UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SISTEMAS DE COMPUTAÇÃO E TELECOMUNICAÇÕES

PROJETO DE UM CONVERSOR ANALÓGICO/DIGITAL POR APROXIMAÇÕES SUCESSIVAS DE 12 *BITS*

RODRIGO DURÃES DE VASCONCELLOS

BELO HORIZONTE
DEZEMBRO/2011

RODRIGO DURÃES DE VASCONCELLOS

PROJETO DE UM CONVERSOR ANALÓGICO/DIGITAL POR APROXIMAÇÕES SUCESSIVAS DE 12 BITS

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

Área de Concentração: Sistemas de Computação e Telecomunicações.

Linha de Pesquisa: Microeletrônica e Microssistemas (MeMSs).

Orientador: Prof. Dr. Diógenes C. da Silva Júnior

BELO HORIZONTE
DEZEMBRO/2011

Dedico esse trabalho a Tatiana e Ana Laura.

AGRADECIMENTOS

A Tatiana pelo companheirismo, compreensão e paciência. Ao pessoal do LabSCI: Adriano, Alair, Átila, Fernando e Renato. Ao pessoal da DHBH: Érlon, Silvério e João.

Um agradecimento especial ao Prof. Diógenes pela oportunidade, pelo suporte e sobretudo, pelo esmero e paciência na orientação dos trabalhos.

Thinking is the hardest work there is,
which is probably the reason why so
few engage in it.

Henry Ford

RESUMO

Este trabalho apresenta o projeto de um conversor analógico/digital por aproximações sucessivas (SAR ADC) de 12 *bits*. As etapas de desenvolvimento do circuito de sinais mistos seguem uma metodologia específica que começa com a criação de modelos em alto nível de abstração e finaliza com a implementação de um circuito integrado (CI). As etapas são realizadas com o auxílio de ferramentas como o Stateflow, Simscape, SystemC/SystemC-AMS e Cadence Design Framework II. O desenvolvimento do conversor surge da crescente demanda por subsistemas de sinais mistos integrados em um SoC (*System on Chip*).

Palavras-chave: conversor analógico/digital, aproximações sucessivas, SystemC, SystemC-AMS, Stateflow, Simscape, circuito integrado.

ABSTRACT

This work presents the project of a 12-bit successive approximation analog-to-digital converter (SAR ADC). The mixed-signal development stages use a specific methodology that starts with the design of high abstraction level models and ends with the design of an integrated circuit (IC). The stages are fulfilled with the help of tools such as Stateflow, Simscape, SystemC/SystemC-AMS and Cadence Design Framework II. The converter development comes from the increasing demand for mixed-signal subsystems integrated into a SoC (System on Chip).

***Keywords**: analog-to-digital converter, successive appoximation, SystemC, SystemC-AMS, Stateflow, Simscape, integrated circuit.*

LISTA DE FIGURAS E LISTAGENS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog-to-digital converter*
AMS	*Analog and mixed-signal*
BCD	*Binary-coded decimal*
CI	Circuito integrado
CM	*Common-mode*
DAC	*Digital-to-analog converter*
DNL	*Differential nonlinearity*
DRC	*Design rule checking*
DSP	*Digital signal processing*
DUT	*Device under test*
ELN	*Electrical linear networks*
ENIAC	*Electronic Numerical Integrator And Computer*
ENOB	*Effective number of bits*
EOC	*End-of-convert*
ERB	Estação Rádio Base
FFT	*Fast Fourier transform*
FPB	Filtro passa-baixa
FS	*Full-scale*
FSM	*Finite-state machine*
INL	*Integral nonlinearity*
IP	*Intellectual property*
LSB	*Least significant bit*
LSF	*Linear signal flow*
LVS	*Layout versus schematic*
MOSFET	*Metal–oxide–semiconductor field-effect transistor*
MSB	*Most significant bit*
NPR	*Noise power ratio*
OSCI	Open SystemC Initiative
PCM	*Pulse-code modulation*
ppm	*Parts per million*
RMS	*Root mean square*
RTL	*Register-transfer level*
S/H	*Sample-and-hold*
SAR	*Successive approximation register*
SFDR	*Spurious free dynamic range*
SHA	*Sample-and-hold amplifier*
SINAD	*Signal-to-noise-and-distortion ratio*
SNR	*Signal-to-noise ratio*
SPS	*Samples per second*
T/H	*Track-and-hold*
TDF	*Timed data flow*
THA	*Track-and-hold amplifier*
THD	*Total harmonic distortion*
TG	*Transmission gate*
VCD	*Value change dump*
VLSI	*Very-large-scale integration*

SUMÁRIO

1 INTRODUÇÃO

Os conversores de dados, circuitos que convertem sinais analógicos em representações digitais ou vice-versa, desempenham um importante papel em um mundo digital crescente. À medida que os produtos eletrônicos lançados realizam um número cada vez maior de operações no domínio digital, os conversores de dados devem prover a passagem dos dados digitais para um mundo inerentemente analógico, bem como o caminho inverso.

O conversor analógico/digital, também denominado conversor A/D ou ADC, é considerado um dispositivo codificador na medida em que converte uma amostra analógica em uma quantidade digital com um determinado número de *bits*. Numerosos tipos de conversores estão disponíveis para infindáveis aplicações. O tipo de aplicação geralmente determina a escolha da técnica de conversão a ser empregada (Gregorian, 1999). O conversor digital/analógico, também denominado conversor D/A ou DAC, realiza a operação inversa do ADC.

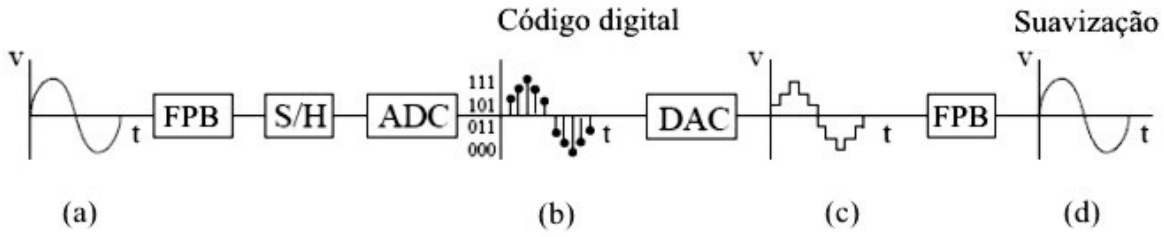

Figura 1.1: características do sinal produzido por uma conversão A/D e D/A.

Baker (Baker, 2008) ilustra o processo de conversão A/D e D/A, conforme visto na figura 1.1, no qual o sinal analógico original (a) é aplicado em um filtro passa-baixa (FPB) para remover quaisquer componentes de alta frequência que possam causar o efeito conhecido como falseamento ou *aliasing*. O sinal é amostrado e retido por um dispositivo de amostragem e retenção (*sample-and-hold* - S/H) e então é convertido (pelo ADC) em um sinal digital (b). Em

seguida, o DAC converte o sinal digital em um sinal analógico (c). Em (d) um filtro passa-baixa (FPB) promove o retorno à conformação "original" do sinal analógico (observando-se eventuais mudanças de fase introduzidas nas conversões). O sinal analógico em (a) contém valores contínuos e infinitos ao passo que o sinal digital em (b) é discreto em relação ao tempo e quantizado, ou seja, discretizado em um dentre um número finito de valores possíveis (em oposição ao valores infinitos do sinal analógico).

De acordo com Oppenheim (Oppenheim *et al*, 2010), os "sinais de tempo contínuo" (*continuous-time signal*) são definidos ao longo de um tempo ininterrupto e são, deste modo, representados por uma variável independente contínua e os "sinais de tempo discreto" (*discrete-time signal*) são definidos em tempos discretos e, desta forma, a variável independente possui valores discretos. Hayes (Hayes, 1999) acrescenta que um sinal de tempo discreto é uma sequência indexada de números reais ou complexos. Assim, um sinal de tempo discreto é uma função de uma variável de valor inteiro *n* denotada, por exemplo, por *x(n)*.

Um sinal de tempo discreto é indefinido para valores não inteiros de *n*. Sinais digitais são aqueles para os quais tanto o tempo quanto a amplitude são discretos (Oppenheim *et al*, 2010).

Do ponto de vista da implementação, o ADC tipicamente contém um ou mais comparadores, chaves, componentes passivos, uma fonte de tensão de referência e uma lógica de controle digital (Gregorian, 1999).

1.1 MOTIVAÇÃO

É sabido que os mais diferenciados tipos de ADC são vastamente encontrados no mercado, fornecidos pela AD (Analog Devices, Inc.), TI (Texas Instruments, Inc.), Linear (Linear Technology Corporation), Maxim (Maxim Integrated Products), etc. No entanto, esses "produtos de prateleira" estão envoltos em tecnologias proprietárias e inúmeras patentes, além

do próprio encapsulamento, que impedem que sejam incluídos em sistemas customizados como em um SoC (*System on Chip*).

1.2 OBJETIVOS

Devido à impossibilidade de se integrar em um CI customizado (*full-custom*) os conversores disponíveis no mercado pelos motivos já citados, surgiu a possibilidade de se desenvolver um ADC que pudesse ser posteriormente incorporado ao projeto de um SoC.

Objetivando a redução no tempo de desenvolvimento, foi aplicada uma metodologia que envolve descrições em alto nível de abstração com a utilização de linguagens de descrição de sistemas eletrônicos (HDL) e de ferramentas como o SystemC-AMS e Matlab. Assim, partindo-se de uma ideia inicial geral, os elementos constituintes do conversor poderiam ser desenvolvidos e posteriormente agrupados para a obtenção de sua conformação final.

As características principais definidas para o conversor são: arquitetura SAR, resolução de 12 *bits*, taxa de amostragem de 100 kSPS, frequência do *clock* de alimentação de 2 MHz, relação sinal-ruído igual a 74 dB, tensão de entrada variando de 0 a V_{DD} (tensão de alimentação), tensão de alimentação de 3,3 V, erro de linearidade diferencial não inferior a – 0,9 LSB e não superior a +1 LSB e erro de linearidade integral entre ± 1 LSB. A arquitetura do ADC por aproximações sucessivas foi escolhida por aliar alta resolução, alta velocidade e utilização de área relativamente pequena.

Não houve, essencialmente, a busca pelo desenvolvimento do "estado da arte" em conversores A/D por aproximações sucessivas. A primeira versão do conversor serviria para agregar o conhecimento necessário de projeto com as respectivas dificuldades e implicações, servindo de base para versões futuras. Além disso, seria verificada a aplicabilidade e eficácia da metodologia empregada quanto à redução no tempo de desenvolvimento, bem como no seu emprego em outros projetos.

1.3 ESTRUTURA DA DISSERTAÇÃO

A presente dissertação, está organizada como segue:

- Capítulos 2, 3 e 4: revisão bibliográfica, elemento chave para o embasamento teórico, definição da terminologia empregada e subsídio para escolha dos componentes mais adequados ao sistema;
- Capítulo 5: metodologia empregada, definindo as etapas e procedimentos do processo de desenvolvimento;
- Capítulo 6: criação dos modelos, dos esquemáticos e do *layout* correspondente ao circuito integrado;
- Capítulo 7: exposição dos resultados obtidos com a aplicação da metodologia proposta; e
- Capítulo 8: considerações finais, desafios e dificuldades encontradas, bem como sugestões para trabalhos futuros.

2 FUNDAMENTOS DA CONVERSÃO DE DADOS

2.1 BREVE HISTÓRICO SOBRE OS CONVERSORES DE DADOS

Conforme Kester (Kester, 2005), até meados da década de 50 os conversores de dados eram primariamente desenvolvidos e utilizados em aplicações bastante especializadas como nas pesquisas do Bell System no PCM, bem como em sistemas de criptografia de mensagens na Segunda Guerra Mundial. A tecnologia de válvulas tornava os conversores caros, volumosos e de alto consumo. Não havia praticamente uso comercial para tais dispositivos.

O computador digital se constituiu em um fator importante por trás do desenvolvimento do ADC comercial. O projeto de desenvolvimento do ENIAC, primeiro computador eletrônico de propósito geral, se iniciou em 1942 e foi revelado ao público em fevereiro de 1946. Foi desenvolvido para o cálculo de trajetórias balísticas para o laboratório de pesquisas em balística do exército norte-americano (*United States Army Ballistic Reseach Laboratory*). O ENIAC levou ao desenvolvimento do primeiro computador digital comercialmente disponível, o UNIVAC, por Eckert e Mauchly.

Aplicações militares representaram inicialmente elementos motivadores para a criação do computador digital. No entanto, com o passar do tempo surgiu a possibilidade de utilização em áreas como a análise de dados e no controle de processos industriais, criando-se assim um maior interesse no processamento digital e, outrossim, na necessidade de conversores de dados. Em 1953, Bernard M. Gordon, um pioneiro no campo da conversão de dados e que havia trabalhado no computador UNIVAC, fundou a EPSCO Engineering que, em 1954, introduziu um ADC de 11 *bits*, 50 kSPS, baseado na tecnologia de válvulas. Esse conversor é creditado como sendo o primeiro de uso comercial. O conversor EPSCO "DATRAC" dissipava 500 W, pesava aproximadamente 68 kg e apresentava medidas generosas (48 x 38 x 66 cm). Como o DATRAC possuía uma função de amostragem e retenção (SHA), esse foi o primeiro ADC comercial adequado para digitalização de sinais em corrente alternada. A figura 2.1 mostra o

"DATRAC" da EPSCO.

No início da década de 60 o desenvolvimento dos circuitos eletrônicos começou a migrar das válvulas para os transistores abrindo, desta forma, novas possibilidades aos projetos de conversores de dados.

Figura 2.1: ADC "DATRAC" valvulado de 11 *bits*, 50 kSPS.

Na década 70 houve uma grande entrada de companhias como a National Semiconductor, Analog Devices, Computer Labs, dentre outras, no campo dos conversores de dados. O mercado de ADC/DAC foi impulsionado por inúmeras aplicações, incluindo-se voltímetros digitais de alta resolução, controle de processos industriais, vídeo digital, sistemas militares avançados de radar *(phased array radio detecting and ranging),* imageamento aplicado à medicina, etc.

Apesar de a maioria dos ADCs do começo da década de 70 serem modulares ou híbridos, houve um considerável esforço por parte dos fabricantes em se produzir um ADC monolítico. No entanto, devido a dificuldades no projeto de bons comparadores, bem como amplificadores nos, até então, recentes processos CMOS (c*omplementary metal–oxide semiconductor*), verificava-se a necessidade da utilização de comparadores e de tensões de

referência externos. Há que se notar que nenhum dos ADCs monolíticos ou híbridos possuíam dispositivos SHA (*sample-and-hold amplifier*) internos. Havia portanto, a necessidade da conexão do SHA ao ADC com uma interface e circuitos de temporização adequados.

Na década de 80, com a crescente disponibilidade de microprocessadores com custo relativamente baixo, memórias de maior velocidade, DSPs e o surgimento de computadores compatíveis com o IBM PC, houve um aumento no interesse em todas as áreas de processamento de sinais. Especificações como a relação sinal-ruído (SNR), relação sinal-ruído e distorção (SINAD), número efetivo de *bits* (ENOB), relação entre potência e ruído (NPR), dentre outras, começaram a surgir na maioria das folhas de dados dos ADCs.

Kester completa que nessa mesma época houve uma proliferação de conversores de alta velocidade em tecnologias bipolar e CMOS com 4, 6, 8, 9 e 10 *bits* em taxas de amostragem variando de 20 MSPS a 100 MSPS.

Nos anos de 1990 as maiores demandas por conversores de dados se encontrava em aplicações de controle de processos industriais, medições, instrumentação, áudio, vídeo e computação gráfica. Além disso, os sistemas de comunicação apresentavam uma demanda cada vez maior por dispositivos de baixo custo, baixo consumo de energia e conversores de alto desempenho aplicados em modems, telefones celulares, bem como na infraestrutura das estações rádio base (ERB). Nessa época, a tecnologia CMOS se tornou o processo escolhido para os conversores de dados de propósito geral e a tecnologia BiCMOS ficou reservada aos dispositivos mais sofisticados (*high end*) e de custo comparativamente elevado.

As tendências iniciadas na década de 90 continuaram nos anos 2000. O consumo de energia caiu e, juntamente, as tensões de alimentação. Tensões de 5 V, 3,3 V, 2,5 V e 1,8 V seguiram em consonância com os nós de processo CMOS 0,6 μm, 0,35 μm, 0,25 μm e 0,18 μm (Kester, 2005).

2.2 TEOREMA DA AMOSTRAGEM

Harry Nyquist, nos idos da década de 20, estudou a sinalização telegráfica com o objetivo de encontrar a taxa máxima de sinalização que poderia ser utilizada em um canal com uma dada largura de faixa e terminou por estabelecer as bases matemáticas da amostragem (Nyquist, 1924, 1928). O trabalho original de Nyquist foi suplementado posteriormente por R. V. L. Hartley (Hartley, 1928).

De maneira simplificada, o critério de Nyquist estabelece que a frequência de amostragem, f_s, seja maior do que duas vezes o valor da mais alta frequência contida no sinal de entrada (contínuo e com largura de faixa limitada), sob pena de haver perda de informação (Shannon, 1949; Moscovici, 2001). Se a frequência de amostragem for menor do que duas vezes a frequência máxima do sinal analógico a ser convertido, ocorrerá o fenômeno conhecido como falseamento (Oppenheim *et al*, 2010).

Kester (Kester, 2005) descreve que o intervalo discreto de amostragem, $T_s = 1/f_s$, deve ser cuidadosamente escolhido para se assegurar uma melhor representação digital do sinal analógico original. Fica claro que quanto maior o número de amostras (maior taxa de amostragem) mais fiel se mostra a representação digital. No entanto, se uma quantidade menor de amostras for tomada (taxas de amostragem baixas), atinge-se um ponto crítico no qual há perda de informação do sinal. O falseamento ocorre quando $f_s \leq 2f_a$, sendo f_a a máxima frequência do sinal analógico amostrado. Um sinal cujas frequências componentes se situam entre f_a e f_b deve ser amostrado a uma taxa $f_s > 2(f_b - f_a)$ para se prevenir a sobreposição nas frequências do sinal por componentes de falseamento.

Para se entender as implicações do falseamento tanto no tempo quanto na frequência pode se considerar, por exemplo, o caso da representação no domínio do tempo de um sinal senoidal amostrado (figura 2.2) em que f_s não corresponde a pelo menos $2f_a$, mas tão somente a um valor ligeiramente superior à frequência da entrada analógica f_a, violando-se, desta forma, o

critério de Nyquist. Verifica-se portanto, o surgimento de uma onda senoidal "falseada" em uma frequência igual a $f_s - f_a$.

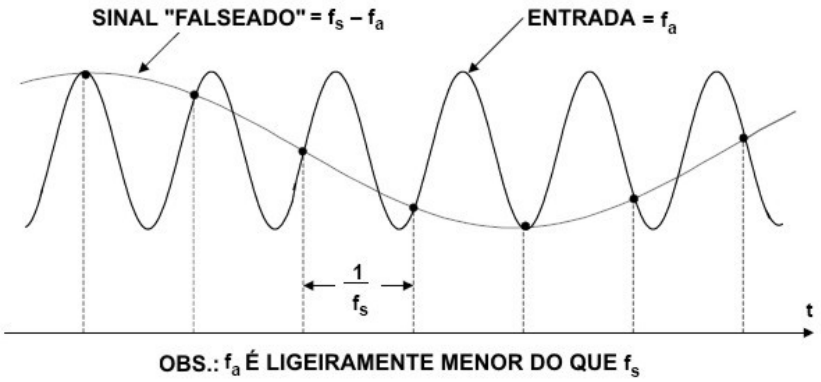

Figura 2.2: falseamento no domínio do tempo.

A representação correspondente no domínio da frequência é mostrada na figura 2.3-b. A saída no domínio da frequência do sinal amostrado mostra "imagens" do sinal original em torno de cada múltiplo de f_s, isto é, em frequências iguais a $|\pm kf_s \pm f_a|$, k = 1, 2, 3, 4, 5, … A figura 2.3-a mostra o caso em que $f_s > 2f_a$, não havendo portanto, imagens sobrepostas.

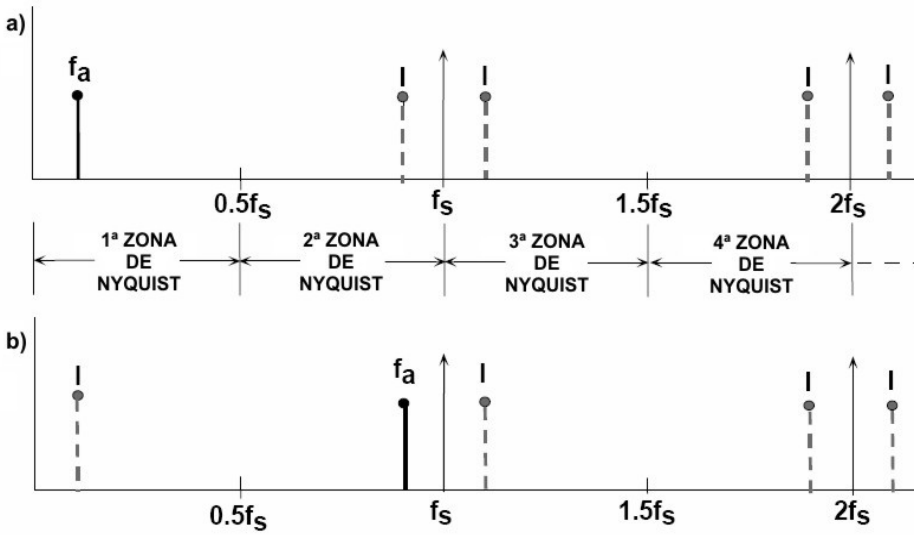

Figura 2.3: sinal analógico f_a amostrado por f_s com imagens (*aliases*) em $|\pm kf_s \pm f_a|$, k = 1, 2, …

A *largura de faixa de Nyquist* é definida como o espectro de frequências que abrange a componente de corrente contínua (CC) até $f_s/2$. O espectro de frequências é dividido em um número infinito de *zonas de Nyquist*, cada uma tendo uma largura igual a $½f_s$, como mostrado na figura 2.3-a. Considerando-se o caso em que o sinal se localiza fora da *primeira zona de Nyquist* (figura 2.3-b), verifica-se que a sua imagem, $f_s - f_a$, localiza-se dentro da mesma ($f_s < 2f_a$). Disso, depreende-se que é necessária algum tipo de filtragem antes do ADC para se remover componentes de frequências que estejam fora da largura de faixa de Nyquist mas, cujos componentes de falseamento se encontrem dentro da mesma. É importante, portanto, notar que se não houver filtragem na entrada do mecanismo de amostragem, quaisquer componentes de frequências (sinal ou ruído) que caiam fora da largura da faixa de Nyquist, em qualquer zona, terão sua componente de falseamento situada dentro da primeira zona. E, por essa razão, o filtro "antifalseamento" é utilizado em quase todos os ADCs para a remoção desses sinais indesejados. O processo de se amostrar um sinal fora da primeira zona de Nyquist é comumente referido como *subamostragem* ou *amostragem harmônica*.

2.3 FILTROS "ANTIFALSEAMENTO"

No processamento de sinais analógicos, utilizando-se sistemas de tempo discreto, em alguns casos é desejável que se minimize a taxa de amostragem. Isso reduz a quantidade de processamento requerido (proporcional ao número de amostras) no sistema. Se a entrada não tiver uma largura de faixa limitada ou se a frequência de Nyquist da entrada for muito alta, uma filtragem prévia pode ser necessária. Para se evitar o fenômeno de falseamento o sinal de entrada deve ser submetido a uma filtragem passa-baixa anteriormente à conversão A/D. Nesse contexto, o filtro passa-baixa é denominado filtro "antifalseamento" (Gregorian, 1999).

