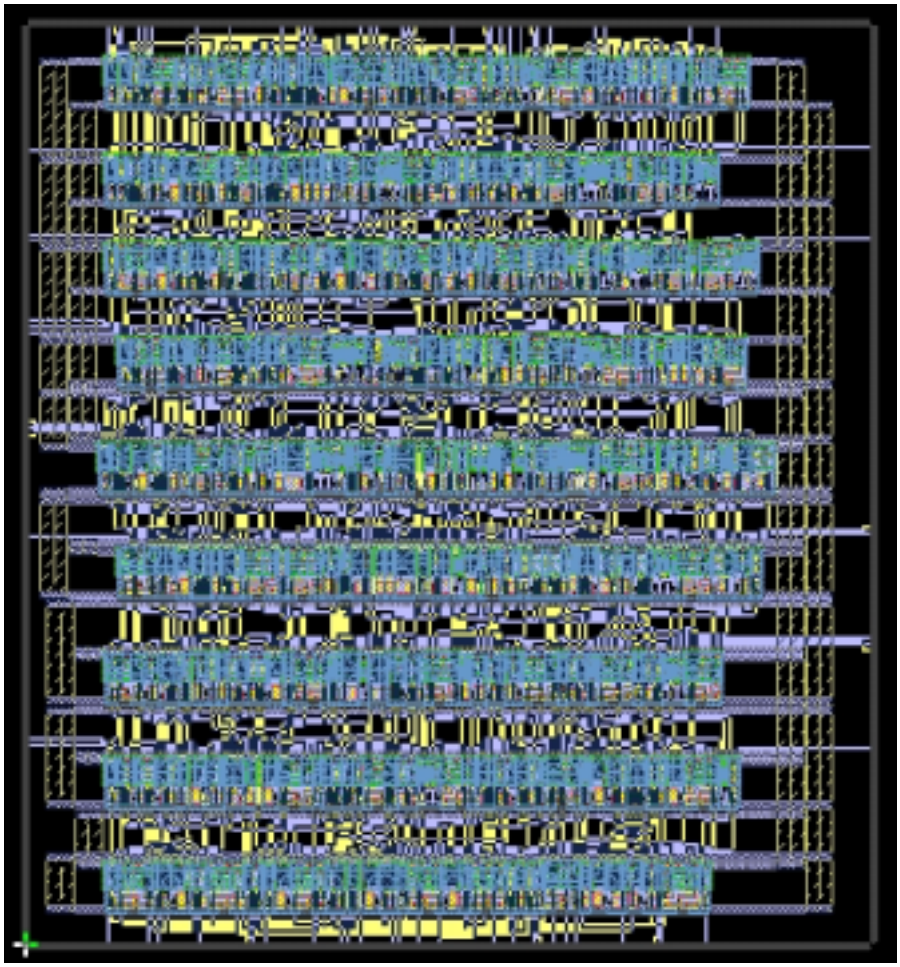


FIGURA B.5 – Layout da Unidade de Deslocamento do Risco

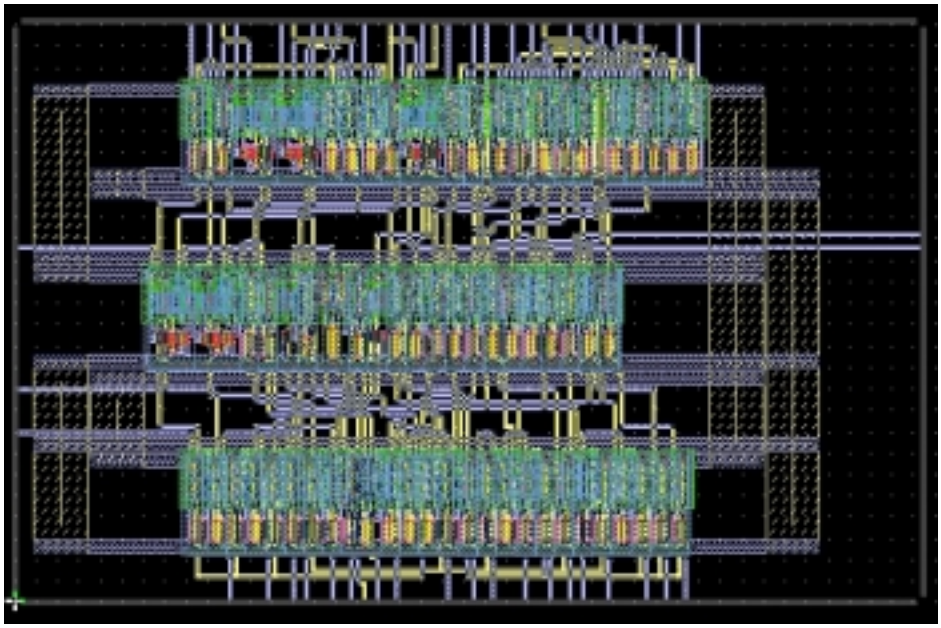