O conhecimento das características do sinal a ser amostrado é de extrema importância para a especificação adequada do filtro. Os filtros se tornam mais complexos à medida que as curvas de atenuação se tornam mais pronunciadas. Por exemplo, um filtro Butterworth fornece

6 dB de atenuação por oitava para cada polo. Assim, para se conseguir 60 dB de atenuação em uma região de transição entre 1 MHz e 2 MHz (1 oitava) haveria a necessidade de um mínimo de 10 polos, tornando o projeto do filtro nada trivial. Outros tipos de filtros se mostram mais adequados como, por exemplo, filtros elípticos (Kester, 2005).

Altas taxas de amostragem reduzem a necessidade de curvas de atenuação acentuadas, reduzindo a complexidade do filtro, a expensas de se utilizar um ADC mais rápido e de se processar os dados a taxas mais altas.

2.4 O SISTEMA DE AMOSTRAGEM DE DADOS

Os ADCs traduzem grandezas analógicas em um formato digital, utilizado no processamento de informação, computação, transmissão de dados e em sistemas de controle. Como parte do processo de conversão, as tensões e correntes são "normalizadas" em um faixa de valores compatível com os valores atribuíveis a um determinado ADC, cujos valores binários representam os estágios lógicos de diferentes níveis de corrente (menos usado) ou de tensão (Kester, 2005).

Conforme mencionado anteriormente, a entrada analógica (tipicamente uma tensão v_{IN}) é normalizada em relação a uma referência (tensão), V_{REF}, e a relação é convertida em uma palavra digital, B_{out}, de *N bits* contendo *b1, b2, ..., b_N*. Sob condições ideais, ignorando-se o ruído e imperfeições nos componentes, a relação entre v_{IN} e V_{REF} é tida como:

$$B_{out} = b_1.2^{-1} + b_2.2^{-2} + ... + b_N.2^{-N} = \frac{v_{\mathrm{IN}} + v_q}{V_{REF}} \qquad \text{(eq. 1)}$$

Na relação acima, o erro de quantização devido ao número finito de *bits* utilizado na conversão é representado por v_q. Tal erro é inerente ao processo e pode ser reduzido por meio do aumento de *N* ou da redução de V_{REF} (Gregorian, 1999).

Na tecnologia de conversão, o final de escala (FS) independe do número de *bits* de resolução. Quando os *bits* com os respectivos pesos são somados estes formam um número

com 2^N valores quaisquer, variando de 0 a $(1 - 2^{-N})$ do valor final de escala. Binários inteiros podem ser interpretados como binários fracionados se todos os valores inteiros forem divididos por 2^N (Kester, 2005).

2.4.1 DISPOSITIVO DE AMOSTRAGEM E RETENÇÃO

Os circuitos de amostragem e retenção, SHA ou S/H, são críticos na conversão de sinais analógicos em digitais. O comportamento de um S/H é análogo ao de uma câmera fotográfica e sua função principal é a de "registrar uma imagem" do sinal analógico e reter o seu valor até que o ADC possa processar a informação (Baker, 2008). A maioria dos conversores atuais possuem uma função de S/H embutida para auxiliar no processamento dos sinais em corrente alternada. Como exemplifica Kester (Kester, 2005), a figura 2.4 mostra um conversor por aproximações sucessivas (SAR) com *N* igual a 12 *bits*, f_s igual a 100 kSPS, tempo de conversão de 8 μs e que não possui a função S/H.

Supondo-se um sinal de entrada como uma onda senoidal cuja amplitude máxima seja representada por $p.2^N/2$, sendo *p* o peso do *bit* menos significativo (1 LSB), tem-se:

$$v(t) = p\,2^{N-1}\,sen(2\pi ft) \qquad \text{(eq. 2)}$$

Se o sinal de entrada variar em mais de 1 LSB durante o processo de conversão (8 μs no exemplo) o dado de saída pode conter erros consideráveis.

Tomando-se a primeira derivada de *v(t)*, da quação 2, tem-se:

$$\frac{dv}{dt} = 2\pi fp\,\frac{2^N}{2}\cos(2\pi ft) \qquad \text{(eq. 3)}$$

A taxa máxima de variação do sinal é, portanto:

$$\frac{dv}{dt}\Big|_{máx} = 2\pi fp\,\frac{2^N}{2} = p2^{N-1}\,2\pi f = p\pi 2^N f \qquad \text{(eq. 4)}$$

Da equação anterior, tem-se:

$$f = \frac{\frac{dv}{dt}|_{máx}}{p\pi 2^N} \qquad \text{(eq. 5)}$$

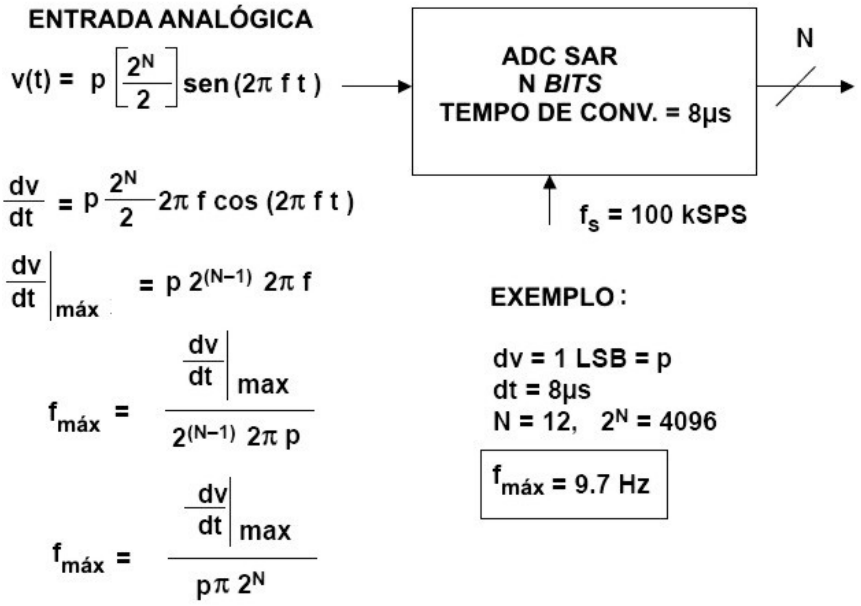

Figura 2.4: limitações na frequência de entrada em um ADC sem S/H (codificador).

Se N = 12 e uma mudança equivalente a 1 LSB ($dv = p$) ocorre durante o tempo de conversão ($dt = 8\ \mu s$), verifica-se um valor máximo na frequência do sinal que pode ser processado sem erros, conforme mostrado abaixo:

$$f_{máx} = \frac{\frac{p}{8\text{x}10^{-6}}}{p\pi 2^{12}} = 9{,}71\ Hz \qquad \text{(eq. 6)}$$

Esse resultado implica que qualquer frequência de entrada superior a 9,71 Hz estará sujeita a erros de conversão, ainda que uma frequência de amostragem de 100 kSPS seja possível com o ADC com tempo de conversão de 8 µs (restando ainda 2 µs).

Para que haja o processamento de sinais em corrente alternada, uma função de amostragem e retenção é adicionada ao sistema, como mostrado na figura 2.5. Um S/H ideal é formado simplesmente por uma chave, um capacitor de retenção seguido por um *buffer* com alta impedância de entrada. A impedância de entrada deve ser alta o suficiente para que o

capacitor se descarregue com valores menores do que 1 LSB durante o tempo de retenção (*hold*). O S/H amostra o sinal no modo de *amostragem* e mantém o sinal constante durante o modo de *retenção*. Os tempos são ajustados de modo que o codificador realize a conversão durante o modo de retenção. No exemplo mostrado anteriormente, o S/H deveria obter o sinal em 2 μs, permitindo uma frequência de amostragem de 100 kSPS, e a capacidade de se processar sinais com frequências de entrada de até 50 kHz (Kester, 2005).

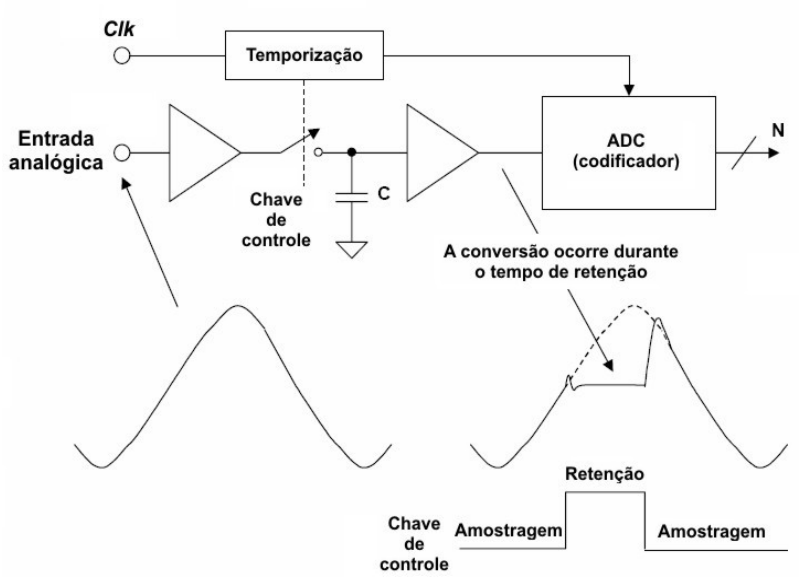

Figura 2.5: função de amostragem e retenção.

O sinal analógico é instantaneamente capturado e retido até o próximo período de amostragem. Não obstante, um período finito de tempo é necessário para que a amostragem ocorra (Baker, 2008).

Durante o período de amostragem o sinal analógico pode continuar variando, ensejando o surgimento de um outro tipo de circuito denominado circuito de "rastreamento e retenção" (THA ou T/H) no qual o sinal analógico é rastreado (*tracked*) durante o tempo requerido à amostragem (Baker, 2008). É importante notar que há uma sutil diferença entre um S/H e um T/H. Estritamente falando, a saída de um S/H não é definida durante o modo de amostragem. No entanto, a saída do T/H rastreia o sinal durante o modo de amostragem ou de rastreamento.

Na prática, a função é geralmente implementada como um T/H e os termos T/H e S/H são comumente usados de maneira intercambiável. As formas de onda vistas na figura acima são aquelas associadas a um dispositivo T/H (Kester, 2005).

2.4.2 CODIFICAÇÃO UNIPOLAR

Nos sistemas de conversão de dados o método de codificação deve estar relacionado à faixa de entrada analógica de um ADC. O caso mais simples é quando a entrada no conversor é sempre uma tensão positiva (unipolar). O código mais usual para esse tipo de sinal é o binário direto *(straight binary)*, mostrado na figura 2.6 (conversor de 4 *bits*). Nesse caso, há $2^N = 16$ níveis distintos, variando de um código 0000_2 até 1111_2. É importante notar que o valor analógico representado pelo código 1111_2 não é o final da escala (FS), mas *FS – 1* LSB. Essa é uma convenção comum na notação da conversão de dados e se aplica tanto para o ADC quanto para o DAC (Kester, 2005). Como o sinal de entrada é contínuo e a saída é discreta, a curva de transferência do conversor se assemelha a uma escada (Baker, 2008).

A figura 2.6 mostra o correspondente nível de tensão para cada código, assumindo-se uma tensão de final de escala de 3,3 V.

Decimal	Escala	+3,3 V FS	Binário
15	FS – 1 LSB = 15/16 FS	3,094	1111
14	14/16 FS	2,888	1110
13	13/16 FS	2,681	1101
12	12/16 FS	2,475	1100
11	11/16 FS	2,269	1011
10	10/16 FS	2,063	1010
9	9/16 FS	1,856	1001
8	8/16 FS	1,650	1000
7	7/16 FS	1,444	0111
6	6/16 FS	1,238	0110
5	5/16 FS	1,031	0101
4	4/16 FS	0,825	0100
3	3/16 FS	0,619	0011
2	2/16 FS	0,413	0010
1	1 LSB = 1/16 FS	0,206	0001
0	0	0,000	0000

Figura 2.6: código binário unipolar, conversor de 4 *bits*.

A função de transferência ou *curva de transferência* de um ADC de 3 *bits* ideal é mostrada na figura 2.7 (Kester, 2005). Há uma determinada faixa de tensão, do valor analógico de entrada, com a qual o conversor produz um determinado código de saída. Essa faixa corresponde à *incerteza de quantização* e é igual a 1 LSB. Em um ADC ideal, a largura das regiões de transição entre códigos adjacentes é zero. Na prática, no entanto, há sempre ruído de transição associado aos níveis produzindo larguras diferentes de zero. Costuma-se definir a entrada analógica correspondente a determinado código na metade de duas regiões de transição adjacentes produzindo, por exemplo, a ocorrência da primeira região de transição em ½ LSB. Cabe salientar que a real característica de transferência não é representada por uma linha, mas por um número de pontos discretos.

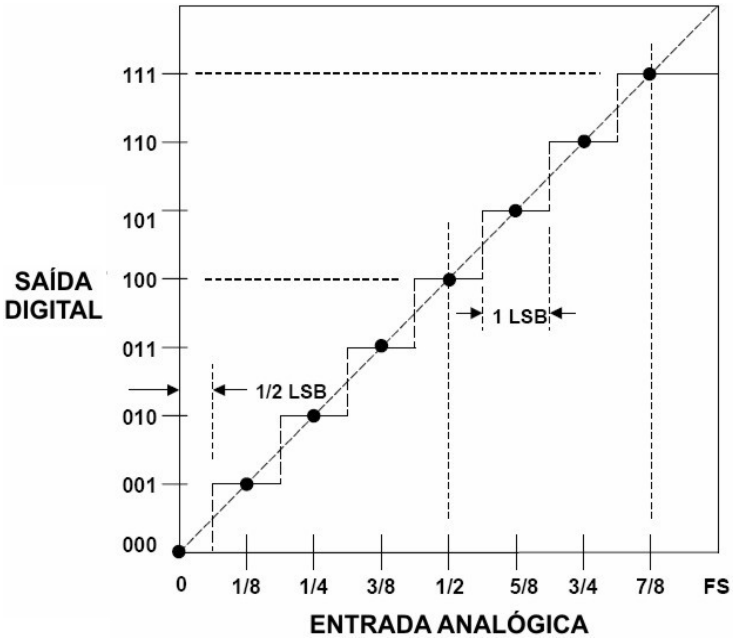

Figura 2.7: curva de transferência para um ADC unipolar de 3 *bits*.

2.4.3 CODIFICAÇÃO BIPOLAR

Em muitos sistemas é interessante a representação das quantidades analógicas positivas e negativas em códigos binários. Complemento de dois, complemento de um, magnitude com

sinal *(sign magnitude)* e binário com compensação atendem às necessidades. No entanto, o binário com compensação, bem como o complemento de dois são os códigos mais utilizados. A relação entre tais códigos é mostrada na figura 2.8, com representação para um sistema de 4 *bits*. Os valores de final de escala estão determinados em ± 3,3 V. Para o binário com compensação o valor de tensão zero é atribuído ao código 1000_2. A sequência de códigos é idêntica à do binário direto. A diferença entre esses sistemas é o valor de meia escala associado ao sinal analógico. O valor mais negativo (-FS + 1 LSB) é atribuído ao código 0001_2 e o valor mais positivo (+FS – 1 LSB) é atribuído ao código 1111_2. E, para que se mantenha uma perfeita simetria em relação ao meio da escala, o código 0000_2, representando -FS não é utilizado na maioria das aplicações. Pode ser utilizado para representar uma condição negativa fora da escala ou simplesmente ser atribuído com o valor 0001_2 (-FS + 1 LSB) (Kester, 2005).

Decimal	Escala	±3,3 V FS	Binário c/ comp.	Compl. de dois	Compl. de um	Magn. c/ sinal
7	+FS – 1LSB = 7/8 FS	2,888	1111	0111	0111	0111
6	6/8 FS	2,475	1110	0110	0110	0110
5	5/8 FS	2,063	1101	0101	0101	0101
4	4/8 FS	1,650	1100	0100	0100	0100
3	3/8 FS	1,238	1011	0011	0011	0011
2	2/8 FS	0,825	1010	0010	0010	0010
1	1/8 FS	0,413	1001	0001	0001	0001
0	0	0,000	1000	0000	*0000	*1000
-1	-1/8 FS	-0,413	0111	1111	1110	1001
-2	-2/8 FS	-0,825	0110	1110	1101	1010
-3	-3/8 FS	-1,238	0101	1101	1100	1011
-4	-4/8 FS	-1,650	0100	1100	1011	1100
-5	-5/8 FS	-2,063	0011	1011	1010	1101
-6	-6/8 FS	-2,475	0010	1010	1001	1110
-7	-FS + 1 LSB = -7/8 FS	-2,888	0001	1001	1000	1111
-8	-FS	-3,300	0000	1000		

Normalmente não usado

	Compl. de um	Magn. c/ sinal
* 0+	0000	0000
* 0-	1111	1000

Figura 2.8: codificação bipolar, conversor de 4 *bits*.

O complemento de dois é idêntico ao binário com compensação com o *bit* mais significativo (MSB) complementado (invertido). A popularidade do complemento de dois

ocorre pela facilidade no processamento das operações matemáticas em computadores e processadores digitais de sinais (DSP). Para efeito de conversão, o complemento de dois consiste no código binário para magnitudes positivas (*bit* de sinal em 0) e no complemento de dois de cada número positivo para se representar o respectivo negativo.

Complemento de um também pode ser usado na representação de números negativos, apesar de ser menos popular do que o complemento de dois e é raramente utilizado nos dias de hoje. O zero é representado como 0+ (código 0000_2) ou 0- (código 1111_2). Essa ambiguidade apresenta inconvenientes aos ADCs e DACs na medida em que estes utilizam apenas um código para representar o zero.

A magnitude com sinal poderia surgir como a representação mais natural para se expressar quantidades analógicas: determine o código apropriado para a magnitude e adicione um *bit* de polaridade. Magnitude com sinal no formato BCD (*binary-coded decimal*) é comum em voltímetros digitais bipolares porém, há o problema de dois códigos disponíveis para representação do zero. Desta forma, essa codificação também traz o mesmo inconveniente apresentado no complemento de um para aplicações de ADC e DAC.

2.4.4 ERROS ESTÁTICOS EM UM ADC

Um fato que se deve ter em mente em relação a um ADC é que a sua saída é digital e, consequentemente, o sinal é quantizado, ou seja, uma palavra de *N bits* representa um entre 2^N estados possíveis. Assim, um ADC tem que "quantizar" o sinal analógico (com infinitos valores) em somente 2^N níveis de quantização (saídas digitais possíveis).

A resolução de conversores de dados pode ser expressa de diferentes maneiras: peso do *bit* menos significativo, partes por milhão relativas ao final de escala (ppm FS), milivolts (mV), etc. O tamanho do LSB para várias resoluções é mostrado na figura 2.9 (Kester, 2005).

RESOLUÇÃO N	2^N	TENSÃO (10V FS)	ppm FS	% FS	dB FS
2-bit	4	2.5 V	250,000	25	– 12
4-bit	16	625 mV	62,500	6.25	– 24
6-bit	64	156 mV	15,625	1.56	– 36
8-bit	256	39.1 mV	3,906	0.39	– 48
10-bit	1,024	9.77 mV (10 mV)	977	0.098	– 60
12-bit	4,096	2.44 mV	244	0.024	– 72
14-bit	16,384	610 μV	61	0.0061	– 84
16-bit	65,536	153 μV	15	0.0015	– 96
18-bit	262,144	38 μV	4	0.0004	– 108
20-bit	1,048,576	9.54 μV (10 μV)	1	0.0001	– 120
22-bit	4,194,304	2.38 μV	0.24	0.000024	– 132

Figura 2.9: resolução e tamanho do ***bit*** menos significativo (LSB).

Tanto para um DAC quanto para um ADC o final de escala digital (todos os *bits* em "1") corresponde a 1 LSB abaixo do final de escala analógico (FS). Assumindo-se que a entrada analógica em um ADC pode apresentar qualquer valor e que, no entanto, a saída digital é quantizada, pode haver uma diferença de até ½ LSB entre o valor real da entrada analógica e o exato valor da saída digital, surgindo assim o erro de quantização. De fato, o processo de conversão A/D é muito mais difícil do que o processo de conversão D/A e muitas arquiteturas de conversores A/D utilizam os conversores D/A como elementos críticos.

O *erro de quantização* (ou *incerteza de quantização*), Q_e, é definido como a diferença entre a entrada analógica real e o valor da tensão de saída (escada) produzida:

$$Q_e = v_{\mathrm{IN}} - V_{escada} \qquad \text{(eq. 7)}$$

em que o valor de saída da escada, V_{escada}, pode ser calculado como:

$$V_{escada} = D.\frac{V_{REF}}{2^N} = D.V_{LSB} \qquad \text{(eq. 8)}$$

sendo D o valor do código digital de saída e V_{LSB} o valor de 1 LSB em volts.

O valor de Q_e pode ser gerado por meio da subtração do valor da escada e da linha pontilhada, como mostra a figura 2.10. Seria vantajoso se o erro de quantização fosse

centralizado em relação ao zero de maneira a se obter o erro em ±½ LSB. Isso é conseguido por meio do deslocamento da curva de transferência para a esquerda em ½ LSB, o que produz a centralização dos códigos em torno dos incrementos de LSB na abscissa (Baker, 2008).

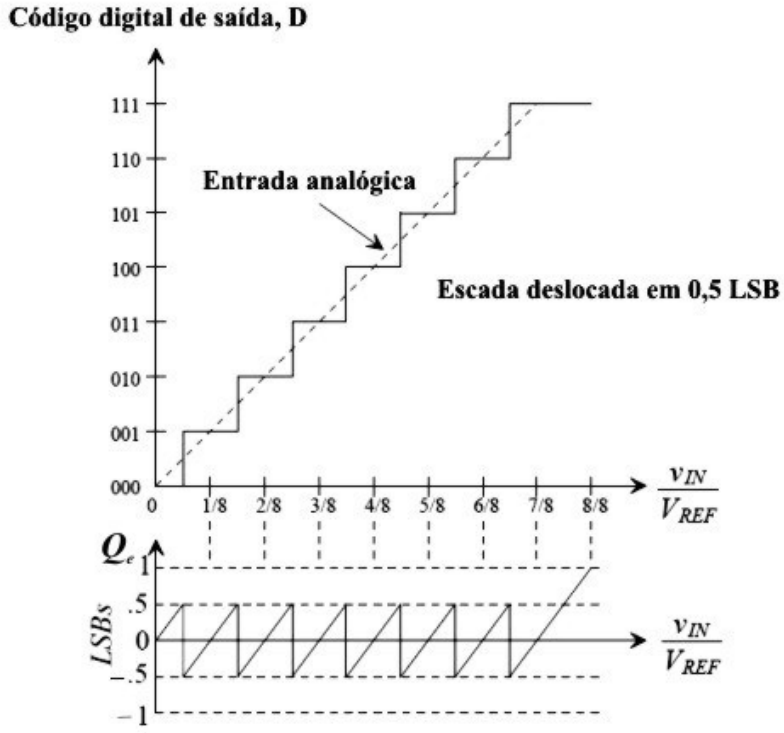

Figura 2.10: curva de transferência para um ADC com Q_e centralizado em relação ao zero.

Os quatro erros estáticos que assurgem em um conversor de dados são: erro de compensação (*offset*), erro de ganho e dois tipos de erros de linearidade (diferencial e integral).

A curva de transferência de um ADC (ou de um DAC) pode ser expressa como uma linha reta dada por $d = c + ga$, em que d representa o código digital, a o sinal analógico, c e g constantes. Em um conversor unipolar o valor ideal de c é zero. O erro de compensação consiste no valor pelo qual o valor real de c difere de seu valor ideal, ou seja, o erro de compensação ocorre quando há uma diferença entre o valor da primeira transição de código e o valor ideal das transições em ½ LSB. O valor do erro de compensação é constante e se torna ideal após a passagem pela tensão de compensação inicial. O erro de ganho consiste no valor pelo qual g difere de seu valor ideal, ou seja, corresponde à diferença entre a inclinação de uma

linha reta ao longo da curva de transferência e a inclinação de um conversor ideal. A figura 2.11 mostra os erros de compensação e de ganho para um conversor bipolar.

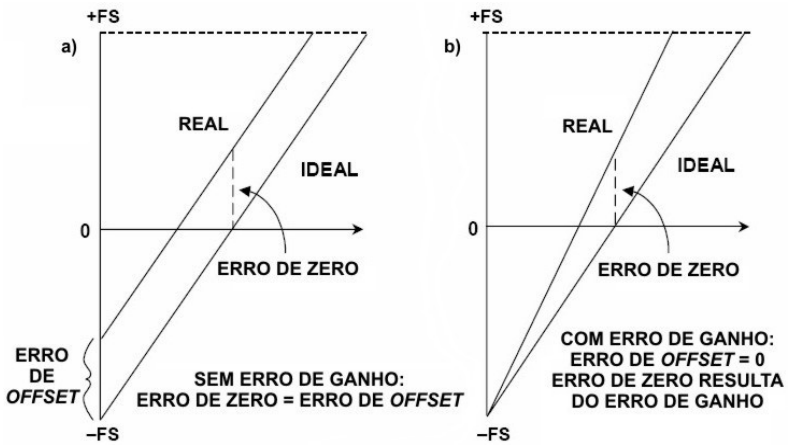

Figura 2.11: (a) erro de compensação e (b) erro de ganho em um conversor bipolar.

O erro de linearidade integral (INL) é definido como o máximo desvio da curva de transferência real de um conversor em relação a uma linha reta (Kester, 2005), ou seja, corresponde ao desvio total de um valor analógico em relação ao valor ideal (Gustavsson *et al*, 2002). Para um ADC, a convenção mais utilizada é a de se traçar uma linha reta ao longo dos meio pontos dos códigos, ou centros dos códigos. Há dois métodos usuais para se escolher a reta: o método do ponto terminal *(end point)* e o método da melhor linha reta *(best straight line),* como mostrado na figura 2.12.

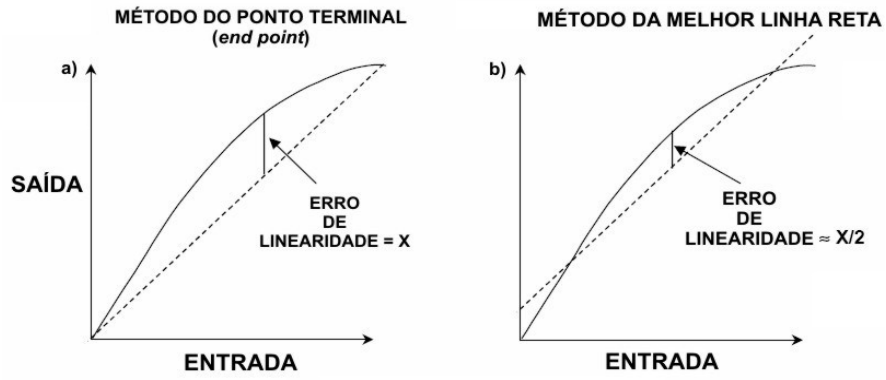

Figura 2.12: (a) método do ponto terminal e (b) método da melhor linha reta (mesmo conversor).

No sistema do ponto terminal, o desvio é medido a partir da linha traçada da origem até o valor de final de escala (após os ajustes de ganho). Esse é o método mais conveniente para a medição da linearidade integral de conversores de dados. No entanto, o método da melhor linha reta provê uma melhor predição da distorção em aplicações em corrente alternada. Em geral, esse método produz um erro de linearidade integral 50% mais baixo do que o valor medido pelo método do ponto terminal. Para aplicações em corrente alternada é melhor que se especifique distorção em vez de linearidade em sinais contínuos, e isso implica que é raramente necessário o uso do método da melhor linha reta para se definir a linearidade de um conversor (Kester, 2005).

O INL pode ser determinado, alternativamente, por meio da inspeção do erro de quantização, como mostrado na figura 2.13. O INL corresponde à magnitude do erro de quantização que surge fora da faixa de ½ LSB de Q_e. Verifica-se, portanto, que o valor Q_e = 1 LSB corresponde ao ponto em que o INL = 0,5 LSB para o código digital 011_2, e Q_e = -1 LSB corresponde ao ponto em que o INL = -0,5 LSB para o código digital 110_2 (Baker, 2008).

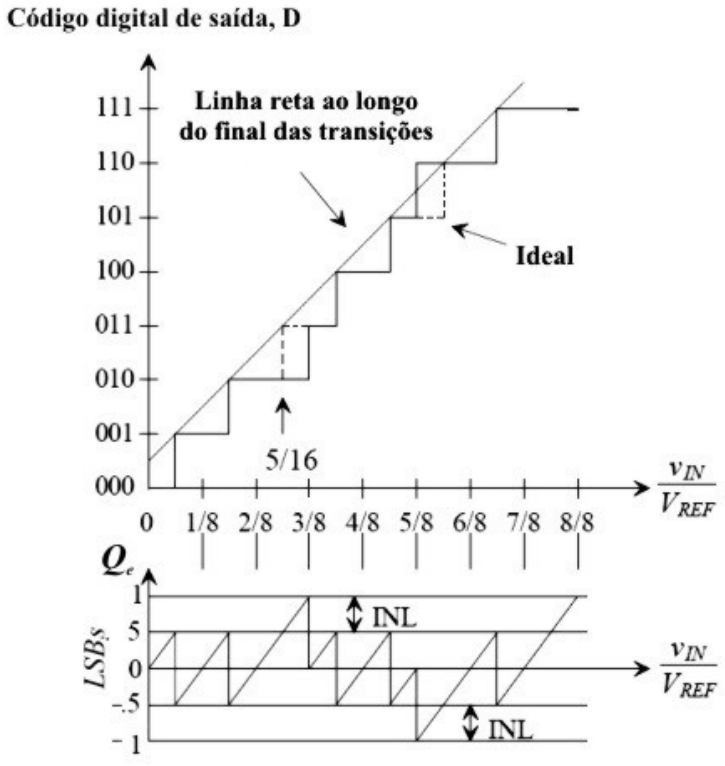

Figura 2.13: curva de transferência de um ADC de 3 *bits* e o Q_e ilustrando o INL.

O erro de linearidade diferencial (DNL) está relacionado à linearidade na transição de código do conversor. O tamanho do degrau em um conversor não ideal pode se desviar do tamanho ideal produzindo o erro (Gustavsson *et al*, 2002). Em um caso ideal, uma mudança de 1 LSB no código digital corresponde a uma mudança de 1 LSB no sinal analógico, ou seja, deve haver exatamente uma mudança correspondente a 1 LSB na entrada analógica para ocorrer uma transição de nível no sinal digital para o próximo. O erro de linearidade diferencial é definido portanto, como o máximo valor de desvio de uma quantidade qualquer, em toda a função de transferência, de seu tamanho ideal de 1 LSB, ou seja, corresponde à diferença entre a largura real de um código de um conversor não ideal e o seu correspondente ao caso ideal (Baker, 2008). Desta forma, quando uma mudança no sinal analógico, correspondente a uma mudança de 1 LSB no sinal digital, for maior ou menor do que o valor de 1 LSB, verifica-se um erro DNL. A figura 2.14 mostra os detalhes da não linearidade diferencial de um ADC (Kester, 2005).

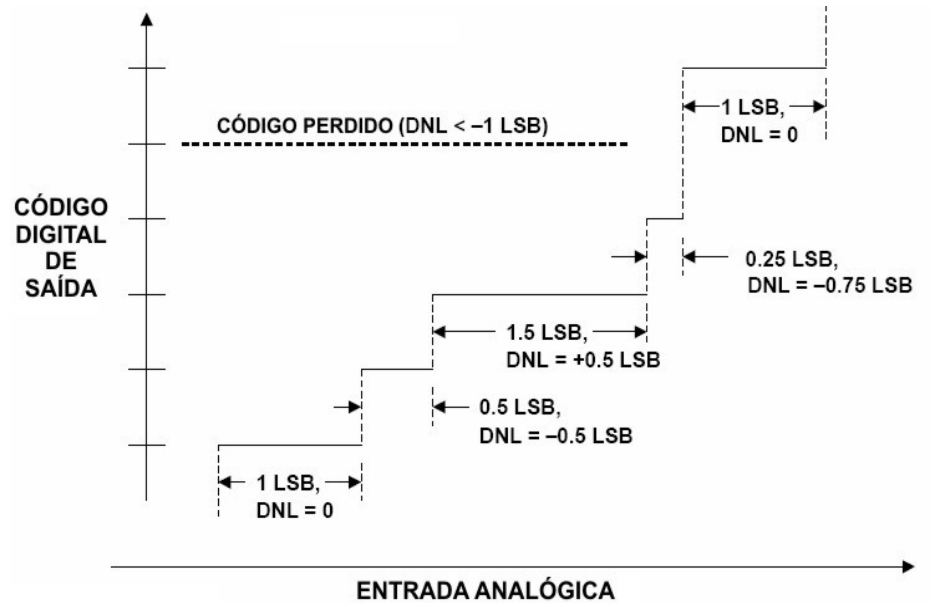

Figura 2.14: detalhes da não linearidade diferencial de um ADC.

Note-se que o erro de quantização está diretamente relacionado ao DNL. À medida que o DNL aumenta na direção de ± LSB, o erro de quantização piora. Erros de DNL podem

produzir a perda de códigos *(missing codes)* no momento da aquisição de dados. Assim, é interessante notar que determinado degrau cujo DNL tenha valor igual a -1 LSB apresenta a perda do código correspondente. Qualquer ADC que apresenta um DNL igual a -1 LSB garante a presença da perda de códigos. Na figura 2.15 se verifica que a largura total para o degrau correspondente à saída 101_2 se perdeu; disso, tem-se que o valor de DNL_5 é -1 LSB. A largura do degrau correspondente a 010_2 é 2 LSB e o valor para o DNL_2 é +1 LSB. No entanto, não há a perda do código correspondente a 011_2 na medida em que a largura do degrau do código 011_2 depende da transição de 100_2. Por essa razão, um ADC que apresenta um DNL maior do que +1 LSB não garante o surgimento da perda de códigos, apesar de não estar excluída a possibilidade (Baker, 2008).

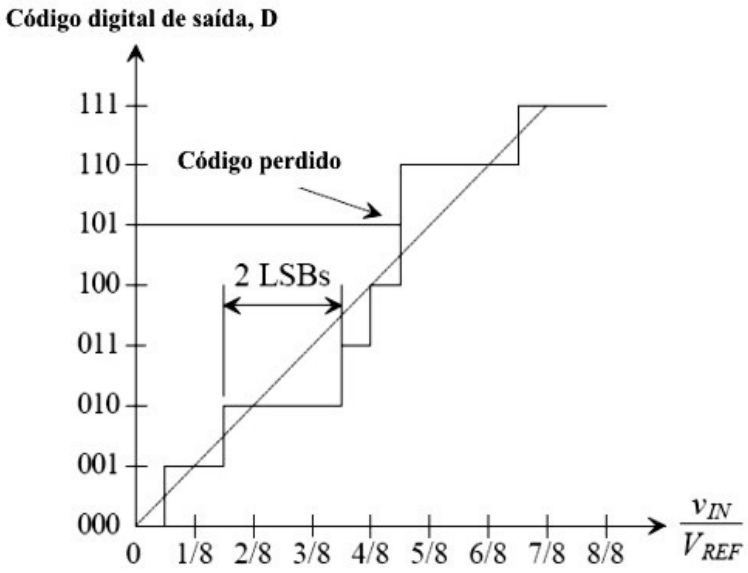

Figura 2.15: curva de transferência de um conversor A/D de 3 *bits* não ideal com perda de código.

2.4.5 ERROS DINÂMICOS EM UM ADC

Os erros dinâmicos associados a um conversor de dados ideal de *N bits* são aqueles relacionados aos processos de amostragem e quantização. O erro máximo que um conversor ideal produz ao digitalizar um sinal corresponde a ± ½ LSB. A curva de transferência de um ADC de *N bits* é mostrada na figura 2.16 (Bennet, 1948).

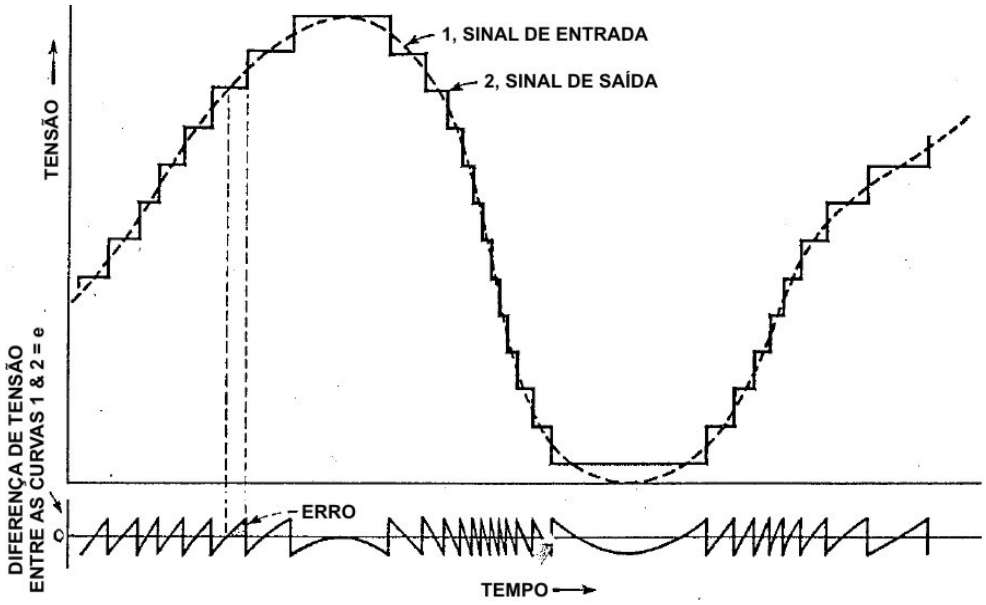

Figura 2.16: sinal quantizado e a correspondente forma de onda do erro.

O erro de quantização de um sinal alternado que se estende por mais do que uns poucos LSBs pode ser aproximado por um forma de onda dente de serra com amplitude *q*, pico a pico, correspondente ao peso de um LSB. Essa análise simplificada se mostra suficientemente acurada para a maioria dos casos de importância prática (Bennet, 1948).

A figura 2.17 mostra o erro de quantização como função do tempo. A equação da forma de onda do erro (dente de serra), com *s* correspondendo à inclinação (*slope*), é dada por:

$$e(t) = st, -\frac{q}{2s} < t < \frac{q}{2s} \qquad \text{(eq. 9)}$$

O valor quadrático médio de *e(t)* pode ser escrito como:

$$\overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{q/2s} (st)^2\,dt = \frac{s^3}{q}\int_{-q/2s}^{q/2s} t^2\,dt = \frac{s^3}{q}\left(\frac{q^3}{12s^3}\right) = \frac{q^2}{12} \qquad \text{(eq. 10)}$$

A raiz do erro quadrático médio do erro de quantização:

$$e_{RMS} = \sqrt{\overline{e^2(t)}} = \frac{q}{\sqrt{12}} \qquad \text{(eq. 11)}$$

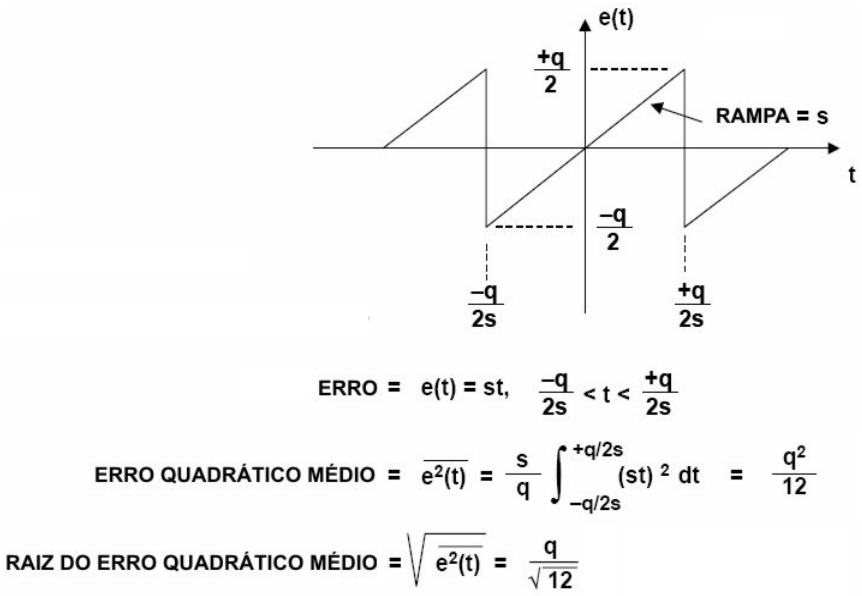

Figura 2.17: ruído de quantização como função do tempo.

O ruído de quantização mostrado é aproximadamente Gaussiano e se espalha de maneira aproximadamente uniforme ao longo da largura de faixa de Nyquist, de 0 Hz a $f_s/2$ (Bennet, 1948). A presunção subjacente que se faz aqui é de que o ruído de quantização é descorrelacionado com o sinal de entrada. Sob certas circunstâncias nas quais há correlação, o ruído de quantização se mostra concentrado indesejavelmente nas várias harmônicas do sinal de entrada (Kester, 2005).

A relação sinal-ruído (SNR) teórica pode ser calculada assumindo-se uma forma de onda de entrada como segue:

$$v(t) = q\frac{2^N}{2}sen(2\pi ft) \qquad \text{(eq. 12)}$$

Calculando-se o valor eficaz do sinal, tem-se:

$$v_{RMS} = \left(\frac{q2^N}{2\sqrt{2}}\right) \qquad \text{(eq. 13)}$$

O valor eficaz da relação sinal-ruído para um conversor ideal de *N bits* é mostrada a seguir:

$$SNR = 20\log\left(\frac{valor\ eficaz\ do\ sinal\ de\ entrada}{valor\ eficaz\ do\ ruído\ de\ quantização}\right) \qquad \text{(eq. 14)}$$

$$SNR = 20\log\left(\frac{q2^N/2\sqrt{2}}{q/\sqrt{12}}\right) = 20\log\left(\frac{2^N\sqrt{12}}{2\sqrt{2}}\right) = 20\log 2^N + 20\log\left(\frac{\sqrt{12}}{2\sqrt{2}}\right) \qquad \text{(eq. 15)}$$

Quando medida na largura de faixa de Nyquist (0 Hz até $f_s/2$), a relação sinal-ruído é tida como:

$$SNR = 6{,}02N + 1{,}76\,dB \qquad \text{(eq. 16)}$$

Se, no entanto, a largura de faixa, *W*, do sinal for menor do que $f_s/2$, a SNR será aumentada devido à menor quantidade de ruído de quantização confinada em *W*. A expressão da relação sinal-ruído para esse tipo de condição é dada por:

$$SNR = 6{,}02N + 1{,}76\,dB + 10\log\left(\frac{f_s}{2\mathrm{W}}\right) \qquad \text{(eq. 17)}$$

2.4.5.1 DISTORÇÃO HARMÔNICA

Há várias maneiras de se medir a distorção de um ADC. Uma análise da transformada rápida de Fourier (FFT) do sinal pode ser usada para se medir a amplitude de suas diferentes harmônicas que se distinguem de outros produtos de distorção por meio de sua localização no espectro de frequências. A figura 2.18 mostra um sinal de 7 MHz amostrado em 20 MSPS, juntamente com a localização das nove primeiras harmônicas. Com o fenômeno de falseamento, as harmônicas de f_a estão dispostas em frequências correspondentes a $|\pm \mathrm{K}f_s \pm nf_a|$, em que *n* representa a ordem da harmônica e $K = 0, 1, 2, 3, \ldots$ Para um valor de $K = 1$ e $n = 2$, por exemplo, tem-se a localização das harmônicas em 6 MHz e 34 MHz.

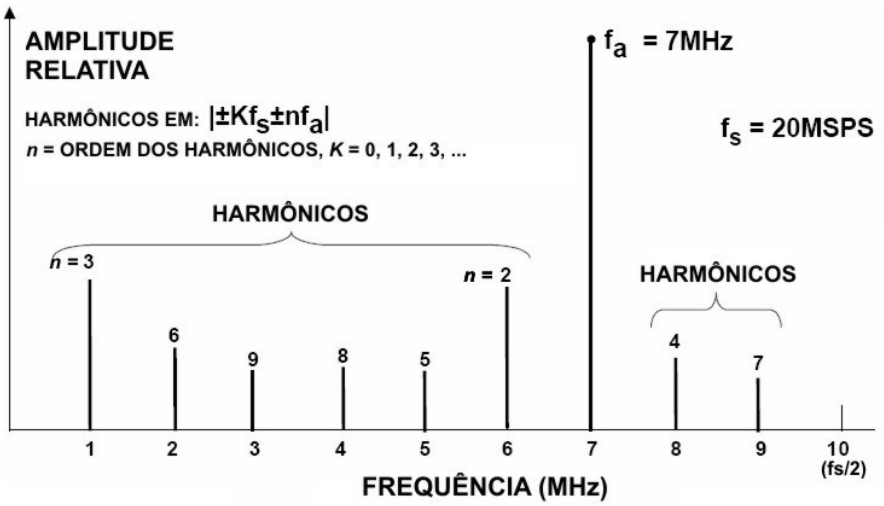

Figura 2.18: situação dos produtos de distorção harmônica.

A distorção harmônica, normalmente em dBc (dB abaixo da portadora – *carrier*), é geralmente especificada com um sinal de entrada próximo ao valor de final de escala (geralmente 0,5 a 1 dB abaixo para evitar recortes na amplitude máxima do sinal), mas pode ser especificada em qualquer nível. A distorção harmônica total (THD) é a medida da distorção harmônica presente, correspondendo à razão entre a soma das potências de todas as harmônicas e a frequência fundamental. Geralmente, apenas as primeiras cinco harmônicas têm significância (Kester, 2005).

A distorção harmônica total acrescida do ruído (THD + N) corresponde à THD com adição de todos os componentes de ruído (excluindo-se o CC). A largura de faixa na qual o ruído é medido, para o caso de uma análise de FFT, compreende 0 Hz até $f_s/2$, configurando-se a relação sinal-ruído e distorção (SINAD). A SINAD corresponde à relação entre a potência do sinal e a potência combinada de todos os componentes de distorção e ruído.

2.4.5.2 NÚMERO EFETIVO DE *BITS*

O número efetivo de *bits* (ENOB) corresponde ao número de *bits* no sinal digitalizado que se encontra acima do patamar de ruído (*noise floor*). Utilizando-se a relação teórica da SNR de um ADC ideal de *N bits* (SNR = 6,02*N* + 1,76), pode-se obter o ENOB a partir do valor medido do SINAD, que é substituído pelo valor de SNR, como segue:

$$ENOB = \frac{SINAD - 1{,}76\,dB}{6{,}02} \qquad \text{(eq. 18)}$$

O ENOB pode ser calculado se utilizando a relação sinal-ruído (SNR sem harmônicas), excluindo-se os harmônicos e deixando somente os termos de ruído. Na prática, faz-se necessária a exclusão das cinco primeiras harmônicas por serem dominantes. Nesse caso, para um ADC ideal de *N bits*, tem-se:

$$N = \frac{SNR - 1{,}76\,dB}{6{,}02} \qquad \text{(eq. 19)}$$

Utilizando-se a definição do ENOB torna-se fácil a comparação entre ADCs ou DACs com o mesmo número de *bits*, mas devido a diferenças na implementação dos circuitos, tais conversores apresentam diferentes desempenhos (Plassche, 2003).