FIGURA B.6 – Layout da Unidade de Constante do Risco

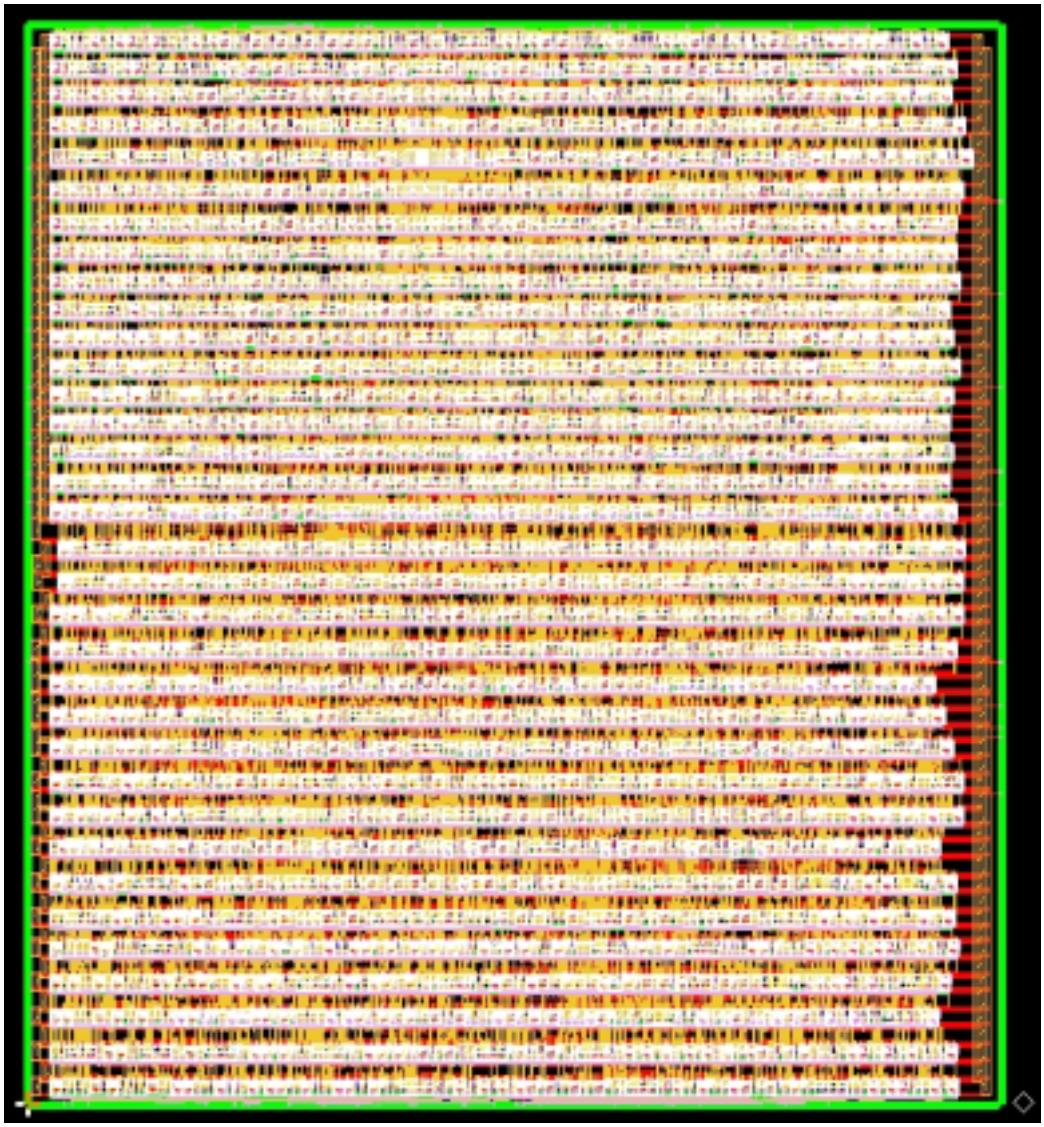


FIGURA B.7 – Layout do Banco de Registradores do Risco