2.4.5.3 FAIXA DINÂMICA LIVRE DE ESPÚRIOS

Quando os conversores são usados com elevadas taxas de amostragem ou a pureza espectral do conversor é importante, pode se obter uma especificação adicional determinando a relação entre a máxima componente do sinal e a maior componente de distorção. Essa relação é denominada faixa dinâmica livre de espúrios (SFDR) (Plassche, 2003). A SFDR é comumente definida como a diferença entre a amplitude da frequência fundamental do sinal e a amplitude da componente espúria mais elevada medida na largura de faixa de interesse (faixa de Nyquist, nesse caso). SFDR é geralmente representada graficamente como uma função da amplitude do

sinal. Pode ser expressa em dBc (relativa à amplitude do sinal de entrada) ou em dBFS, nesse caso relativamente ao final de escala do ADC, como mostrado na figura 2.19 (Kester, 2005).

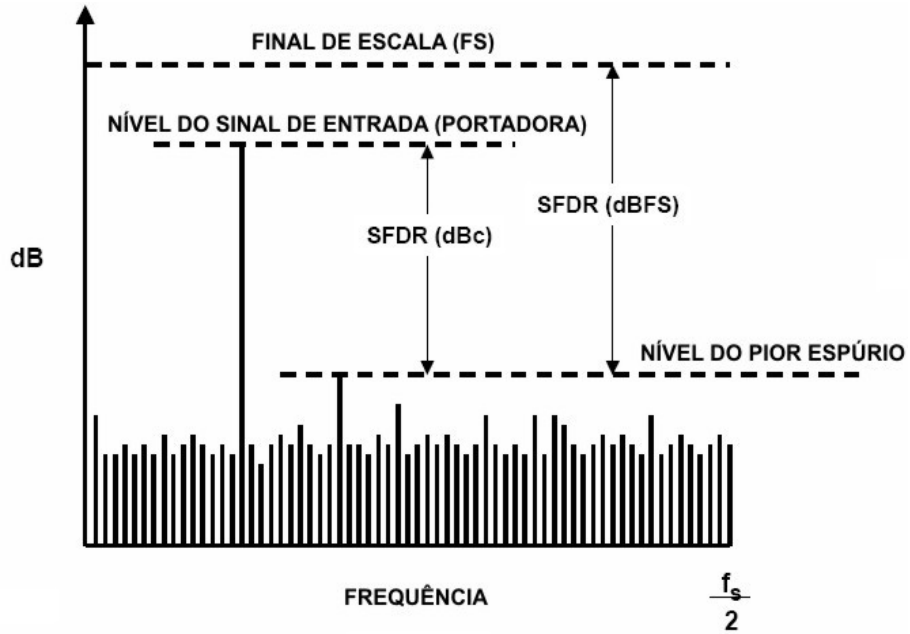

Figura 2.19: faixa dinâmica livre de espúrios.

Conversores com uma boa linearidade integral geralmente fornecem uma SFDR maior do que a relação sinal-ruído do sistema (Plassche, 2003).

3 PRINCIPAIS ARQUITETURAS DOS CONVERSORES DE DADOS

As arquiteturas do ADC descritas a seguir compreendem aquelas cujo maior esforço de desenvolvimento tem sido empregado, a saber: conversores instantâneos ou *flash*, aproximações sucessivas, *pipeline* e sigma-delta (Σ-Δ).

3.1 INSTANTÂNEO (FLASH)

Em muitas aplicações há a necessidade de se ter o menor tempo de conversão possível. Isso levou ao desenvolvimento de conversores de alta velocidade que utilizam técnicas paralelas para se conseguir tempos curtos de conversão (Geiger *et al*, 1990). O ADC *flash* ou paralelo é o tipo mais rápido em comparação a qualquer outro conversor. Utiliza um comparador por nível de quantização ($2^N - 1$) e 2^N resistores, como mostrado na figura 3.1 (Baker, 2008).

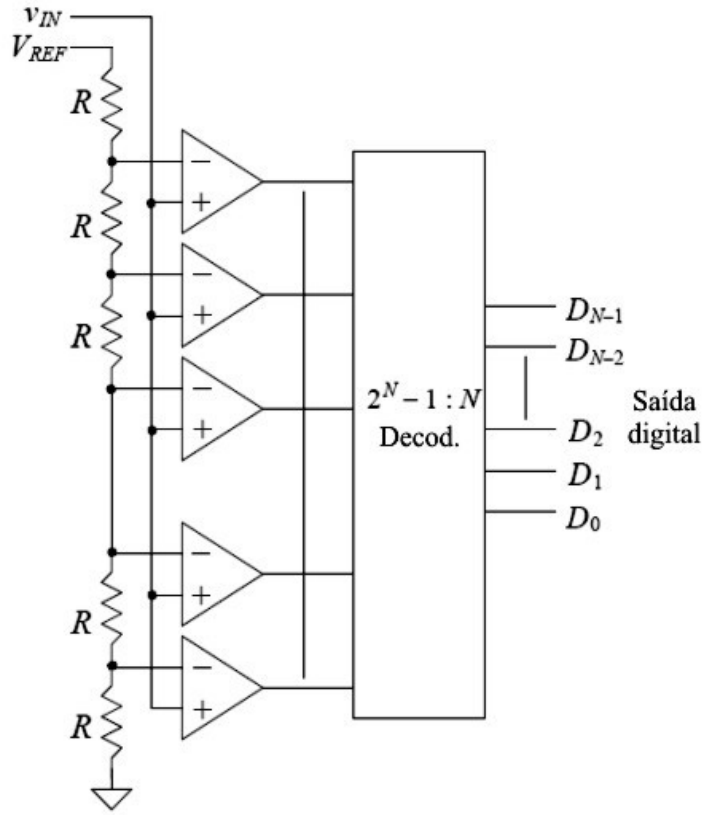

Figura 3.1: conversor paralelo (*flash*).

Cada comparador possui uma tensão de referência de uma cadeia de resistores que equivale a 1 LSB maior do que do comparador mais baixo na cadeia. Para uma dada tensão de entrada, todos os comparadores abaixo de determinada posição terão suas tensões de entrada maiores do que suas respectivas tensões de referência e apresentarão na saída o nível lógico "1". Os comparadores acima da posição mencionada terão uma tensão de referência maior do que a tensão de entrada e apresentarão na saída o nível lógico "0". As saídas dos 2^N - 1 comparadores se comportam de maneira análoga a um termômetro de mercúrio e, desta forma, o código de saída é, às vezes, denominado *código termômetro*. As 2^N – 1 saídas de dados são processadas por um decodificador que produz uma saída binária de *N bits*.

O conversor paralelo mostrado converte um sinal analógico em uma palavra digital em um ciclo de *clock*, com duas fases por período. Durante a primeira fase a tensão analógica de entrada é amostrada e aplicada às entradas do comparador. Durante a segunda fase é determinada a palavra digital correta. Assim, a conversão está limitada em quão rápida é essa sequência de eventos (Geiger *et al*, 1990). O grande número de comparadores conectados a v_{IN} resulta em capacitâncias parasitas que geralmente limitam a velocidade do conversor (Johns *et al*, 1997).

Por utilizar um grande número de resistores e comparadores, esse tipo de arquitetura está limitada a baixas resoluções. Os problemas inerentes ao *flash* incluem resolução limitada, alta dissipação de potência devido ao grande número de comparadores de alta velocidade (especialmente em taxas de amostragem maiores do que 50 MSPS) e uma demanda por área de *chip* relativamente alta, implicando também na elevação do custo de fabricação.

Na prática, os conversores *flash* estão disponíveis em até 10 *bits*, sendo os mais comuns com 8 *bits* de resolução. As taxas máximas de amostragem podem chegar até 1 GSPS (geralmente disponíveis nos processos que empregam GaAs), com alto consumo de energia.

3.2 APROXIMAÇÕES SUCESSIVAS (SAR)

O ADC por aproximações sucessivas tem sido o esteio da aquisição de dados por muitos anos. A sua simplicidade permite tanto alta resolução quanto alta velocidade, mantendo ao mesmo tempo uma área de *chip* relativamente pequena.

A figura 3.2 mostra os componentes básicos do conversor cuja operação se baseia em N comparações sucessivas entre a entrada analógica, v_{IN}, e a tensão de realimentação, V_f. É análoga a um processo de pesagem no qual a quantidade desconhecida é comparada a uma quantidade de referência. A primeira comparação determina se v_{IN} é maior ou menor do que $\frac{1}{2}V_{máx}$, em que $V_{máx}$ representa o máximo valor de entrada no conversor. O próximo passo determina em qual ¼ da variação se encontra v_{IN}. Em cada etapa sucessiva ocorre um estreitamento na faixa de possíveis resultados por um fator 2 (Barna *et al*, 1973; Hnatek, 1976).

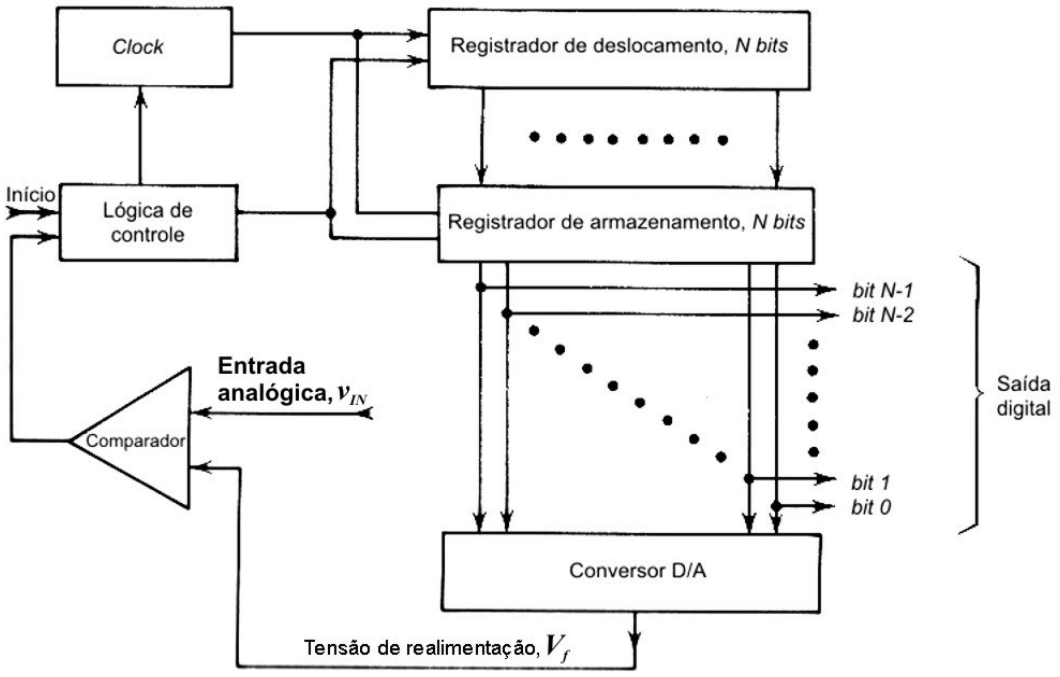

Figura 3.2: diagrama esquemático do ADC por aproximações sucessivas.

O ciclo de conversão se inicia por meio da amostragem do sinal analógico a ser convertido. A seguir, o circuito da "lógica de controle" assume que o MSB é "1" e que os

demais *bits* são “0”. A palavra digital é aplicada ao “conversor D/A”, que produz um sinal analógico de $½V_{REF}$, que é comparado à entrada analógica amostrada, v_{IN}. Se a saída do comparador for alta, a lógica de controle determina que o MSB seja “1”. Se a saída do comparador for baixa, a lógica de controle determina que o MSB seja “0”. Isso completa o primeiro passo na sequência de aproximação. O processo continua com o teste do segundo *bit* e assim sucessivamente até que todos os *bits* da palavra digital tenham sido definidos pelo processo de aproximação sucessiva (Geiger *et al*, 1990). O fim da conversão é geralmente indicada por um sinal de *fim de conversão* (EOC).

A “lógica de controle” da figura 3.2 é comumente chamada de *registrador por aproximações sucessivas* (SAR) (Geiger *et al*, 1990).

Não necessariamente um conversor de 16 *bits* gasta duas vezes o tempo de conversão correspondente a um conversor de 8 *bits*. Na prática, um ADC por aproximações sucessivas de 8 *bits* pode concluir a sua conversão em centenas de nanossegundos ao passo que um de 16 *bits* irá geralmente levar vários microssegundos. Isto se deve ao fato da necessidade de estabilização dos *bits* antes da tomada de decisão, o que envolve grupamentos de 8 e 16 *bits*, tornando a relação não necessariamente linear (Kester, 2005).

O limite de acurácia do ADC depende grandemente da acurácia do DAC. Se o DAC não produzir a tensão analógica correta com a qual será comparada a tensão de entrada, toda a saída do conversor conterá erros (Baker, 2008).

Um tipo popular de implementação do SAR ADC utiliza um arranjo de capacitores com pesos binários (totalizando $2^N C$) como DAC, como mostrado na Figura 3.3 (McCreary *et al*, 1975; Gregorian, 1999; Baker, 2008). A variável C corresponde ao capacitor unitário e pode assumir qualquer valor, conforme as necessidades do projeto. O conversor amostra o sinal de entrada e então executa uma busca binária baseando na quantidade de carga em cada capacitor do DAC.

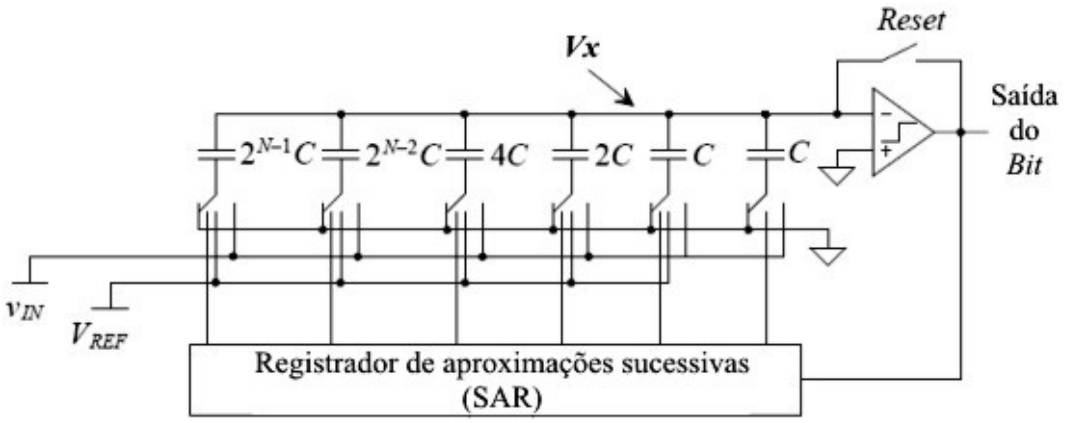

Figura 3.3: ADC por aproximações sucessivas com redistribuição de carga.

A operação do conversor pode ser dividida em quatro fases distintas: fase de cancelamento da tensão de compensação na qual a chave "Reset" é fechada permitindo a descarga dos capacitores, bem como o cancelamento automático da tensão de compensação do comparador*;* fase de amostragem ("Reset" permanece fechada) na qual as placas inferiores dos capacitores são ligadas à tensão de entrada v_{IN}; fase de retenção (*hold*) na qual a chave "Reset" é aberta e as placas inferiores dos capacitores são ligadas ao terra. Assim, $Vx = -v_{IN} + V_{OS}$ (tensão de compensação); fase de redistribuição de carga: o processo se inicia quando a placa inferior do capacitor correspondente ao MSB é ligada em V_{REF}. Se a saída do comparador for alta (nível alto) a placa inferior do capacitor MSB permanece conectada a V_{REF}, caso contrário é conectada ao terra. A tensão em $Vx = -v_{IN} + V_{OS} + D_{N-1}(V_{REF}/2)$, e sucessivamente $Vx = -v_{IN} + V_{OS} + D_{N-1}(V_{REF}/2) + D_{N-2}(V_{REF}/4)$... até que haja a convergência aproximada ao valor de V_{OS}, que corresponde à tensão de compensação do comparador.

À medida que a resolução aumenta o tamanho do capacitor correspondente ao MSB se torna um problema, demandando uma área considerável no *chip*. Uma maneira de se reduzir o tamanho dos capacitores consiste em repartir ou dividir o arranjo (*split array*). Nesse caso, um capacitor adicional de atenuação é usado para separar o arranjo, obtendo-se um arranjo menos

significativo (LSB) e um arranjo mais significativo (MSB). A Figura 3.4 mostra a conformação de um arranjo dividido (6 *bits*) (Baker, 2008).

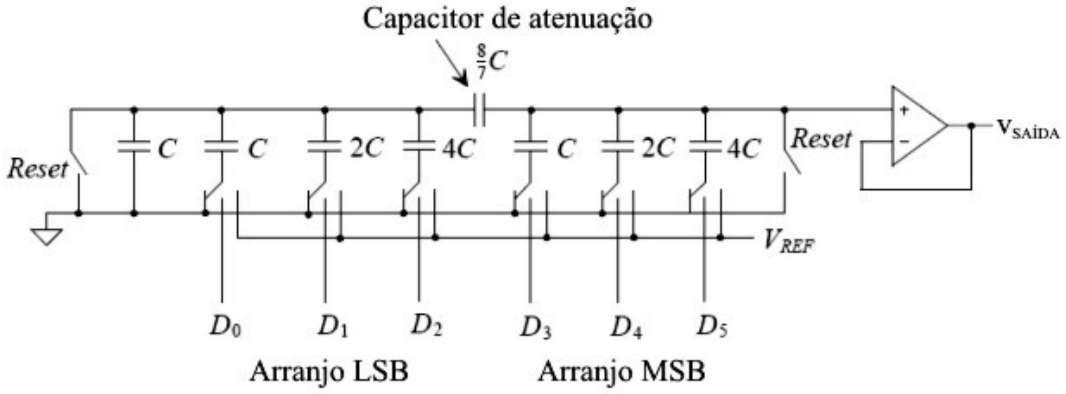

Figura 3.4. DAC (6 *bits*) utilizando divisão do arranjo de capacitores.

O valor do capacitor de atenuação pode ser obtido por:

$$C_{atenuação} = \frac{soma\ do\ arranjo\ LSB}{soma\ do\ arranjo\ MSB} . C \qquad \text{(eq. 20)}$$

3.3 *PIPELINE*

O *pipeline* é um conversor de *N* passos, com 1 *bit* sendo convertido por estágio. Consistindo em *N* estágios conectados em série, consegue atingir alta resolução (10 a 13 *bits*) em relativas altas velocidades. Conforme mostrado na figura 3.5, cada estágio contém um ADC de 1 *bit* (um comparador), um S/H, um somador e o ganho de dois amplificadores.

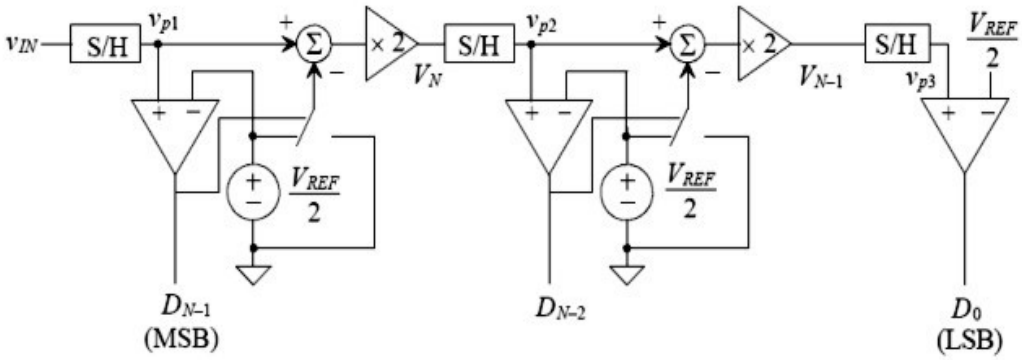

Figura 3.5: diagrama de blocos do ADC *pipelined*.

Como mostra Baker (Baker, 2008), cada estágio perfaz as seguintes operações:

1. Depois de amostrado o sinal de entrada, este é comparado a $V_{REF}/2$. A saída de cada comparador corresponde à conversão do bit para aquele estágio;
2. Se $v_{IN} > V_{REF}/2$ (saída do comparador igual a 1), $V_{REF}/2$ é subtraído do sinal retido e o resultado é passado ao amplificador. Se $v_{IN} < V_{REF}/2$ (saída do comparador igual a 0), o sinal de entrada original é passado ao amplificador. A saída de cada estágio do conversor é denominada *resíduo*;
3. O resultado da soma é multiplicado por 2 e é passado para o S/H do próximo estágio.

Apesar de levar N ciclos de *clock* para processar cada sinal de entrada (latência N), uma nova amostra pode ser admitida no *pipeline* a cada ciclo. Assim, a taxa de processamento é de uma amostra/ciclo, mas a complexidade é somente proporcional a N, o que torna o conversor *pipeline* uma boa escolha quando é importante uma demanda por área menor no *chip* (Johns *et al*, 1997).

Um aspecto interessante nesse conversor é a necessidade de se ter grande acurácia nos primeiros estágios (*bits* mais significativos). Um pequeno erro no primeiro estágio se propaga pelo conversor, resultando em um erro muito maior no final da conversão. Cada estágio subsequente requer menos acurácia do que o estágio anterior, de forma que um cuidado especial deve ser tomado em consideração aos primeiros estágios (Baker, 2008).

3.4 SIGMA-DELTA (Σ-Δ)

Em 1962, Inose, Yasuda e Murakami (Inose *et al*, 1962, 1963) trabalharam na arquitetura proposta por C. C. Cutler, em 1954, que introduziu o princípio da "sobreamostragem" (*oversampling*) e "moldagem do ruído" (*noise shaping*). Os circuitos experimentais elaborados por eles utilizavam dispositivos de estado sólido para implementarem moduladores sigma-delta de primeira e segunda ordem. Nessa época surgiu o nome delta-sigma que perdurou até a década de 70 quando foi modificado para sigma-delta pelos engenheiros da

AT&T. É interessante notar que até então todos os trabalhos estavam relacionados à transmissão do sinal digital "sobreamostrado" diretamente, não sendo vislumbrada a implementação de um ADC com taxa de Nyquist. Em 1969, D. J. Goodman descreveu um ADC sigma-delta com um filtro digital e um dizimador seguindo o modulador (Kester, 2005). Apesar de os ADCs com taxa de Nyquist serem bastante rápidos, sua resolução está limitada a 10 ou 12 *bits* com as tecnologias atuais devido a problemas na equalização entre componentes e não idealidades nos circuitos (Allen *et al*, 2002).

A figura 3.6 mostra o ADC sigma-delta básico de 1 *bit* e de múltiplos *bits* (figuras 3.6-a e 3.6-b, respectivamente).