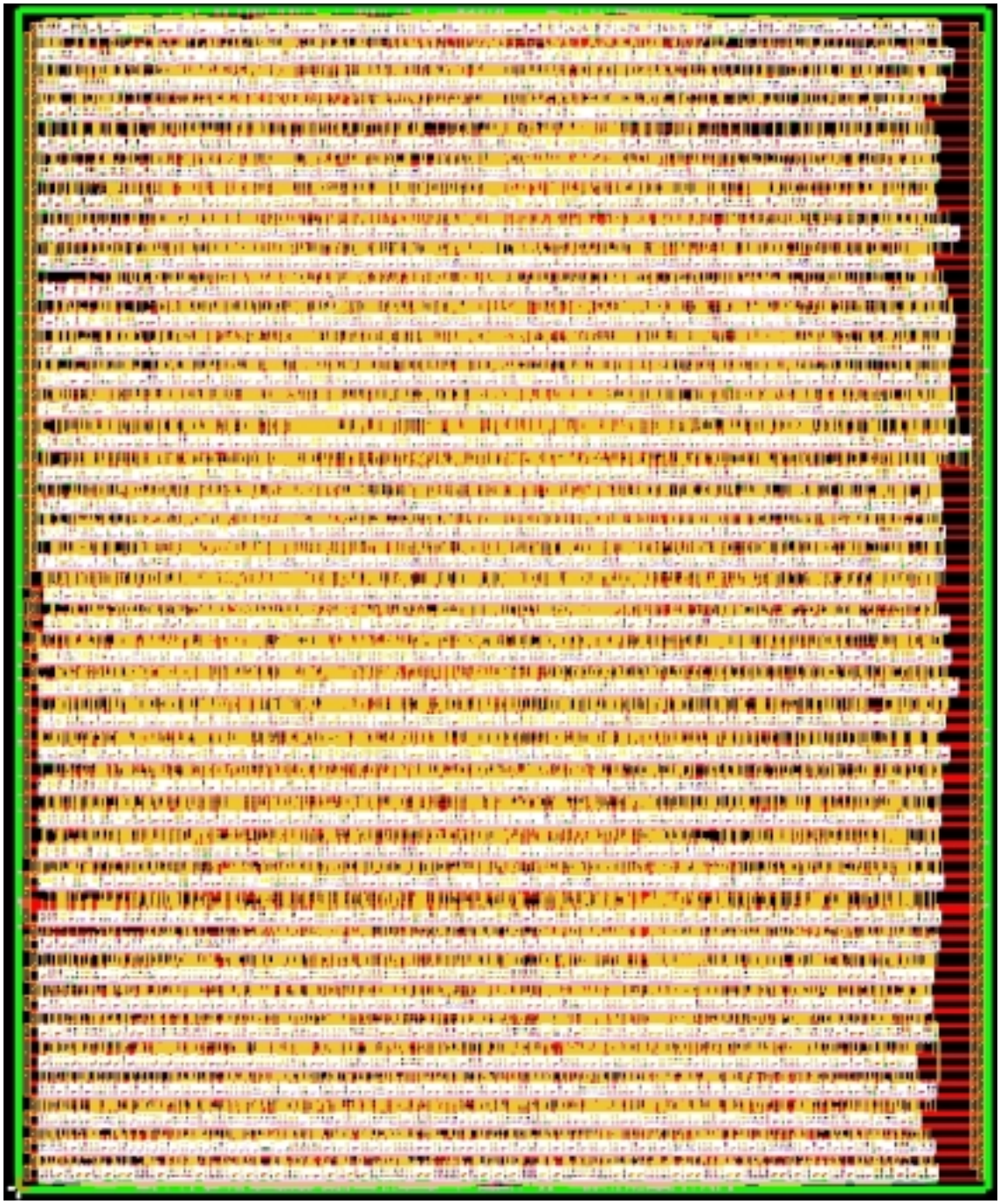


FIGURA B.8 – Layout da Parte Operativa do Risco

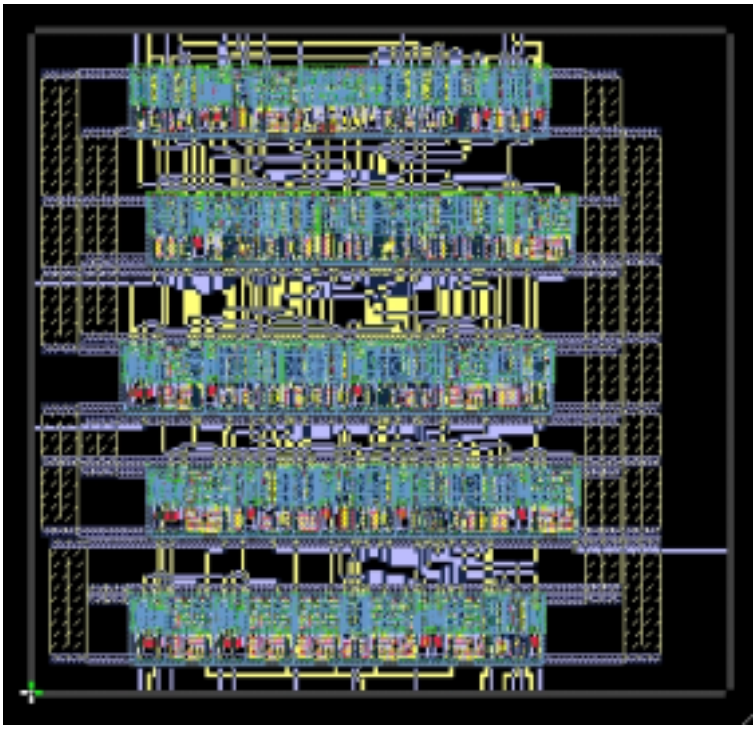


FIGURA B.9 – Layout da Parte de Controle do Risco

FIGURA B.10 – Layout final do Risco (Parte Operativa e Parte de Controle)