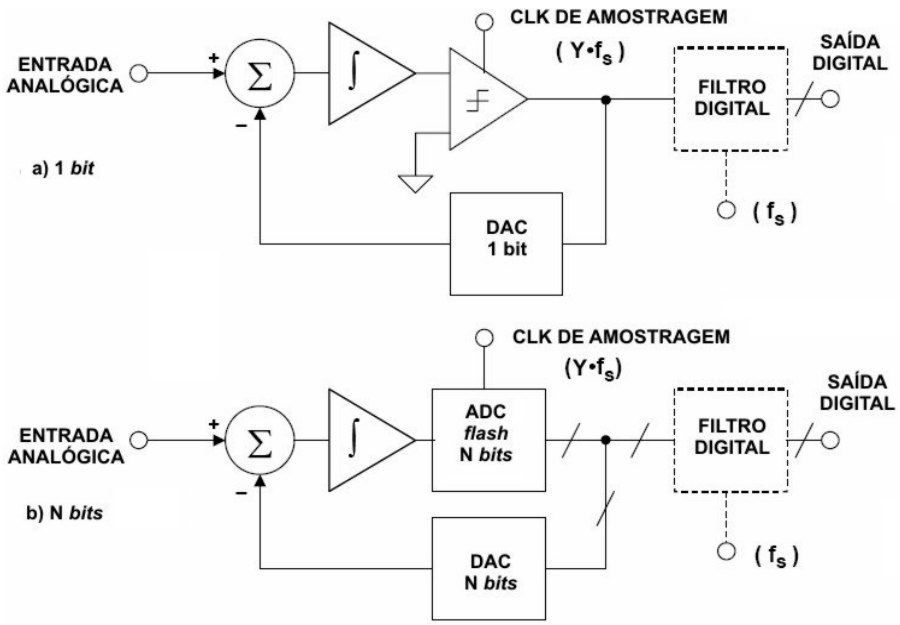

Figura 3.6: ADC sigma-delta de 1 *bit* e N *bits*.

O modulador sigma-delta básico aumenta a relação sinal-ruído em baixas frequências moldando o ruído de quantização de tal forma que grande parte deste ocorra fora da largura de faixa de interesse. O filtro digital então remove o ruído localizado fora da largura de faixa de interesse e o *dizimador* (*decimator*) reduz a taxa de dados de saída de volta à taxa de Nyquist (Kester, 2005).

"Sobreamostragem" e "moldagem do ruído" representam as duas técnicas chave

empregadas nesses conversores. Diferentemente dos conversores com taxa de Nyquist, nos quais cada palavra digital é obtida da quantização "exata" de uma amostra de entrada, nos conversores com "sobreamostragem" cada saída é obtida de uma sequência de amostras de entrada grosseiramente quantizadas. Assim, a operação de amostragem é feita em uma taxa muito mais alta do que na taxa de Nyquist (Allen *et al*, 2002).

A figura 3.7-a mostra uma análise no domínio da frequência na qual um ADC ideal de *N bits* apresenta um ruído de quantização eficaz de $q/\sqrt{12}$ uniformemente distribuído na faixa de Nyquist, de 0 Hz até $f_s/2$, sendo q o valor de um LSB e f_s a frequência de amostragem. No entanto, se o ADC não é ideal e o ruído ultrapassa o ruído de quantização teórico mínimo, então a sua resolução real será menor do que *N bits* e, desta forma, será tida como:

$$ENOB = \frac{SNR - 1{,}76}{6{,}02} \qquad \text{(eq. 21)}$$

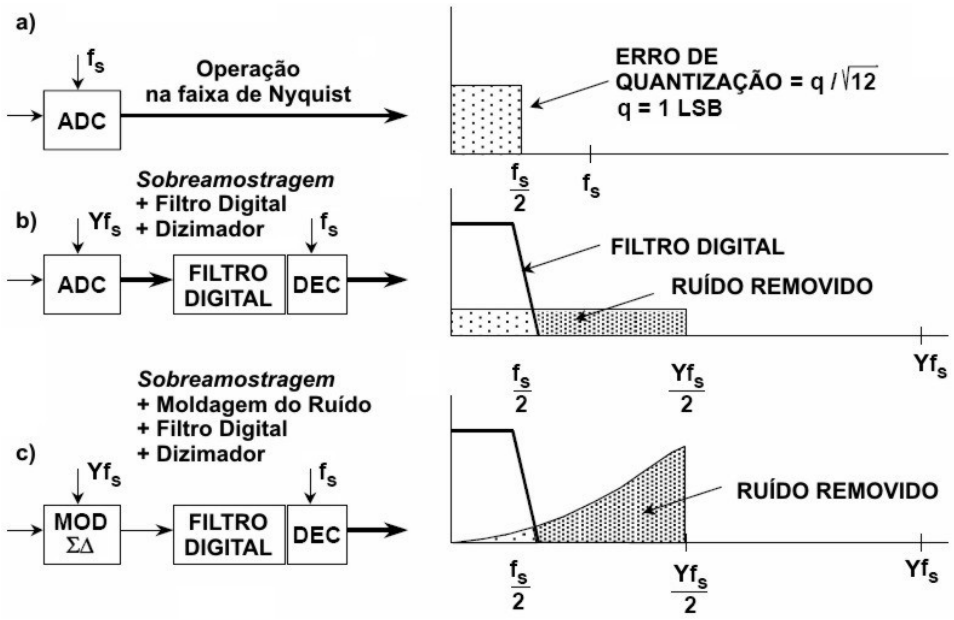

Figura 3.7: a) ruído de quantização; b) "*sobreamostragem"*, filtragem digital e dizimação; c) "*sobreamostragem"*, filtragem digital, dizimação e "moldagem do ruído".

A figura 3.7-b mostra o caso em que é escolhida uma taxa de amostragem mais alta, Yf_s, de maneira que o valor eficaz do ruído de quantização permaneça $q/\sqrt{12}$ distribuído, porém, em uma faixa maior (0 Hz até $Yf_s/2$). Aplicando-se um filtro digital passa baixa à saída, grande parte do ruído de quantização é removido sem, no entanto, afetar o sinal desejado, melhorando-se assim, o ENOB. Isso permite a realização de uma conversão A/D de alta resolução com um ADC de baixa resolução. O fator Y é geralmente denominado *razão de* "sobreamostragem". Observa-se nesse caso que a "sobreamostragem" traz um benefício adicional que é o relaxamento dos requisitos do filtro analógico "antifalseamento".

Uma vez que o filtro digital reduz a largura de faixa, a taxa de dados de saída pode ser menor do que a taxa de amostragem original (Yf_s) e ainda assim satisfazer o critério de Nyquist. Isso pode ser conseguido por meio da passagem de cada *M-ésimo* resultado para a saída, descartando-se o restante em um processo conhecido como "dizimação" por um fator M. A despeito de sua origem (elemento de composição *decim-*, antepositivo do latim *decimus*), M pode assumir qualquer valor inteiro, desde que a taxa de dados de saída seja maior do que duas vezes a largura de faixa do sinal. A dizimação não produz qualquer perda de informação.

A figura 3.7-c mostra o funcionamento do conversor Σ-Δ, que não necessita de uma relação de "sobreamostragem" muito alta pois este não somente limita a faixa de passagem do sinal, como também molda o ruído de quantização de tal maneira que a maior parte do mesmo permaneça fora da faixa de passagem do sinal. Utilizando-se simplesmente a "sobreamostragem" para melhorar a resolução, necessita-se de um fator 2^{2N} para se obter um aumento de N *bits* na resolução (Kester, 2005). Assim, para aumentar determinada resolução em 4 bits, por exemplo, a razão de "sobreamostragem" deve ser aumentada 256 vezes (Allen, 2002).

Tomando-se um ADC de 1 *bit* (um comparador), alimentado pela saída de um integrador e alimentando-se o integrador com um sinal de entrada somado com a saída de um DAC de 1 *bit*, alimentado pela saída do ADC, tem-se o modulador Σ-Δ de primeira ordem,

como mostrado na figura 3.8. Adicionando-se um filtro digital passa-baixa e o dizimador na saída digital, tem-se um ADC Σ-Δ.

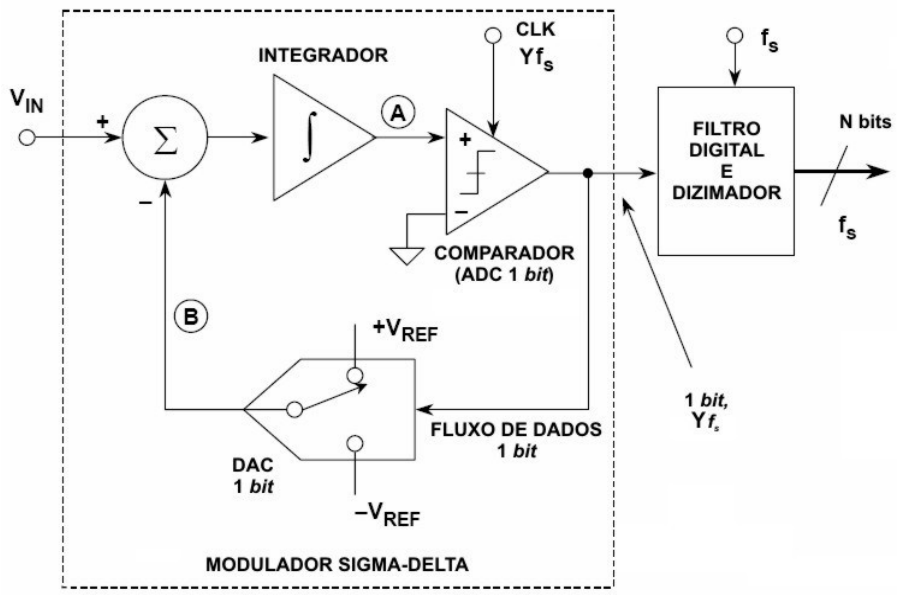

Figura 3.8: ADC Σ-Δ de primeira ordem.

O integrador no modulador é representado por um filtro analógico passa-baixa com função de transferência *H(t) = 1/f*. Essa função de transferência possui uma resposta em amplitude inversamente proporcional à frequência de entrada. Em essência, o filtro analógico apresenta um efeito passa-baixa no sinal e um efeito passa-alta no ruído de quantização. Desta forma, o filtro analógico realiza a função de "moldagem do ruído" no modulador Σ-Δ.

O diagrama de blocos para o modulador Σ-Δ de segunda ordem é mostrado na figura 3.9 (Kester, 2005). ADCs Σ-Δ de terceira ordem e de ordens superiores foram tidos inicialmente como potencialmente instáveis sob determinados valores de entrada. Análises recentes utilizando ganhos finitos (em vez de infinitos) no comparador mostraram que isso não é necessariamente verdadeiro, mas mesmo que instabilidades começassem a ocorrer, o processador digital de sinais no filtro digital e o dizimador podem ser projetados para reconhecerem a instabilidade incipiente e reagir de maneira a preveni-la.

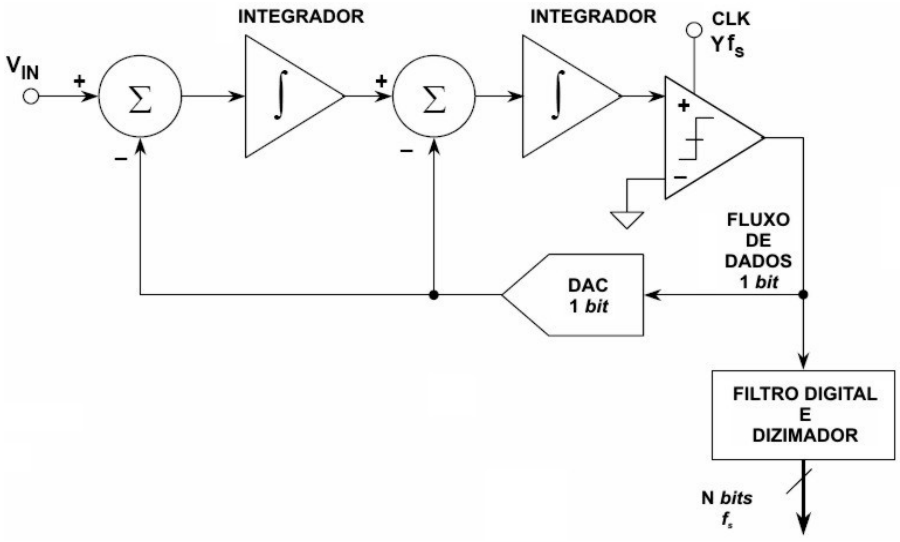

Figura 3.9: ADC Σ-Δ de segunda ordem.

O filtro digital é parte integrante de todos os ADCs Σ-Δ. Não há meios de removê-lo. O tempo de estabilização do filtro afeta certas aplicações, especialmente quando se utiliza ADC Σ-Δ em sistemas multiplexados.

4 COMPARADORES DE TENSÃO E CIRCUITOS ANALÓGICOS DINÂMICOS

Os comparadores representam o segundo componente mais largamente utilizado em circuitos eletrônicos, depois dos amplificadores operacionais (Gregorian, 1999). O comparador é um circuito que compara um sinal de entrada analógico com outro sinal analógico (ou referência) e produz na sua saída um sinal binário baseado em tal comparação (Allen *et al*, 2002). A aplicação mais importante de um comparador de tensão de alta velocidade é vista em um ADC. De fato, a velocidade de conversão está limitada ao tempo requerido para que ocorra a resposta (decisão) do comparador (Gregorian, 1999).

O comparador é formado por três estágios: pré-amplificador de entrada (*pré-amp*), circuito de decisão ou realimentação positiva e pós-amplificador ou *buffer* de saída (*pós-amp*). O estágio (ou estágios) de pré-amp amplifica o sinal de entrada para melhorar a sensibilidade do comparador (reforça o valor mínimo de entrada com o qual o comparador "toma" a decisão) e isola a entrada de ruídos de chaveamento advindos do estágio de realimentação positiva. O estágio de realimentação positiva determina qual dos sinais de entrada é maior. O pós-amp amplifica essa informação e produz um sinal digital (Baker, 2008).

Conforme mostrado na figura 4.1, um comparador pode ser visto como um ADC de 1 *bit*. Se o valor de entrada estiver acima de determinado limiar, a saída apresenta um determinado valor lógico, se estiver abaixo do limiar a saída apresenta outro valor lógico. Ao contrário de um amplificador diferencial, um comparador ordinário não utiliza realimentação negativa, e sua saída é um nível lógico que indica qual das duas entradas está em um potencial mais alto. Amplificadores operacionais não são projetados para serem usados como comparadores pois podem facilmente entrar em saturação e por apresentarem respostas lentas. No entanto, não se pode considerar a assertiva uma regra pois há casos nos quais pode ser desejável o uso de amplificadores operacionais como comparadores.

Os comparadores utilizados em ADCs necessitam de boa resolução, o que implica em

alto ganho. Isso pode levar a oscilações indesejáveis quando a entrada diferencial se aproxima de zero. Com o objetivo de se prevenir tal situação, comumente se adiciona histerese por meio de uma pequena quantidade de realimentação positiva. Na figura 4.1, verificam-se os efeitos da histerese na função de transferência. Muitos comparadores apresentam um ou dois milivolts de histerese para a ocorrência da ação de disparo (resposta) e para a prevenção de instabilidades na região de transição provocadas pela realimentação local. Note-se que a resolução do comparador não pode ser menor do que a histerese e, desta forma, valores elevados de histerese geralmente não são aplicáveis.

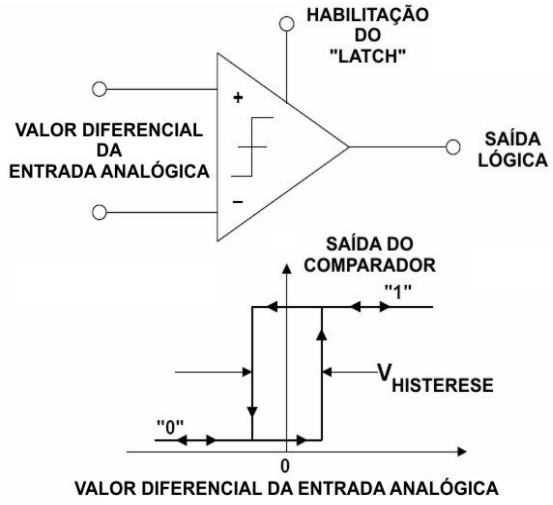

Figura 4.1: o comparador como ADC de 1 *bit*.

Além das características usuais que devem ser levadas em consideração no projeto de comparadores (resolução, velocidade, dissipação de potência, tensão de compensação, etc.), uma outra característica que pode resultar em consideráveis erros de conversão é a *metaestabilidade* (Kester, 2005). A metaestabilidade ocorre quando a diferença na entrada de um comparador é pequena, fazendo com que o circuito demande muito tempo para produzir uma saída lógica bem definida (Razavi, 1995). O problema do estado metaestável é mostrado na figura 4.2. Há três condições de tensão diferencial de entrada: (1) grande tensão diferencial de entrada, (2) pequena tensão diferencial de entrada e (3) nenhuma tensão diferencial (zero). A equação aproximada que descreve a tensão de saída, $V_o(t)$ é dada por:

$$V_o(t) = \Delta V_{IN} A e^{t/\tau} \qquad \text{(eq. 22)}$$

sendo que ΔV_{IN} corresponde à tensão diferencial de entrada no momento do acionamento do *latch*, A é o ganho do pré-amplificador, τ é a constante de tempo de regeneração do *latch* e t o tempo decorrido após a retenção da saída do comparador (*latched*).

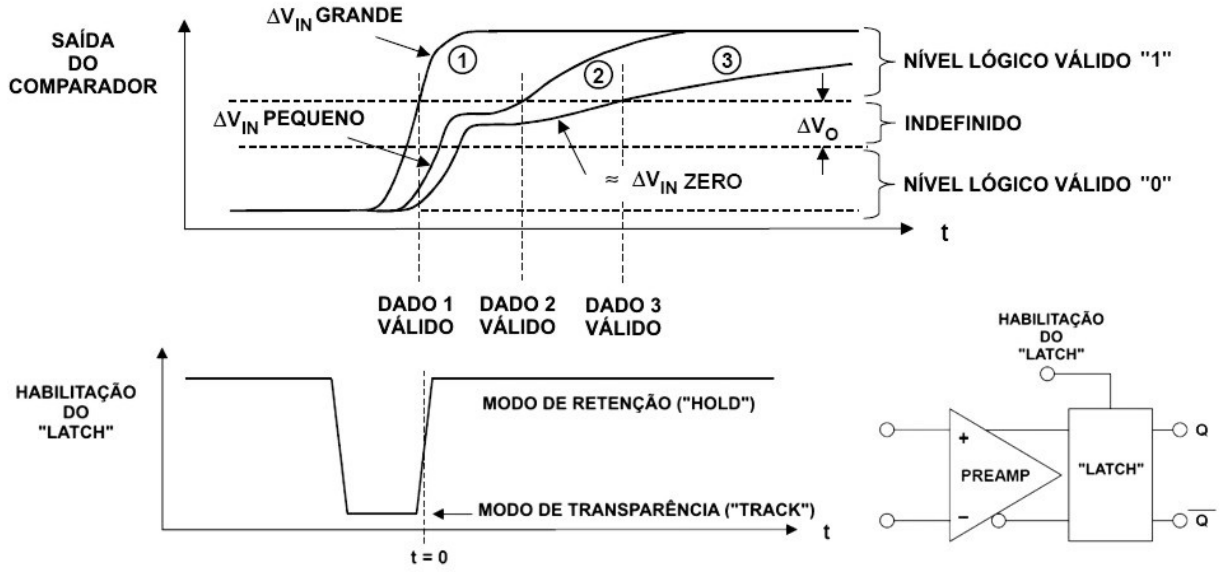

Figura 4.2: erros devido ao estado metaestável de um comparador.

Para pequenos valores de tensão diferencial de entrada a saída demora mais para atingir um determinado nível lógico válido. Se o dado de saída é lido enquanto estiver em uma região entre o nível lógico “1” e o nível lógico “0” haverá ocorrência de erro na conversão. Se a tensão diferencial de entrada for exatamente zero o tempo necessário para ocorrer um nível lógico válido pode se tornar bastante longo (teoricamente infinito). No entanto, histerese e ruído tornam tal condição improvável.

Do ponto de vista de projeto a metaestabilidade do comparador pode ser minimizada por meio do aumento no ganho, A, da diminuição da constante de tempo de regeneração, τ, ou permitindo um tempo suficiente para que a saída do comparador atinja o nível lógico válido. Do ponto de vista de operação, o efeito da metaestabilidade pode ser percebida na *taxa de erros de bit* (BER) (Kester, 2005).

4.1 ELEMENTOS BÁSICOS DO COMPARADOR

Os circuitos de polarização (*bias circuits*), espelhos de corrente (*current mirrors*), amplificadores de estágio único, seguidores de fonte, bem como os estágios diferenciais são tipicamente combinados para a criação de circuitos com funções complexas. O amplificador diferencial e o comparador são exemplos de como circuitos simples são combinados para a formação de circuitos com funções complexas (Gregorian, 1999).

4.2 ESPELHOS DE CORRENTE

As fontes de corrente MOS são similares às fontes bipolares, sendo que os espelhos de corrente funcionam sob o princípio de que dispositivos idênticos com tensões iguais entre porta e fonte (V_{GS}) e dreno e fonte (V_{DS}) apresentam correntes de dreno iguais. Na figura 4.3, assumindo-se que M1 e M2 possuem a mesma largura (W) e comprimento (L) e que ambos estão em saturação, as respectivas correntes de dreno, I_1 e I_2, são determinadas, de maneira geral, pelos valores individuais de V_{GS}. Observando-se então, que $V_{GS1} = V_{DS1} = V_{GS2}$, o que se espera é que ambos os MOSFETs tenham as mesmas correntes de dreno (Gregorian, 1999). Um circuito como o da figura 4.3 é chamado de "espelho de corrente" porque a corrente de dreno I_2 é um reflexo da corrente I_1, quase como uma imagem de espelho (Malvino, 1997).

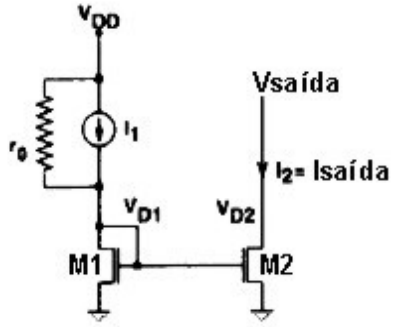

Figura 4.3: espelho de corrente básico.

4.3 CIRCUITOS DE POLARIZAÇÃO

Um elemento ideal de polarização de corrente ou tensão é independente das tensões (CC) de alimentação ($V_{DD} > 0$ e $V_{SS} \leq 0$) e da temperatura. Divisores de tensão puramente resistivos são raramente usados na tecnologia MOS. A utilização de resistores para se obter a corrente de polarização pode resultar em correntes que são bastante dependentes das fontes de alimentação (V_{DD} e V_{SS}) e da temperatura. De fato, as correntes de polarização aumentam rapidamente com o aumento no valor da fonte de alimentação (Gregorian, 1999; Baker, 2008). É desejável portanto, que a fonte de corrente de polarização não varie conforme as mudanças em V_{DD} ou V_{SS}. Uma alternativa ao resistor é a sua substituição por um MOSFET, como mostrado na figura 4.4 (Baker, 2008). Se M3 for visto como um resistor, a corrente de M1 será "espelhada" por M2. Se, no entanto, M1 for visto como um resistor, a corrente de M3 será "espelhada" por M4. Lembrando que $V_{DD} = V_{SG3} + V_{GS1}$, I_{REF} pode ser obtida de acordo com as relações entre largura e comprimento dos transistores, observando-se a condição de saturação dos mesmos.

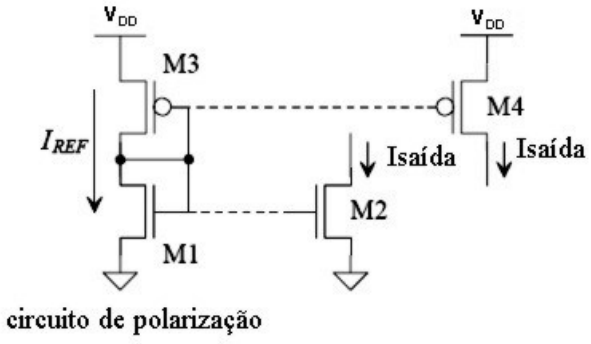

Figura 4.4: circuito de polarização somente com MOSFET.

4.4 AMPLIFICADOR DIFERENCIAL

Um sinal diferencial é definido como aquele medido entre dois nós que apresentam excursões de sinal iguais e opostas em torno de um potencial fixo. O potencial "central" na

sinalização diferencial é denominado nível em modo comum (CM) (Razavi, 2001). O amplificador diferencial (*amp-dif*) é usado na entrada de um amplificador para permitir que as tensões de entrada não afetem a polarização dos estágios de ganho. A figura 4.5 mostra um amp-dif com um espelho de corrente como carga (Baker, 2008). Para melhorar o desempenho e aumentar o ganho do amplificador, uma fonte de corrente pode ser usada como carga, ou seja, uma carga resistiva é substituída pela impedância de saída (pequenos sinais) da fonte de corrente (Gregorian, 1999).

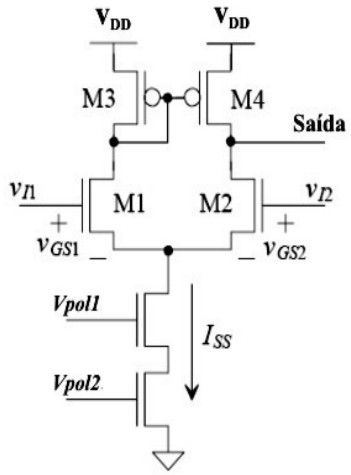

Figura 4.5: amplificador diferencial com espelho de corrente como carga.

Na figura acima, um desequilíbrio entre as correntes de dreno de M1 e M2 faz com que a saída do amp-dif excursione tanto na direção de V_{DD} quanto na direção de V_{SS} (Baker, 2008). As tensões v_{GS1} e v_{GS2} correspondem, respectivamente, às tensões entre porta e fonte dos transistores M1 e M2 dispostos na configuração denominada par diferencial (*par-dif*). A diferença entre as tensões de entrada, v_{I1} e v_{I2}, é $v_{GS1} - v_{GS2}$. As tensões V_{pol1} e V_{pol2} são utilizadas na polarização do *par-dif*.

4.5 CIRCUITO DE POLARIZAÇÃO MULTIPLICADOR BETA

Conforme mostrado na figura 4.6, modificando a posição do resistor no lado do dreno para o lado da fonte, adicionando-se um diodo (M1) como elemento de corrente a ser espelhada pelo dispositivo alimentado por V_{poln}, em conjunto com um fator multiplicador a fim de diminuir o valor de V_{GS} necessário à condução de I_{REF} (geralmente atribuindo a largura de M2 maior do que a de M1), obtém-se um circuito denominado *circuito de referência multiplicador beta*. Assim, utilizando-se um valor maior de β em M2, ou seja, multiplicando o β de M1 por um fator Z de maneira que $\beta_2 = Z.\beta_1$, tem-se que $V_{GS1} = V_{GS2} + RI_{REF}$. Lembrando que $\beta = \mu_n C'_{ox} W/L$, sendo μ_n a mobilidade de elétrons, C'_{ox} a capacitância do óxido por área, W a largura e L o comprimento do transistor. Se L_2 for igual a L_1, a relação entre β_2 e β_1 pode ser satisfeita com $W_2 = Z.W_1$. As tensões V_{polp} e V_{poln} correspondem às tensões de polarização utilizadas nos circuitos alimentados pelo circuito multiplicador beta, conforme visto na figura 4.8.

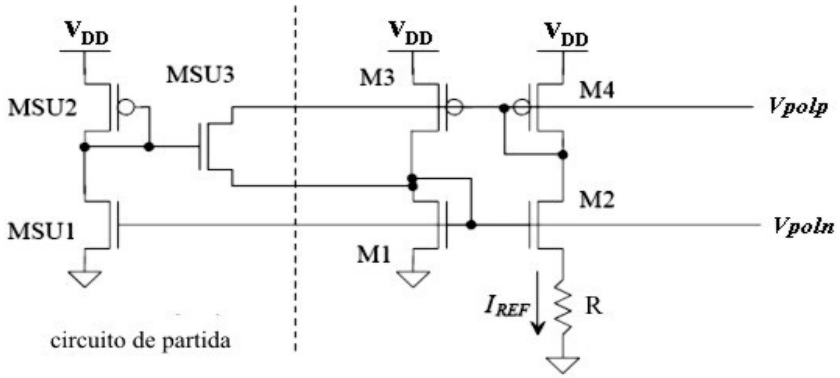

Figura 4.6: multiplicador beta como elemento de polarização.

Em circuitos com autopolarização há dois pontos de operação possíveis: o primeiro corresponde ao funcionamento usual com polarização e o segundo, não desejável, ocorre quando não há corrente fluindo no circuito. O estado indesejável ocorre quando as portas de

M1 e M2 estão em terra enquanto as portas de M3 e M4 estão em V_{DD}. Quando nesse estado, a porta de MSU1 está em terra e consequentemente este se encontra desligado. A porta de MSU2 se encontra em algum ponto entre V_{DD} e $V_{DD} - V_{th}$ (tensão de limiar do PMOS). MSU3, que se comporta como uma chave NMOS, conduz produzindo um vazamento de corrente das portas de M3 e M4 para as portas de M1 e M2. Isso faz com que a corrente atinja o estado desejado e MSU3 se desliga. Durante a operação normal, o circuito de partida não deve afetar a operação do multiplicador Beta e a corrente em MSU3 deve ser zero (ou bastante pequena).

4.6 CIRCUITO DE DECISÃO

O circuito de decisão é o coração do comparador, devendo ser capaz de discriminar sinais com diferenças muito pequenas (milivolts). O circuito mostrado na figura 4.7 utiliza a realimentação positiva da conexão cruzada entre os transistores M6 e M7 para aumentar o ganho do elemento de decisão. Assumindo-se inicialmente que i_{op} é muito maior do que i_{om}, M5 e M7 se encontram "ligados" e M6 e M8 "desligados". Quando i_{om} cresce e i_{op} decresce, começa a ocorrer a mudança dos estados de M5 e M7 para "desligados" e M6 e M8 para "ligados".

Se os transistores tiverem o mesmo tamanho, a mudança nos estados ocorre quando i_{op} e i_{om} forem iguais. Valores diferentes no tamanho dos transistores fazem com que o comparador apresente histerese.

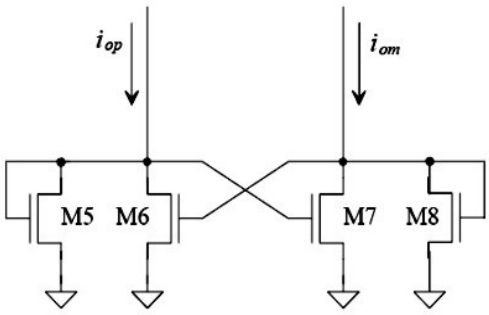

Figura 4.7: circuito de decisão com realimentação positiva.

4.7 COMPARADOR DE PROPÓSITO GERAL

A figura 4.8 mostra a conformação de um comparador de propósito geral, linha-a-linha, (*rail-to-rail*) com entrada em modo comum. O pré-amp consta de um amplificador diferencial com elementos PMOS e NMOS, que alimentam o circuito de decisão, para que seja mantida a característica linha-a-linha do comparador. O pós-amp (*buffer* de saída) é um amplificador diferencial PMOS cujo propósito é converter a saída do circuito de decisão em um sinal lógico (0 ou V_{DD}). Note-se que as correntes do amp-dif PMOS (entrada do comparador) são adicionadas às correntes (saída) do NMOS para assegurar que sempre haja fornecimento de corrente ao circuito de decisão (Baker, 2008).

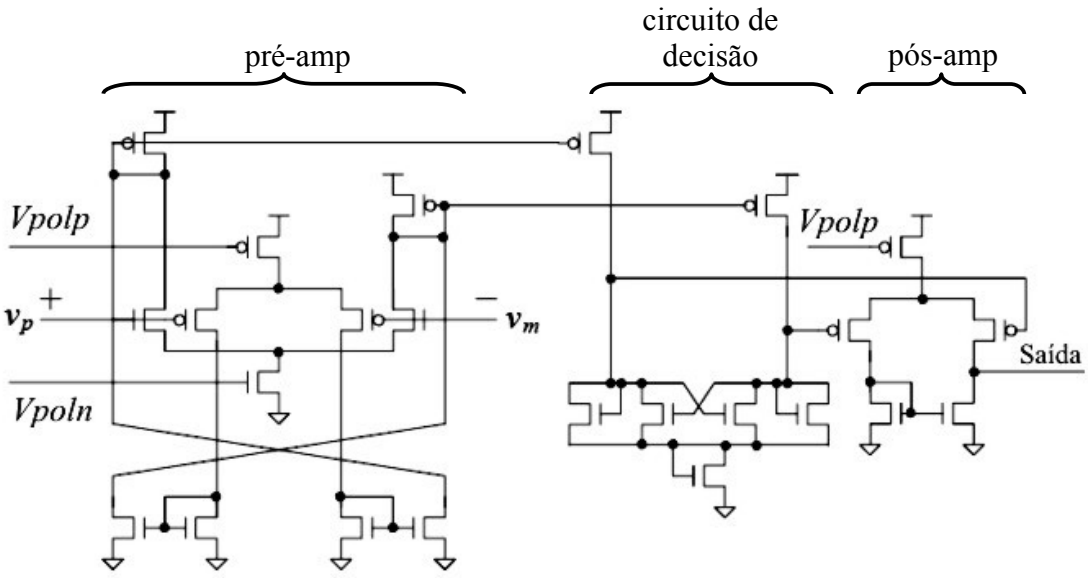

Figura 4.8: comparador de propósito geral, linha-a-linha, com entrada em modo comum.

O projeto do comparador, ao contrário de um projeto puramente digital (utilizando biblioteca de células digitais) não pode ser realizado utilizando componentes de comprimento mínimo por razões de ganho, tensão de compensação, dentre outros fatores (Gregorian, 1999).

4.8 CIRCUITOS ANALÓGICOS DINÂMICOS

Os circuitos analógicos dinâmicos exploram o fato de que a informação pode ser armazenada em um capacitor ou na capacitância de porta de um dispositivo MOSFET por um período de tempo. Um componente fundamental em qualquer circuito dinâmico (digital ou analógico) é a chave. Do ponto de vista da implementação digital, um transistor NMOS é uma chave quase perfeita quando da passagem do nível lógico "0", o que não ocorre quando da passagem do nível lógico "1" (valor máximo correspondendo a $V_{DD} - V_{th}$). Com o transistor PMOS ocorre o inverso (nível lógico "0" não menos do que V_{th}).

Quando um NMOS ou PMOS são usados como chaves "imperfeitas", são comumente denominados transistores de passagem (*pass transistor*). A combinação de um transistor NMOS em paralelo com um transistor PMOS configura o que se chama de porta de transmissão (*transmission gate* - TG) ou porta de passagem (*pass gate*) (Weste *et al*, 2005).

Conforme verificado em Baker (Baker, 2008), enquanto as chaves MOS oferecem benefícios substanciais, estas não estão livres de elementos de degradação. Dois efeitos não ideais tipicamente associados a essas chaves podem, em última análise, limitar o seu uso em algumas aplicações, particularmente nos conversores de dados. Os dois efeitos são conhecidos como injeção de carga (*charge injection*) e alimentação por meio do sinal de *clock* (*clock feedthrough*). A figura 4.9 ilustra o processo de injeção de carga no qual a carga no óxido de porta do MOSFET (resultante da inversão do canal quando no estado "ligado") é injetada no capacitor e em $V_{entrada}$ quando a chave muda do estado "ligado" para "desligado".

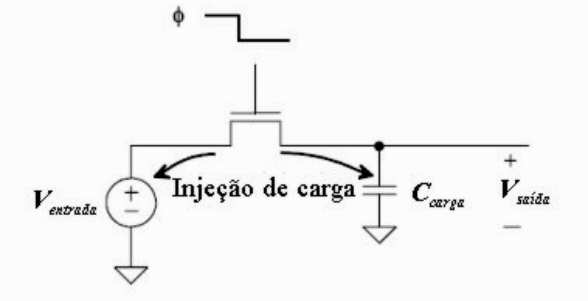

Figura 4.9: mecanismo básico de injeção de carga.

A carga injetada em C_{carga} resulta em um mudança na tensão armazenada, que não é linear em relação a $V_{entrada}$ devido à tensão de limiar do MOSFET. Nos sistemas de amostragem de dados, a injeção de carga resulta em erros de não linearidade.

Considerando-se o circuito mostrado na figura 4.10, quando o sinal de *clock*, Φ, aplicado à porta do transistor muda para o nível alto, o sinal de *clock* alimenta as capacitâncias de porta/dreno e porta/fonte. Nesse caso, o valor final de $V_{saída}$ não é afetado por essas capacitâncias pois quando a chave está "ligada" o sinal de entrada, $V_{entrada}$, está conectado a C_{carga} por meio da referida chave. No entanto, quando o sinal de *clock* muda para o nível baixo, ou seja, quando o NMOS "desliga", uma porção do sinal Φ surge em C_{carga} devido à existência de um divisor de tensão capacitivo entre as capacitâncias de porta/dreno (fonte) e C_{carga}. C_{ox} corresponde à capacitância de sobreposição entre o óxido de porta e os eletrodos de fonte e dreno.

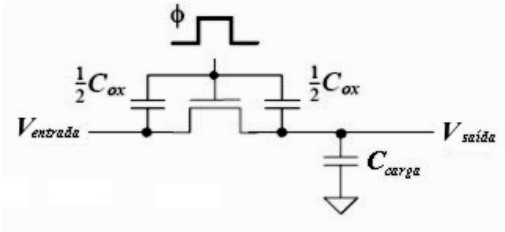

Figura 4.10: alimentação indireta do sinal de *clock*.

Um dos métodos mais utilizados para a redução dos efeitos da injeção de carga e alimentação indireta por meio de capacitâncias é a aplicação de uma chave fictícia (*dummy*) no circuito, como mostrado na figura 4.11. Nesse caso, uma chave M2 com a fonte e dreno em curto-circuito é colocada em série com a chave principal M1. O sinal de *clock* que alimenta a chave fictícia é complementar ao sinal que alimenta a chave M1.

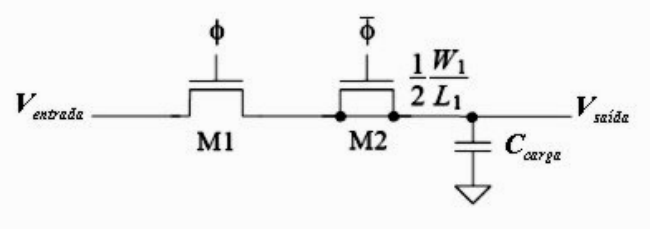

Figura 4.11: circuito com chave fictícia utilizada para minimizar a injeção de carga.

Quando M1 “desliga”, metade da carga do canal é injetada na chave fictícia, explicando o porquê do tamanho de M2 ser a metade do tamanho de M1. Apesar de M2 estar em curto-circuito, um canal ainda pode ser induzido por meio da aplicação de uma tensão de porta. Assim, a carga injetada por M1 é essencialmente igualada com a carga induzida por M2, cancelando a injeção de carga total. Quando M2 “desliga”, metade de sua carga é injetada em ambas as direções. No entanto, devido ao fato da fonte e dreno estarem em curto-circuito e de M1 estar “ligada”, toda a carga de M2 será injetada na fonte, que também está carregando C_{carga}. Desta forma, a injeção de carga de M2 não irá afetar o valor da tensão em C_{carga}.

Um outro método de impedimento da injeção de carga e da alimentação indireta do sinal de *clock* é a substituição da chave por uma TG. Isso resulta em pequenas mudanças em $V_{saída}$ porque os sinais complementares utilizados irão atuar de maneira a se cancelarem. No entanto, essa abordagem requer controle preciso nos sinais complementares de *clock*, ou seja, os sinais devem “chavear” exatamente ao mesmo tempo.

5 METODOLOGIA

A abordagem convencional para o projeto de circuitos frequentemente envolve muita interação ao nível da montagem funcional do circuito. Devido à complexidade de muitos projetos em VLSI e o custo dos recursos associados ao projeto e fabricação do CI, a abordagem convencional se restringe a poucas aplicações (Geiger *et al*, 1990).

5.1 MÉTODOS DE PROJETO

A análise e projeto de circuitos integrados com frequência necessita de abordagens em vários níveis de abstração (Razavi, 2001). De acordo com Geiger (Geiger *et al*, 1990), duas abordagens no projeto de circuitos integrados são "filosoficamente" identificáveis. Na primeira abordagem, "de baixo para cima" (*bottom-up*), o projetista começa no nível do transistor ou porta e, a partir daí cria subcircuitos com complexidade crescente, interconectados posteriormente para se conseguir a funcionalidade requerida. Na segunda, "de cima para baixo" (*top-down*), o projetista repetidamente decompõe as especificações ao nível do sistema em grupos e subgrupos de tarefas mais simples.

Em uma abordagem *top-down* genérica, a arquitetura do *chip* é definida como um diagrama de blocos que são simulados e otimizados por meio de um sistema simulador. Da simulação em alto nível são derivados os requisitos para os blocos individuais do circuito. Os circuitos são então projetados individualmente para atenderem às especificações. Finalmente, o *chip* completo emerge e é verificado em relação aos requisitos originais (Kundert, 2010).

5.2 PROPOSTA

A metodologia *top-down*, em um nível amplo, foi empregada no projeto do ADC SAR de 12 *bits* para permitir, inicialmente, um entendimento geral da dinâmica do sistema, bem

como a verificação da interação entre os diferentes blocos, incluindo-se a relação entre a porção digital e a porção analógica. Assim, primeiramente foram utilizadas as ferramentas Stateflow® e Simscape™ do Simulink® (Mathworks Inc.) para criação de um modelo com características menos restritivas, incluindo-se a utilização de componentes ideais, com o foco no comportamento geral do sistema. Posteriormente, vislumbrando o reuso em sistemas de maior complexidade, como um *SoC*, criou-se um modelo em SystemC™/SystemC-AMS com características de projeto com um grau de refinamento inerentemente maior. Finalmente foi desenvolvido o CI, utilizando a ferramenta DFII da Cadence (Cadence Design Systems, Inc.). A figura 5.1 mostra o fluxo de desenvolvimento adotado.

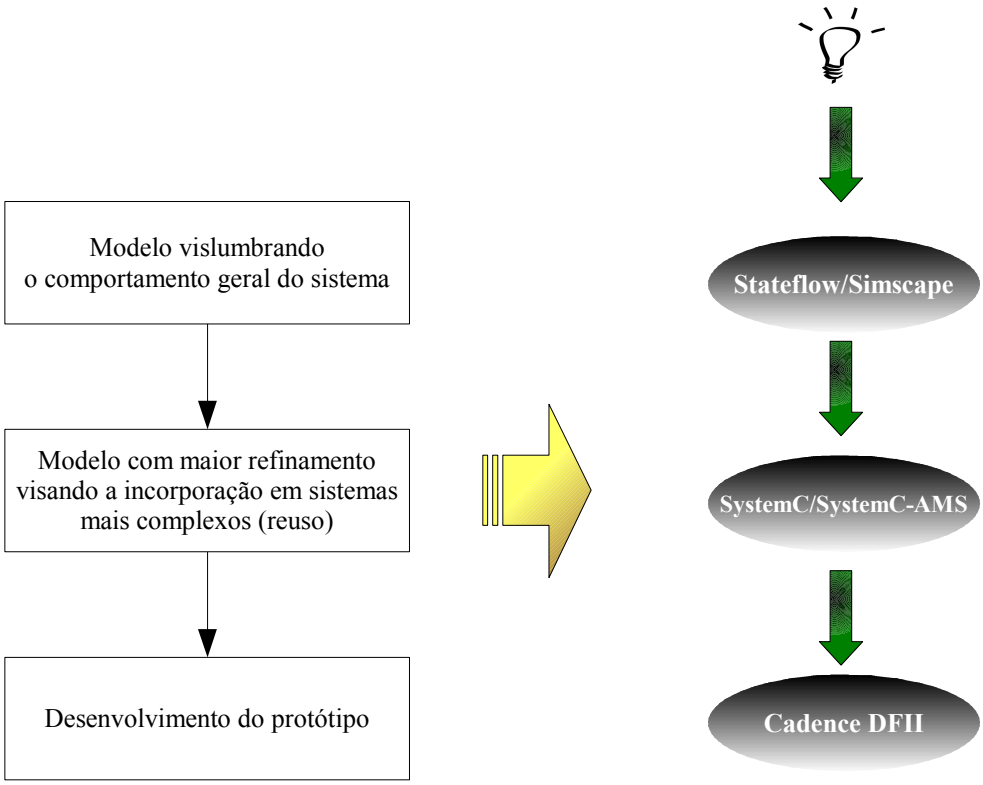

Figura 5.1: fluxo de desenvolvimento do ADC SAR de 12 *bits*.

5.2.1 SIMULINK (STATEFLOW/SIMSCAPE)

O Simscape é uma extensão do Simulink com ferramentas para modelagem e simulação de sistemas físicos, tais como o elétrico e componentes eletromagnéticos. Diferentemente de

outros blocos do Simulink, que representam operações matemáticas ou atuam sobre sinais, os blocos do Simscape representam componentes com os quais se constrói um modelo de um sistema de maneira semelhante à montagem do sistema físico.

Os modelos do Simscape empregam a aproximação de uma rede física para modelar os componentes construtivos (blocos) correspondentes a elementos físicos como amplificadores operacionais, motores, etc. Do modelo construído, que se assemelha a um esquemático, o Simscape automaticamente constrói equações que caracterizam o comportamento do sistema. Tais equações são integradas ao restante do modelo do Simulink.

O Stateflow é uma ferramenta de desenho gráfico que funciona com o Simulink para modelar e simular sistemas reativos. Sistemas reativos comutam de um modo de operação para outro em resposta a eventos e condições. Tais sistemas são comumente usados para modelar a lógica de controle dinâmico de determinado componente físico como um motor, bomba, etc. Sistemas reativos podem ser modelados como máquinas de estados finitos (FSM). O Stateflow permite a criação de quadros (*charts*) nos quais uma série de transições direcionam um fluxo lógico de um estado ao outro (MathWorks, 2010).

Na criação do modelo do ADC, a porção de controle digital foi implementada utilizando-se FSM no Stateflow e a porção analógica utilizando-se os elementos disponíveis no Simscape.

Para corroborar os resultados obtidos no modelo Simulink, procedeu-se à criação do modelo em SystemC/SystemC-AMS.

5.2.2 LINGUAGEM SYSTEMC/SYSTEMC-AMS

A linguagem SystemC, oriunda da confluência de quatro correntes de ideias (Synopsys, Frontier Design, IMEC e OSCI *Language Working Group*), é baseada na linguagem C++, estendendo suas capacidades por meio de uma biblioteca de classes e de um núcleo de simulação (*kernel*) destinados ao desenvolvimento de sistemas digitais. A extensão permite

uma hierarquia estruturada, bem como suporte à simulação. Oferece ganhos substanciais de produtividade por permitir o desenvolvimento de componentes, tanto de *hardware* quanto de *software*, em alto nível de abstração. O alto nível de abstração permite ao projetista um melhor entendimento do funcionamento do sistema como um todo, bem como o reuso, aumentando-se assim, os ganhos de produtividade (Bhasker, 2002; Grötker, 2002; Müller, 2003; Black, 2004).

Estritamente falando, SystemC não é uma linguagem, mas uma biblioteca de classes dentro de uma linguagem bem estabelecida, C++. Uma característica importante do SystemC é a sua habilidade em prover níveis mais altos de abstração para todos os componentes de um projeto. O gerenciamento da abstração é a arma mais poderosa no combate à complexidade (Black, 2004).