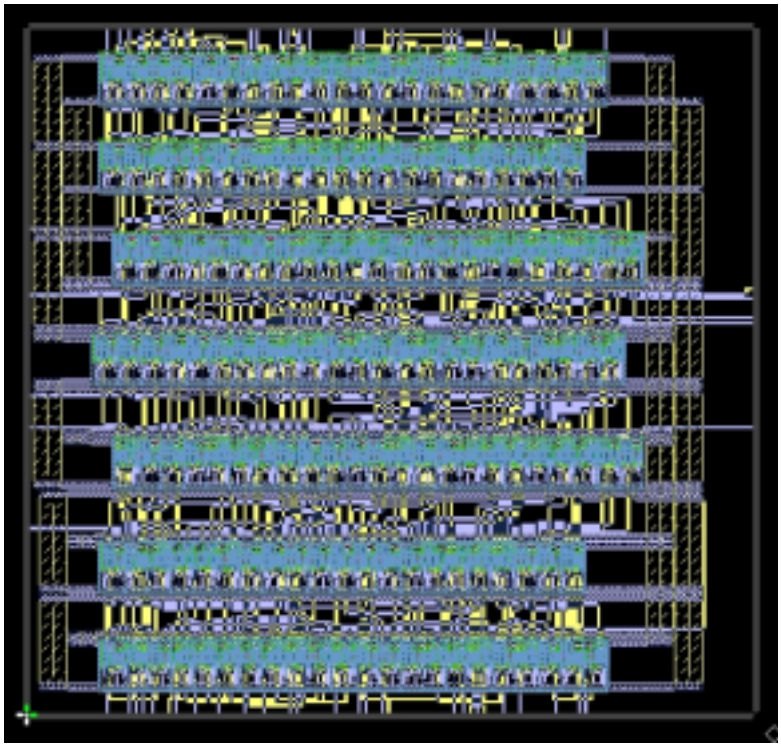


FIGURA B.11 – Layout do Shiftador utilizado no gerador de senóides

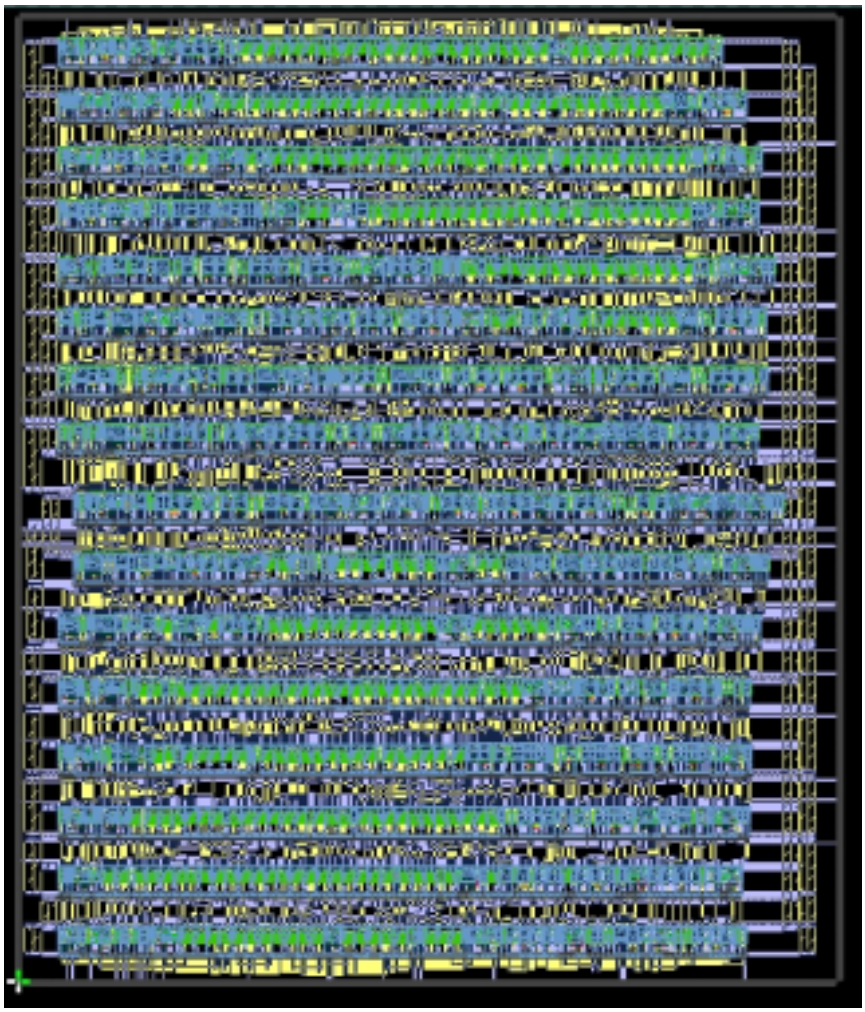


FIGURA B.12 – Layout do Gerador de senóides

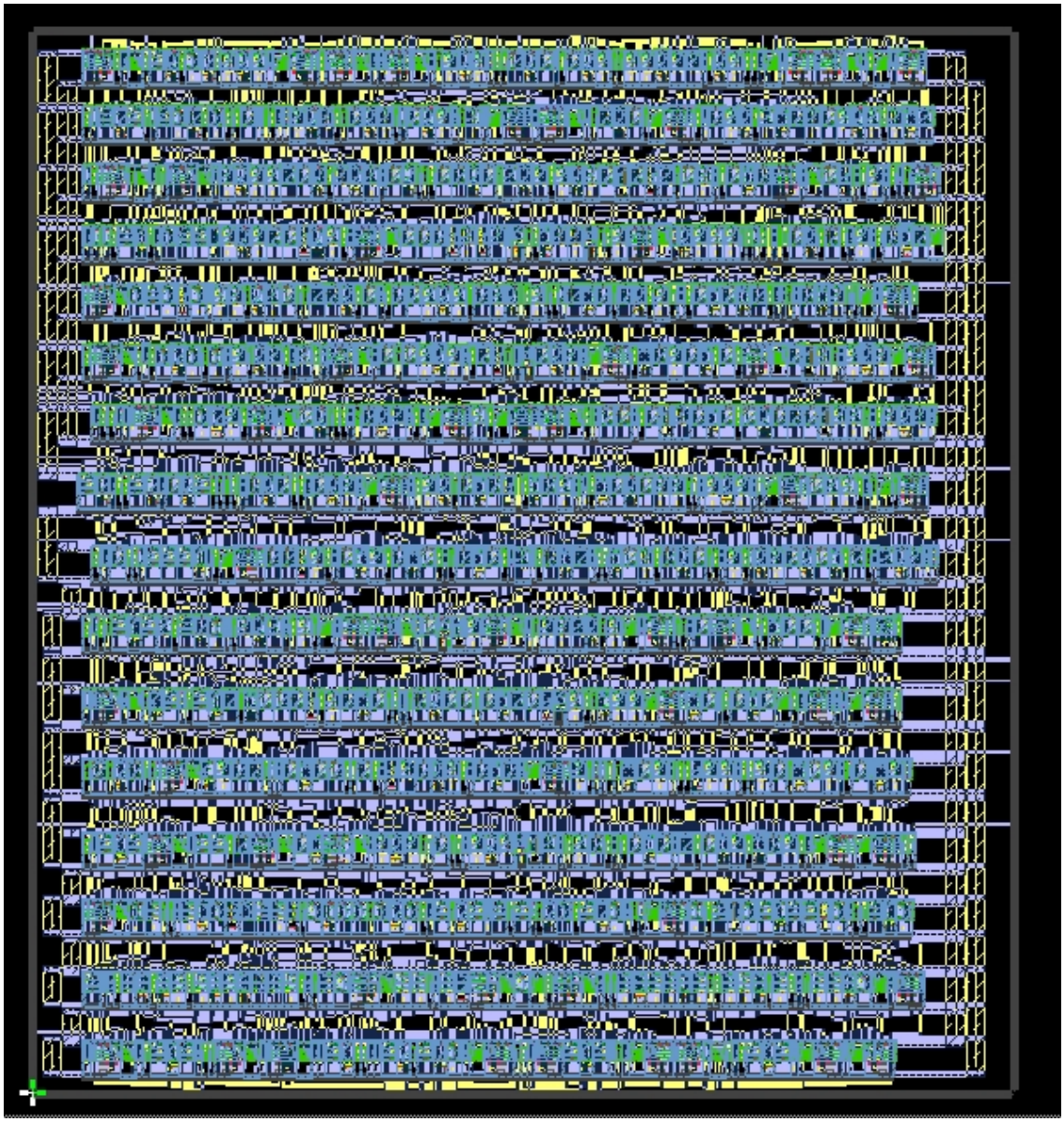


FIGURA B.13 – Layout do Modulador sigma-delta

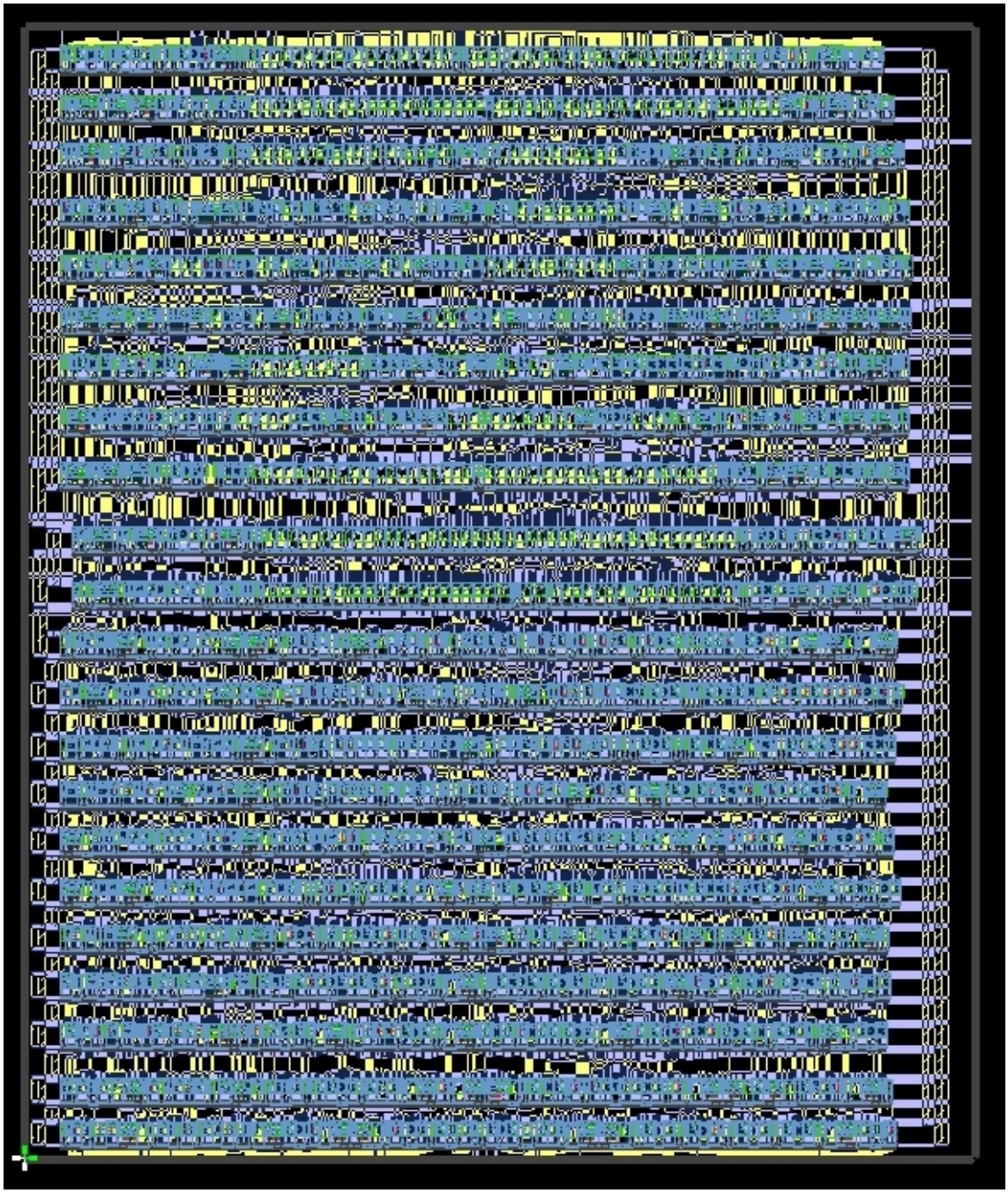


FIGURA B.14 – Layout do sistema gerador de sinais analógicos

FIGURA B.15 – Layout do circuito final (Risco e Gerador de sinais analógicos)

Bibliografia

[CAN92] CANDY, J.C.; TEMES, G.C. Oversampling methods for A/D and D/A conversion. In: **Oversampling Delta-Sigma Data Converters.** New York: IEEE Press, 1992.

[COT97] COTA, E.F.; Di DOMENICO, E.J.; LUBASZEWSKI, M. ATPG para teste de Circuitos Analógicos e Mistos. In: WORKSHOP IBERCHIP, 3., 1997, Mexico. [**Anais**...] Mexico: Departamento de Ingineria Electrica, 1997.

[EIM70] EIMBINDER, J. **Application Considerations for Linear Integrated Circuits**. New York, NY: Mactier Publishing Company, 1970.

[FRE78] FRERKING, M. E. **Crystal Oscillator Design and Temperature Compensation**. New York, NY: Van Nostrand Reinhold Company, 1978.

[GRE86] GREGORIAN, R.; TEMES, G. C. **Analog MOS Integrated Circuits for Signal Processing.** New York, NY: John Wiley & Sons Inc., 1986.

[HAU98] HAURIE, X.; ROBERTS, G. W. Arbitrary-Precision Signal Generation for Mixed-Signal Built-In-Self-Test. **IEEE Transactions on Circuits and Systems,** New York, v. 45, n. 11, Nov. 1998.

[HNA88] HNATEK, E.R. **A Users Handbook of D/A and A/D Converters**. Malabar, FL: Robert E. Krieger Publishing Company, 1988.

[JOH73] JOHNSON, K.R. Resistive bridge and Op Amp Form a Multiplying DAC. **Electronic Design**, New York, n. 19, Sept. 1973.