A crescente redução no tempo para se lançar novos dispositivos eletrônicos (*time to market*), assim como a crescente complexidade dos mesmos, dentre outros fatores, propiciou o surgimento do SystemC. Sua utilização, por exemplo, pode ajudar a acelerar a verificação dos elementos funcionais do sistema na medida em que permite o desenvolvimento sem que haja a preocupação com os detalhes finais da implementação do dispositivo real (Bhasker, 2002).

Um dos objetivos fundamentais do SystemC é permitir a modelagem ao nível do sistema (*system-level modeling*), ou seja, a modelagem de sistemas em um nível de abstração acima do RTL, incluindo sistemas que podem ser implementados em *software*, *hardware* ou alguma combinação dos dois. Naturalmente que a modelagem em RTL também pode ser realizada em SystemC e, em adição a isso, pode-se desenvolver modelos acima do RTL e, posteriormente, refinando-os ao RTL dentro de uma linguagem e ambiente únicos (Grötker, 2002).

Conforme (OSCI SystemC Lang Ref, 2003), a figura 5.2 mostra a arquitetura do SystemC. Os blocos sombreados fazem parte do núcleo do padrão. Um sistema SystemC consiste em um conjunto de um ou mais módulos. Os módulos permitem a descrição das estruturas e tipicamente contém os processos, portas, dados internos, canais e instâncias de

outros módulos. As portas são objetos pelos quais o módulo se comunica com outros módulos. Os dados internos e os canais permitem a comunicação entre os processos.

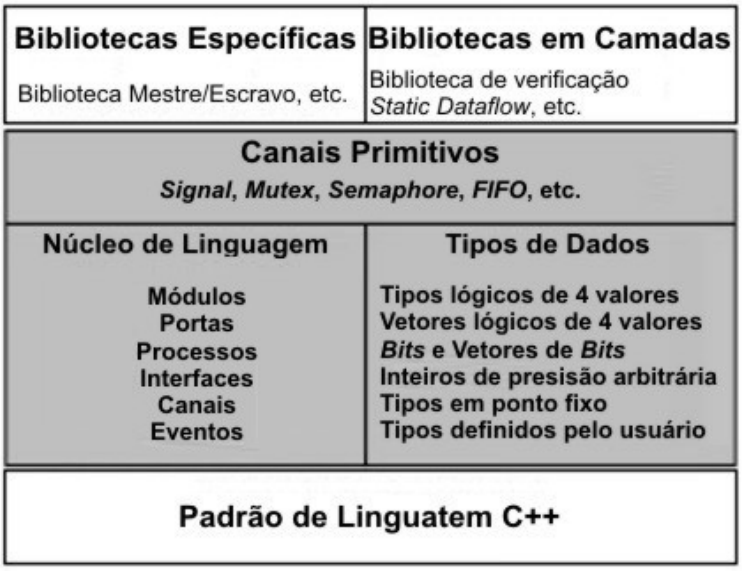

Figura 5.2: arquitetura do SystemC.

Os eventos são os objetos básicos de sincronização. São usados para sincronizar processos e implementam o comportamento de bloqueio entre canais. Os processos são iniciados por gatilho ou têm a execução induzida por meio da sensibilidade a eventos.

Os módulos permitem ao projetista “quebrar” sistemas complexos em pedaços menores e mais manejáveis, bem como esconder algoritmos e representação de dados internos de outros módulos. Os processos correspondem à unidade básica de funcionalidade fornecendo o mecanismo para a simulação de comportamentos paralelos (Grötker, 2002).

O padrão SystemC-AMS define uma biblioteca de classes C++, baseada no SystemC, para o desenvolvimento de sistemas que contém funcionalidades analógicas ou de sinais mistos (*analog/mixed signal*). Assim, o padrão é construído sobre o IEEE 1666-2005 (*SystemC Language Reference Manual*) introduzindo novas semânticas de execução e metodologias de modelagem ao nível do sistema para projeto e verificação de sistemas de sinais mistos (Müller, 2003; OSCI SystemC AMS Lang Ref, 2010; OSCI SystemC User's Guide, 2010).

As extensões SystemC-AMS definem os formalismos de modelagem necessários ao suporte da modelagem comportamental do AMS em diferentes níveis de abstração. Tais formalismos são implementados através de diferentes modelos de computação, MoC (*Model of Computation*), que é um conjunto de regras que definem o comportamento e a interação entre os módulos AMS primitivos instanciados dentro de um determinado módulo, a saber: *Timed Data Flow* (TDF), *Linear Signal Flow* (LSF) e *Electrical Linear Networks* (ELN). As semânticas de execução baseadas no TDF introduzem a modelagem e simulação de tempo discreto sem a carga extra da programação dinâmica imposta pelo núcleo de eventos discretos do SystemC. O formalismo do LSF permite a modelagem do comportamento de tempo contínuo oferecendo um conjunto consistente de módulos primitivos tais como de adição, multiplicação, integração ou atraso. Um modelo LSF define um sistema de equações lineares. No ELN, a modelagem de redes com componentes elétricos é suportada pelo instanciamento de elementos lineares primitivos tais como resistores ou capacitores, usados na descrição de relações de tempo contínuo entre tensões e correntes. Um conjunto restrito de elementos lineares primitivos e chaves está disponível para se modelar o comportamento elétrico (OSCI SystemC User's Guide, 2010).

Bhasker (2002) apresenta de maneira didática inúmeros exemplos de aplicações do SystemC, mostrando inicialmente os componentes de determinado circuito seguidamente da respectiva implementação do código. Essa abordagem auxilia o projetista na familiarização dos procedimentos de desenvolvimento utilizando a terminologia do SystemC.

O modelo criado em Simulink (Stateflow/Simscape) serviu de parâmetro para a criação do modelo em SystemC e SystemC-AMS. De fato, a maneira como os diferentes componentes do sistema se inter-relacionam foi percebida a partir do primeiro modelo criado e, a partir desse, foi criado o modelo em SystemC/SystemC-AMS. O que o SystemC oferece ao desenvolvedor como os módulos (SC_MODULE), processos (SC_METHOD e SC_THREAD), portas, sinais, formas de onda de saída (Bhasker, 2002), etc. foi plenamente

utilizado na criação do modelo. Os módulos foram aplicados como os blocos básicos de construção para o particionamento da implementação.

Implementações em SystemC podem ser vistas em Müller (Müller, 2003) utilizando-se, por exemplo, programas como o *Synopsys CoCentric System Studio*^{TM}. Neste projeto, no entanto, optou-se pela utilização de programas largamente distribuídos e de forma gratuita como o compilador *Eclipse IDE para desenvolvedores em C/C++* e o visualizador de formas de onda *GTKWave Analyzer*.

As extensões SystemC-AMS permitiram o desenvolvimento dos componentes analógicos do sistema de sinais mistos com uma integração natural ao SystemC, visto que são construídas em consonância com o padrão IEEE 1666-2005.

Terminada a criação do modelo e executadas as devidas simulações, prosseguiu-se à etapa final do projeto, o desenvolvimento do CI.

5.2.3 CADENCE DESIGN FRAMEWORK II

O ambiente DFII da Cadence integra uma variedade de atividades de desenvolvimento, incluindo a criação de esquemáticos (Composer), criação de layout (Virtuoso e Virtuoso-XL), verificação das regras de desenho no layout (Diva®) e comparação entre layout e esquemático (Diva). Esses são todos programas individuais que desempenham uma parte do processo de desenvolvimento do CI, mas que são acessados por meio do DFII. Note-se que muitos desses programas foram desenvolvidos por diferentes empresas, adquiridas pela Cadence, e aglutinados no DFII após a aquisição. Desta maneira, alguns se integram melhor do que outros (Brunvand, 2010).

Nesta etapa final do projeto foram aplicados os conhecimentos adquiridos quando do desenvolvimento dos modelos como os blocos constituintes, os sinais de entrada, os sinais de saída, as etapas de controle, temporização, etc. Essa etapa foi completada com o auxílio do Design Framework II (DFII) da Cadence.

6 IMPLEMENTAÇÃO

Conforme a metodologia empregada, o desenvolvimento do projeto se iniciou com a criação de um modelo do conversor em Simulink, posteriormente a criação de um modelo em SystemC/SystemC-AMS e por fim, o projeto do *layout* com o auxílio das ferramentas Cadence Design Framework II (DFII) e de acordo com as especificações do processo CMOS 0.35 μm da X-FAB (X-FAB Semiconductor Foundries).

6.1 MODELO STATEFLOW/SIMSCAPE

Os elementos de controle digital do conversor, o SAR propriamente dito, foram implementados como uma máquina de estado finitos (FSM) no Stateflow. Conforme verificado na figura 6.1, dois eventos de entrada, nominados *Clock* e *Clock_falledge*, alimentam o SAR sendo o primeiro com gatilho nas transições de subida e descida (*Trigger: Either*) e o segundo com gatilho na transição de descida (*Trigger: Falling*). Os dois sinais foram combinados em um "multiplexador" (bloco *Mux* disponível no Simulink) que combina as entradas em um vetor único de saída. As outras entradas necessárias ao funcionamento do sistema foram nominadas *Comp_in* e *Startup*, respondendo pelo sinal de entrada do comparador e pelo comando de início de conversão, respectivamente. As saídas *S1* a *S13* correspondem aos sinais de controle das chaves ligadas ao arranjo de capacitores, a saída *S0* corresponde ao sinal de controle da chave responsável pela fase inicial de cancelamento da tensão de compensação do comparador e do carregamento dos capacitores e a saída *S14* corresponde ao sinal de controle da chave seletora da tensão de entrada, v_{IN}, e da tensão de referência, V_{REF}. A saída *EOC* corresponde ao sinal indicativo de uma conversão em curso ou de término de conversão. Outros elementos de "escopo local" também são componentes do SAR e serão vistos posteriormente no detalhamento da FSM propriamente dita.

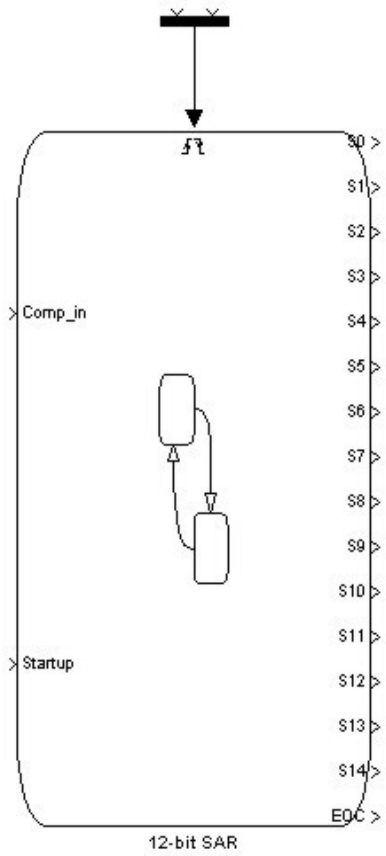

Figura 6.1: conformação do controle digital – SAR.

Na terminologia do Stateflow, estados que contém outros estados são denominados *superestados*. Um estado é considerado *subestado* quando este estiver contido em outro. Os estados possuem uma decomposição que dita o tipo de subestado englobado. A decomposição pode ser *exclusiva* (*OR*) ou *paralela* (*AND*). A decomposição exclusiva de um superestado é indicada por linhas sólidas em seus subestados e é usada para descrever modos do sistema que são mutuamente exclusivos. Quando um estado possui decomposição exclusiva, somente um subestado pode estar ativo por vez. A decomposição paralela de um superestado é indicada quando seus subestados apresentam linhas pontilhadas. Esse tipo de representação é utilizada quando estados de mesmo nível hierárquico estão ativos ao mesmo tempo. A atividade entre estados paralelos é essencialmente independente.

O rótulo (*label*) do estado, localizado no canto esquerdo superior do retângulo

correspondente permite a introdução de diretrizes opcionais para execução de ações: *entry*, *during*, *exit*, *on* e *bind*. A ação precedida pelo prefixo *entry* ocorre no momento em que o estado se torna ativo. O prefixo *during* determina a execução de ações quando o estado já está ativo ou quando ocorre algum evento. Em *exit* a execução da ação ocorre quando o estado se torna inativo. Em *on*, quando ocorre um evento, a ação associada ao mesmo é executada. *Bind* determina que somente o estado pai ou seus estados filhos podem realizar a ação vinculada à diretriz.

As transições são linhas terminadas por setas que ligam um objeto (um estado, por exemplo) ao outro. Uma *condição* associada à transição especifica uma expressão booleana que, quando verdadeira, valida a sua ocorrência. Uma *transição padrão* especifica qual estado exclusivo será executado quando houver ambiguidade entre dois ou mais estados (exclusivos) adjacentes. A transição padrão possui somente o objeto de destino, ou seja, não há objeto de origem.

O controle digital do conversor é composto por um superestado, nominado *SAR*, que engloba os subestados correspondentes às seguintes operações:

1. *Startup*: início de conversão propriamente dito;
2. *Offset_cancel*: cancelamento da tensão de compensação do comparador;
3. *Sample:* amostragem;
4. *Hold*: retenção;
5. *Shift_register*: componentes dos registradores de deslocamento;
6. *Storage_register*: componentes de armazenamento de informação do *bit;* e
7. *End_of_conv*: fim de conversão.

A figura 6.2 contempla o detalhamento dos elementos do *SAR*, conforme descrição acima.

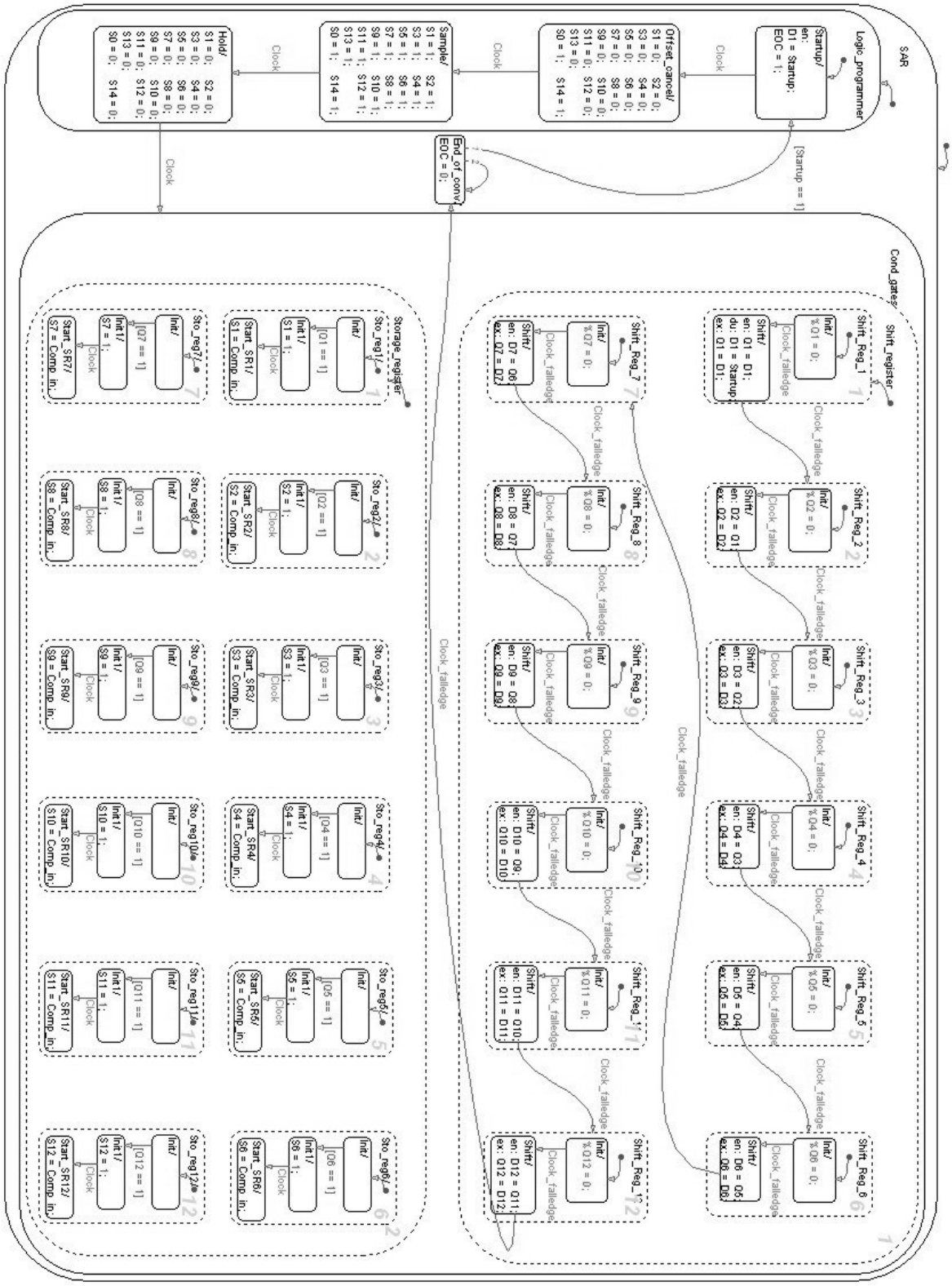

Figura 6.2: visualização da FSM correspondente ao controle digital SAR.

O comando *Startup* dispara o processo de conversão, levando a saída *EOC* ao nível lógico "1" e definindo a entrada do primeiro registrador de deslocamento com o nível lógico do correspondente sinal de *Startup* (nível lógico "1"). Uma transição do sinal de *Clock* produz a mudança do estado *Startup* para o próximo estado correspondente ao cancelamento da tensão de compensação do comparador (*Offset_cancel*). As saídas *S1* a *S13* assumem o valor lógico "0" fazendo com que as placas inferiores dos capacitores sejam aterradas. As saídas *S0* e *S14* assumem o valor lógico "1" de maneira que o valor da tensão de compensação do comparador seja carregado nas placas superiores dos capacitores (chave *S0*) e o valor da tensão de entrada v_{IN} esteja disponível no barramento ligado às placas inferiores dos capacitores (*S14*).

Na próxima transição do sinal de *Clock* ocorre a mudança para o estado correspondente à amostragem (*Sample*) no qual as saídas *S1* a *S13* assumem o valor lógico "1" permitindo que o valor de v_{IN} seja carregado nas placas inferiores dos capacitores. A transição seguinte do *Clock* produz a mudança para o estado correspondente à retenção (*Hold*), no qual as saídas *S0* a *S14* assumem o valor lógico "0" permitindo a retenção do valor de v_{IN} nos capacitores e preparando o conversor para a etapa de redistribuição de carga.

Seguindo o processo, o sinal de *Clock* produz a mudança para o estado *Cond_gates,* cuja decomposição é paralela. Verifica-se no subestado *Shift_Reg_1* ações de "entrada" (*en*), "saída" (*ex*) e "durante" (*du*) que são desencadeadas pela transição de descida do sinal *Clock_falledge*, o que determina um registrador de deslocamento sensível à borda de descida do *clock*. Desta forma, o sinal inicialmente disponível na entrada do *flip-flop* é copiado para sua saída nos eventos de descida do sinal de *clock*. Na sequência, o dado correspondente ao *Shift_Reg_1* é copiado ao *Shift_Reg_2,* e assim sucessivamente. Paralelamente a isso, a condição *[Q1 == 1]* no elemento de armazenamento *Sto_reg1* determina que sua saída seja colocada no nível lógico "1" de maneira que o *bit* mais significativo (MSB) do conversor seja testado no processo de redistribuição de carga. Após a próxima transição do *clock*, a "resposta" do comparador é copiada (e armazenada) na sua saída.

Sucessivamente, *Sto_reg2* é colocado na condição de "ativo" ocorrendo assim o teste do próximo *bit* mais significativo, até que todos os *bits* sejam testados. Ao final do processo de redistribuição de carga uma última transição negativa do sinal de *clock* (*Clock_falledge*) produz a transição para o estado *End_of_conv* que responde pela última ação, *EOC = 0*, indicando que o dado produzido na conversão está disponível. Esse estado permanece ativo (laço infinito) até que a condição *[Startup == 1]* seja satisfeita, significando que o comando para uma nova conversão foi dado e todo o processo descrito anteriormente ocorre novamente.

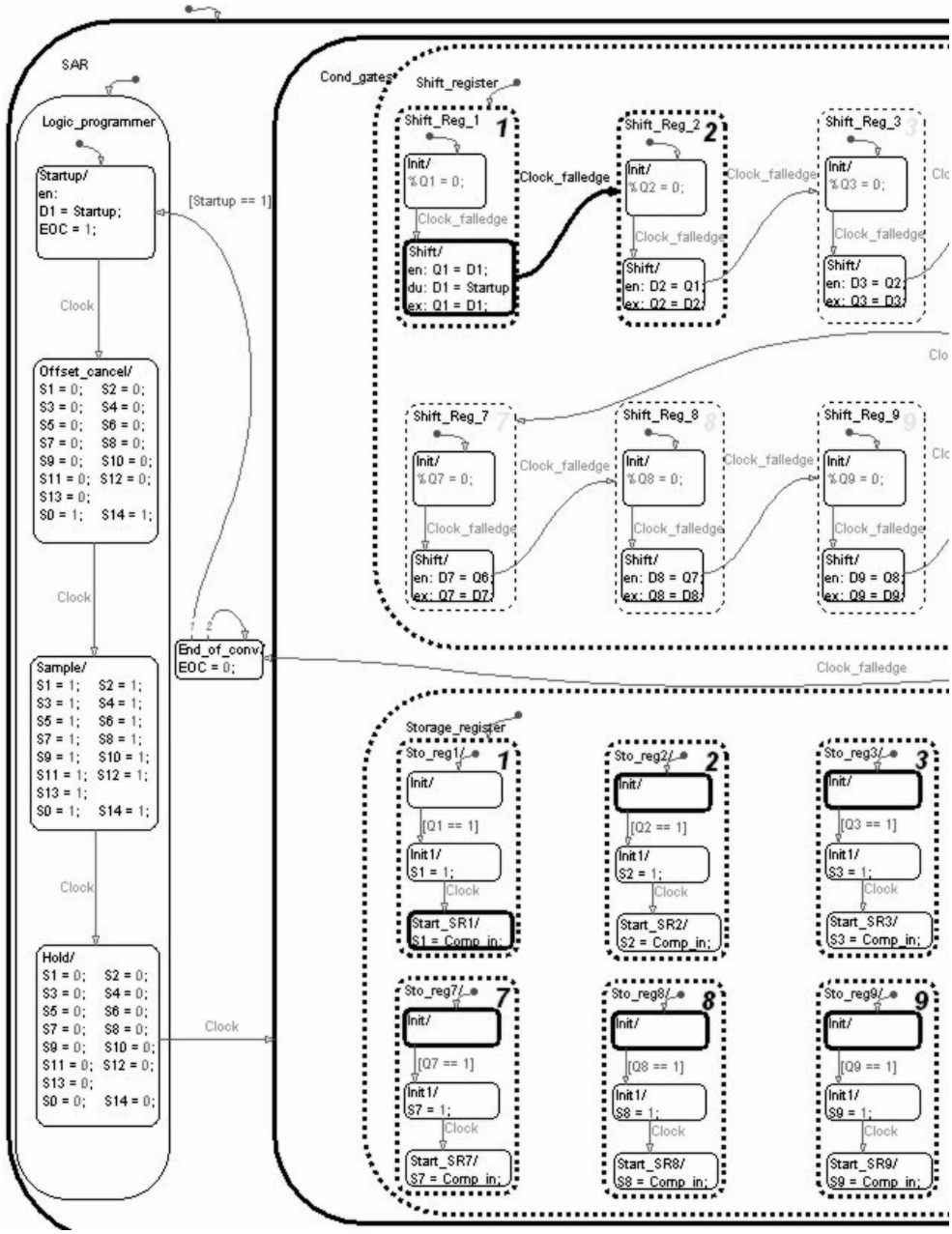

Figura 6.3: visualização parcial do processo de execução da FSM.