[JUN93] JUNQUEIRA A.D. **Risco**: Micro-processador RISC CMOS de 32 Bits.1993. 256f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[KIM88] KIM, K.; TRONT, J. G.; HA, D. S. Automatic Insertion of BIST Hardware Using VHDL. In: IEEE/ACM DESIGN AUTOMATION CONFERENCE, DAC, 25., 1988. **Proceedings...** New York: ACM, 1988.

[LES90] LESLIE, T. C.; SINGH, B. An Improved sigma-delta modulator architecture. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 1990. **Proceedings...** New York: IEEE**,** 1990.

[LUB96] LUBASZEWSKI, M.; MIR, S.; PULZ, L. ABILBO: Analog Built-In Block Observer. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1996. **Digest of technical papers.** Los Alamitos: IEEE Computer Society Press, 1996.

[McC85] McCLUSKEY, E. J. Built-In Self-Test Techniques. **IEEE Design and Test of Computers,** New York, v. 2, n. 2, April 1985.

[NOR96] NORSWORTHY, S. R. A minimal multi-bit digital noise shaping architecture. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 1996. **Proceedings...** New York: IEEE, 1996.

[NOR96a] NORSWORTHY, S. R.; TEMES, G.C.; SHREIER, R. **Delta-sigma data converters:** Theory, design and simulation. New York, NY: IEEE Press, 1996.

[PAP99] PAPACHRISTOU, C. A.; MARTIN, F.; NOURANI, M. Microprocessor Based Testing for Core-Based System on Chip. In: IEEE/ACM DESIGN AUTOMATION CONFERENCE, DAC, 36., 1999. **Proceedings...** New York: ACM, 1999.

[PAW96] PAWELSKI, P.; LUBASZEWSKI, M. S. Analog BIST and Boundary Scan in Mixed Signal Boards. In: UFRGS MICROELECTRONICS SEMINAR, 11., 1996. **Proceedings...** Porto Alegre: CPGCC da UFRGS, 1996.

[REN97] RENOVELL, M. et al. A Multi-Mode Signature Analyzer for Analog and Mixed Circuits. In: IFIP TC10WG 10.5 INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 9., 1997. **VLSI**: Integrated systems on silicon. London: Chapman & Hall, 1997.

[ROB95] ROBERTS, G. W.; LU, A. K. **Analog Signal Generation for Built-In-Self-Test of Mixed-Signal Integrated Circuits**. London: Kluwer Academic Publishers, 1995.

[ROB96] ROBERTS, G. W. Metrics, Techniques and Recent Developments in Mixed-Signal Testing. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1996. **Digest of technical papers.** Los Alamitos: IEEE Computer Society Press, 1996.

[TON95] TONER, M. F.; ROBERTS, G. W. On the Pratical Implementation of Mixed Analog-Digital BIST. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE,CICC, 17., 1995. **Proceedings...** Santa Clara: IEEE, 1995.

[UCH88] UCHIMURA, K. et al.; Oversampling A-to-D and D-to-A converters with multistage noise shaping modulatoors. **IEEE Transactions on Acoustic Speech Signal Processing**, New York, v. AASP-36, Dec. 1988.

[WES93] WESTE, N. H. E.; ESHRAGHIAN, K. **Principles of CMOS VLSI Design: A System Perspective.** New York, NY: Addison-Wesley Publishing Company, 1993.

[ZIM2000] ZIMMERMANN, F. L. O.; SUSIN, A. A.; RENOVELL, M. A Microprocessor with Analog Capabilities. In: IEEE INTERNATIONAL MIXED-SIGNAL TESTING WORKSHOP, IMSTW, 6., 2000. **Proceedings...** Montpellier: IEEE, 2000.

[ZIM2000a] ZIMMERMANN, F. L. O.; SUSIN, A. A. A Microprocessor with analog capabilities. In: UFRGS MICROELECTRONICS SEMINAR, 15., 2000. **Proceedings...** Canoas: Gráfica da ULBRA, 2000.

[ZIM2001] ZIMMERMANN, F. L. O.; SUSIN, A. A. A Digital Self-Testable Wave Generator Core for Analog Signals. In: WORKSHOP IBERCHIP, 7., 2001, Uruguai. **[Anais...]** Uruguai: Departamento de Ingineria Electrica, 2001.