Cabe notar que durante a execução, cada transição é facilmente visualizada por meio de uma animação, e os tempos de execução podem ser ajustados para um acompanhamento mais detalhado (*Debug*). A figura 6.3 mostra parte do processo de animação (contornos escuros) que permite a fácil visualização do andamento das fases da máquina de estados finitos.

A porção analógica do conversor foi implementada utilizando componentes do Simscape. A figura 6.4 mostra os blocos das chaves *SW_1* e *SW_14*.

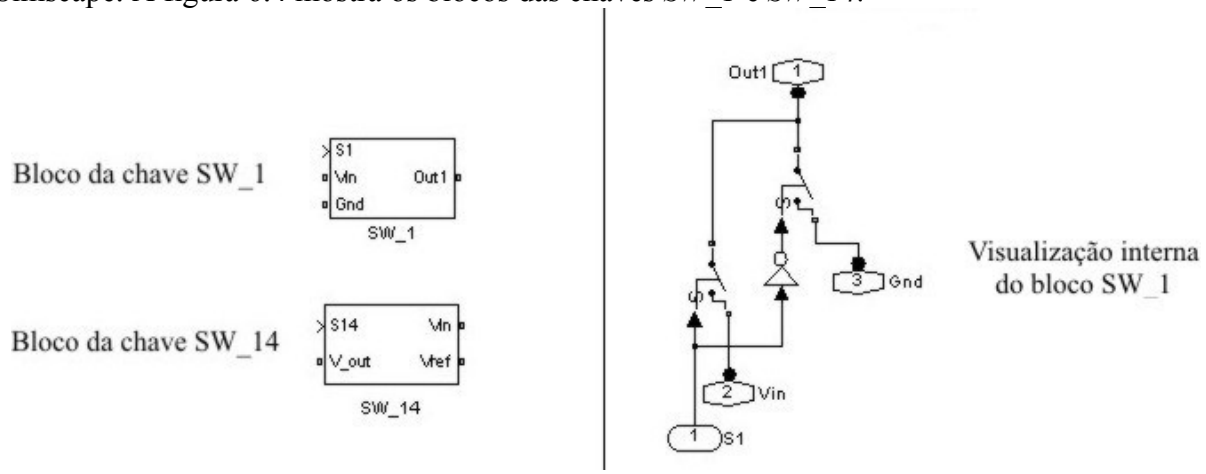

Figura 6.4: conformação dos blocos de chaves.

A criação de blocos permite a aglutinação de muitos componentes, tornando a visualização geral mais fácil. No bloco correspondente à chave *SW_1* é possível verificar o terminal *S1* que é ligado ao controle digital *SAR*. As entradas *Vin* e *Gnd* permitem a ligação da placa inferior do capacitor correspondente ao terminal de terra ou à tensão de entrada/referência. O terminal *Out1* completa o circuito de acordo com o sinal em *S1*. A chave *SW_14* é ligeiramente diferente por selecionar (de acordo com o sinal em *S14*) as entradas *Vin* ou V_{REF}, ou seja, a tensão de referência ou a tensão de entrada a ser convertida.

A figura 6.5 mostra a conformação final do ADC SAR de 12 *bits*. Os arranjos MSB e LSB (vide exemplo na figura 3.4) dos capacitores estão discriminados por meio das terminações “A” e “B”, respectivamente. Desta forma, os capacitores do arranjo MSB estão nominados como 1CA, 2CA, 4CA, etc. Igualmente, os capacitores do arranjo LSB estão nominados como 1CB, 2CB, 4CB, etc.

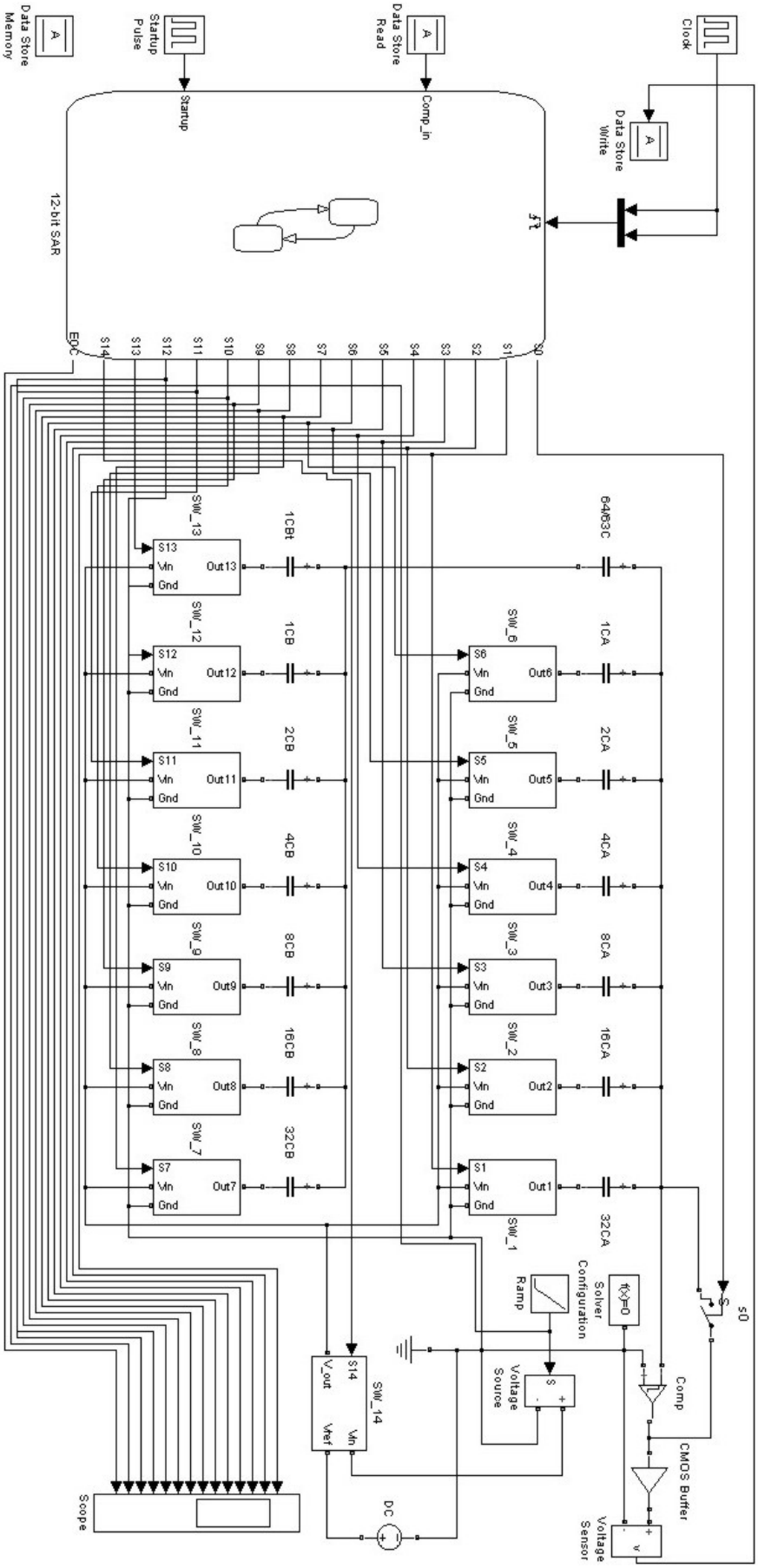

Figura 6.5: conformação final do conversor A/D SAR de 12 *bits*.

Alguns componentes que não fazem parte do circuito são requeridos pelo Simulink para que ocorra o funcionamento do modelo. *Data Store Memory*, juntamente com os blocos *Data Store Read* e *Data Store Write,* são necessários para não haver erros em portas envolvidas em laços contendo outros subsistemas e que, de alguma forma, dependam de respostas futuras. Esse tipo de erro ocorre particularmente quando se opta pelo uso da configuração "Solver" nas preferências do Simulink. Quando se utiliza a opção "Use local solver" na configuração do bloco *Solver Configuration* esse tipo de erro não ocorre. O bloco *Ramp* é o elemento gerador da rampa (aplicada em V_{IN}) usada para a excitação do conversor (0 a 3,3 V). Os blocos *Voltage Source* e *Voltage Sensor* são utilizados na interface entre os sinais do Simscape e do Simulink. O bloco *Scope* é o elemento de visualização dos sinais gerados durante a simulação (mostra a sua entrada em relação ao tempo de simulação).

Na sequência, foi feito o modelo em SystemC/SystemC-AMS com o intuito de se corroborar os resultados obtidos com o modelo apresentado.

6.2 MODELO SYSTEMC/SYSTEMC-AMS

A modelagem do conversor em SystemC/SystemC-AMS, assim como no modelo Simulink, foi segmentada em uma porção de controle digital, implementada em SystemC, e em uma porção analógica (capacitores, chaves, fontes de tensão, comparador, etc.), implementada em SystemC-AMS.

Stateflow e Simscape estão disponíveis no Simulink, que por sua vez está disponível no Matlab®. No entanto, as bibliotecas do SystemC e do SystemC-AMS devem ser compiladas e incluídas em um compilador C/C++. Além disso, as formas de onda produzidas, necessárias à visualização das saídas, também necessitam de algum tipo de programa que consiga abrir os arquivos gerados e produzir os gráficos. Foram utilizadas versões gratuitas tanto do compilador, o *Eclipse C/C++*, quanto do visualizador das formas de onda, o *GTKWave*.

O controle digital foi implementado em três módulos: *SAR*, *LOGIC_PROGRAMMER* e *END_OF_CONVERSION*.

O módulo *SAR* basicamente responde pela criação dos objetos e instanciação dos seguintes elementos:

1. Registradores de deslocamento (módulos *DFF*);
2. Elementos de armazenamento (módulos *DFF_W_EN*);
3. Módulo *LOGIC_PROGRAMMER;* e
4. Módulo *END_OF_CONVERSION*.

No módulo *SAR* ocorre também a criação das formas de onda correspondentes aos sinais de *clock* (*clk*), início de conversão (*start_up*), "resposta" do comparador (*comp_in*), saídas dos registradores de deslocamento (*q1* a *q12*), saídas de controle digital (*s1* a *s14*) e da indicação de fim de conversão (*eoc*).

A listagem 6.1 mostra os extratos do código correspondente ao módulo *SAR*. Nas linhas 17 a 23 verificam-se os trechos correspondentes à criação dos objetos relacionados aos registradores de deslocamento, elementos de armazenamento e dos módulos *LOGIC_PROGRAMMER* e *END_OF_CONVERSION*. As linhas 26 a 47 mostram os trechos correspondentes à instanciação dos objetos e nas linhas 51 a 53 observam-se os trechos correspondentes à criação das formas de onda. Ressalta-se que não são vistos nos extratos do código todos os elementos correspondentes a cada objeto.

Conforme será visto posteriormente, durante a execução do DUT (dispositivo sob teste), as formas de onda são gravadas em um arquivo VCD (*value change dump*) para posterior visualização.

```
#ifndef SAR_H_
#define SAR_H_
class SAR : public sc_core::sc_module
{
     public:
     sc_core::sc_in <bool> start_up, comp_in, clk;
     (…)
     sc_core::sc_out<bool> s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, ...;

    SC_HAS_PROCESS(SAR);

    sc_core::sc_trace_file *fp_SAR;          // Declare Local VCD filepointer

    SAR(sc_core::sc_module_name _n, sc_core::sc_trace_file *_fp_SAR):
     sc_core::sc_module(_n), fp_SAR(_fp_SAR)
    {
        dff *dff_shift1 = new dff("shift_reg1");
        (...)
        dff_w_en *dffwp_sto1 = new dff_w_en("storage_reg1");
        (...)
        logic_programmer *log_pro = new logic_programmer("logic_programmer");

        end_of_conversion *eoc = new end_of_conversion("end_of_conv");

        /*** Logic_programmer ***/
        log_pro->start_up_lp_in(start_up);
        log_pro->EOC_lp_in(EOC);
        log_pro->clk_lp_in(clk);

        /*** Shift_register ***/
        // Shift_Reg_1
        dff_shift1->din(d1);         // (external in) din = d1 (start_up), q1 = d1
        (*dff_shift1).clk(clk);      // clock negative transition
        dff_shift1->dout(q1);        // q1 = dout
        (...)

        /*** End_of_conv ***/
        (*eoc).start_up_ec_in(start_up);
        eoc->clk_ec_in(clk);
        (*eoc).q12_ec_in(q12);
        eoc->EOC_ec_out(EOC);

        /*** Storage_register ***/
        // Sto_reg1
        (*dffwp_sto1).din(comp_in); //(external in) din = comp_in, s1 = comp_in
        (*dffwp_sto1).clk(clk);      // clock negative transition
        (*dffwp_sto1).dout(s1);      // s1 = dout
        (...)

        /*** Osciloscope ***/
        sc_core::sc_trace(fp_SAR,clk,"clk");
        (...)
        sc_core::sc_trace(fp_SAR,EOC,"EOC");
    }};
#endif /* SAR_H_ */
```

Listagem 6.1: extratos do código do controle digital *SAR*.

Na listagem 6.2 estão os extratos do código correspondentes ao módulo *LOGIC_PROGRAMMER*. Diferentemente da implementação em Simulink, as fases de cancelamento da tensão de compensação do comparador, amostragem, retenção e redistribuição de carga foram implementadas como elementos de seleção *switch/case* com execução sensível

às transições do *clock* (linha 64). O *clk_lp_in* corresponde ao sinal de *clock* que alimenta o módulo, conforme verificado na instanciação no módulo *SAR* (listagem 6.1, linha 28). Assim, ao término de cada seleção (*case*) o próximo estágio é selecionado, e assim sucessivamente. A primeira seleção (*case: Start_up*) verifica se o sinal de início de conversão foi dado (ativo em nível alto) e ao mesmo tempo se não há uma conversão em curso, ou seja, se o sinal EOC (*EOC_lp_in*) está em nível alto ou baixo (linha 25). Satisfeitas as duas condições, o próximo estágio é selecionado (*case: Offset_cancel*) para que ocorra o cancelamento da tensão de compensação do comparador. As saídas de controle digital são então definidas conforme o processo usual, ou seja, *s0* e *s14* em nível lógico "1" (*true*) e *s1* a *s13* em nível lógico "0" (*false*).

```
1  #ifndef LOGIC_PROGRAMMER_H_
2  #define LOGIC_PROGRAMMER_H_
3  class logic_programmer : public sc_core::sc_module
4  {
5      private:
6      enum stage {Start_up, Offset_cancel, Sample, Hold, Charge_redistribution};
7
8      public:
9      sc_core::sc_in <bool> clk_lp_in, start_up_lp_in, EOC_lp_in;
10     (...)
11
12     SC_HAS_PROCESS(logic_programmer);
13
14     void run_up()
15     {
16            stage token = Start_up;
17
18            while(true)
19            {
20                   wait();
21
22                   switch (token)
23                   {
24                          case Start_up: // 0
25                                 if (start_up_lp_in && !EOC_lp_in)
26                                        token = Offset_cancel;
27                                 break;
28
29                          case Offset_cancel: // 1
30                                 s1_lp_out_s.write(false);
31                                 (...)
32                                 token = Sample;
33                                 break;
34
35                          case Sample: // 2
36                                 s1_lp_out_s.write(true);
37                                 (...)
38                                 token = Hold;
39                                 break;
40
41                          case Hold: // 3
```

(continua...)

```
42                                  s1_lp_out_s.write(false);
43                                  (...)
44                                  d1_lp_out.write(true);
45                                  token = Charge_redistribution;
46                                  break;
47
48                            case Charge_redistribution: // 4
49                                  s1_lp_out_s.write(false);
50                                  (...)
51                                  if (clk_lp_in.posedge())
52                                        wait();
53
54                                  d1_lp_out.write(false);
55                                  token = Start_up;
56                                  break;
57                            default: // 5
58                                  cout << "\nError: convertion phase exception\n";
59    }}}
60
61    logic_programmer(sc_core::sc_module_name name) : sc_core::sc_module(name)
62    {
63     SC_THREAD(run_up);
64     sensitive << clk_lp_in;
65    }
66 };
67 #endif /* LOGIC_PROGRAMMER_H_ */
```

Listagem 6.2: extratos do código do módulo *LOGIC_PROGRAMMER*.

O módulo *END_OF_CONVERSION*, mostrado na listagem 6.3, responde pela mudança do sinal de término de conversão (ou conversão em curso), sensível à transição negativa do *clock* (*clk_ec_in*) e da saída do último registrador de deslocamento (*q12_ec_in*), conforme verificado na linha 28. Caso seja verificado o comando de início de conversão (*start_up_ec_in*), o sinal EOC é posto em nível lógico “1” (linhas 16 e 17) e, a partir da saída do último registrador de deslocamento, o sinal EOC é posto em nível lógico “0” (linha 20), indicando o término da conversão.

```
1 #ifndef END_OF_CONVERSION_H_
2 #define END_OF_CONVERSION_H_
3 class end_of_conversion : public sc_core::sc_module
4 {
5      public:
6      sc_core::sc_in <bool> clk_ec_in, start_up_ec_in, q12_ec_in;
7      sc_core::sc_out<bool> EOC_ec_out;
8
9     SC_HAS_PROCESS(end_of_conversion);
10
11     void p1()
12     {
13     while(true)
14     {
15            wait();
16            if(start_up_ec_in)
```

(continua...)

```
17                  EOC_ec_out.write(true);
18            if(q12_ec_in)
19            {
20                  EOC_ec_out.write(false);
21            }
22     }
23    }
24
25    end_of_conversion(sc_core::sc_module_name name) : sc_core::sc_module(name)
26    {
27     SC_THREAD(p1);
28     sensitive << clk_ec_in.neg() << q12_ec_in;
29    }
30 };
31 #endif /* END_OF_CONVERSION_H_ */
```

Listagem 6.3: código do módulo *END_OF_CONVERSION*.

Os componentes analógicos do conversor, implementados em SystemC-AMS, foram encerrados no módulo *CHARGE_REDIST_DAC*. Nesse módulo, portanto, foi feita a declaração, criação de objetos e instanciação dos capacitores, chaves, terminais, fontes de tensão, nós de referência, condutores, etc. A listagem 6.4 mostra os extratos do código do sobredito módulo. Nas linhas 6 e 8 constam as declarações dos nós utilizando-se o modelo de computação ELN do SystemC-AMS. A linha 14 mostra a declaração da fonte de tensão utilizada para excitar a entrada do conversor com uma rampa variando de 0 a 3,3 V e nas linhas 22 a 26 ocorre a criação do objeto e as ligações dos nós nos terminais positivo (*p*), negativo (*n*) e de entrada de sinal (*inp*) da fonte. O "sensor" de tensão (linha 43) é utilizado para a conversão das tensões do ELN em uma saída discreta aplicável aos módulos do SystemC.

```
1 #ifndef CHARGE_REDIST_DAC_H_
2 #define CHARGE_REDIST_DAC_H_
3 class charge_redist_DAC : public sc_core::sc_module
4 {
5     private:
6     sca_eln::sca_node node_comp_out, node_vref;
7     (...)
8     sca_eln::sca_node_ref gnd;
9
10    public:
11    sc_core::sc_out <double> v_out;
12    sca_tdf::sca_in <double> v_tdf_in;
13    sca_eln::sca_vsource *v_dc; // child module declaration
14    sca_eln::sca_tdf::sca_vsource *tdf_vin;// voltage src driven by a TDF input signal
15    (...)
16    sca_eln::sca_de::sca_rswitch *sw0; // offset cancel switche
17    (...)
18
19    SC_CTOR(charge_redist_DAC)
```

(continua...)

```
20      {
21              /* Input sources */
22              tdf_vin = new sca_eln::sca_tdf::sca_vsource ("TDF_Vin");
23              (*tdf_vin).set_timestep(1.0, sc_core::SC_NS);//set timestep/period
24              (*tdf_vin).p(node_vin);
25              (*tdf_vin).n(gnd);
26              (*tdf_vin).inp(v_tdf_in);
27
28              v_dc = new sca_eln::sca_vsource("v_DC");
29              (*v_dc).offset = 3.3; // 3.3V DC Source
30              (*v_dc).p(node_vref);
31              (*v_dc).n(gnd);
32              (...)
33
34              /* Switches */
35              sw0 = new sca_eln::sca_de::sca_rswitch ("sw0");
36
37              (*sw0).ctrl(s0_ctrl); // 1st s5 switch. Off_state => s5 = 0;
38              (*sw0).p(node_vx);
39              (*sw0).n(node_comp_out);
40              (...)
41
42              /* Voltage sensor */
43              v_sensor = new sca_eln::sca_de::sca_vsink ("V_Sensor", 1.0);
44
45              (*v_sensor).p(gnd);
46              (*v_sensor).n(node_vx);
47              (*v_sensor).outp(v_out);
48
49              (...)
50      }
51 };
52 #endif /* CHARGE_REDIST_DAC_H_ */
```

Listagem 6.4: extratos do código do módulo *CHARGE_REDIST_DAC*.

Um módulo que não faz parte do conversor mas que é necessário na fase de teste é o gerador de rampa, mostrado na listagem 6.5, que alimenta a tensão de entrada v_{IN}. O módulo utiliza o modelo de computação TDF do SystemC-AMS para produzir uma simples rampa, com inclinação determinada pela variável *slope* (linha 9).

```
1 #ifndef RAMP_SRC_H_
2 #define RAMP_SRC_H_
3 SCA_TDF_MODULE (ramp_SRC)
4 {
5       sca_tdf::sca_out <double> v_tdf_out;
6       double t, slope;
7       void processing()
8       {
9               slope = 3.3e4;
10              t = get_time().to_seconds();
11              v_tdf_out.write(slope * t);
12      }
13      SC_CTOR (ramp_SRC)
14      {
15              set_timestep(1.0, sc_core::SC_NS);
16 }}; #endif /* RAMP_SRC_H_ */
```

Listagem 6.5: módulo gerador de rampa.

Um extrato do código do *testbench* do conversor é mostrado na listagem 6.6. A linha 18 mostra a definição do *clock* de 2 MHz. Nas linha 20 e 24 são criados os arquivos de saída VCD que são visualizados por meio do *TwinWave* (linha 57).

```
1 #include <systemc-ams.h>
2 #include "dff.h"
3 #include "dff_w_en.h"
4 #include "logic_programmer.h"
5 #include "end_of_conversion.h"
6 #include "SAR.h"
7 #include "charge_redist_DAC.h"
8 #include "comparator.h"
9 #include "ramp_SRC.h"
10
11 int sc_main(int argc, char* argv[])
12 {
13    sc_core::sc_set_time_resolution(1.0, sc_core::SC_NS);
14    sca_eln::sca_node node_v_check, node_vx_check, node_vin_check;
15    sca_tdf::sca_signal <double> vin_check;
16    sc_core::sc_signal <bool> start_up_pulse, EOC;
17    (...)
18    sc_clock clk  ("clk" ,500 , SC_NS, 0.5, 0, SC_NS, true); // clock f = 2.0 MHz
19    sc_core::sc_trace_file *fp_SAR; // VCD filepointer
20    fp_SAR = sc_core::sc_create_vcd_trace_file("SAR_12bit"); // Create vcd file
21    fp_SAR -> set_time_unit(1.0, sc_core::SC_NS); // set tracing resolution to ns
22
23    sca_util::sca_trace_file *fp_C_RED;
24    fp_C_RED = sca_util::sca_create_vcd_trace_file("SAR_Chrg_redist");
25    sca_util::sca_write_comment( fp_C_RED, "Charge redistribution data." );
26    sca_util::sca_trace(fp_C_RED, node_v_check ,"node_v");
27    sca_util::sca_trace(fp_C_RED, node_vx_check ,"node_vx");
28    (...)
29    ramp_SRC ramp("Ramp");
30    ramp.v_tdf_out(vin_check);
31
32    charge_redist_DAC DAC("DAC");
33    DAC.s0_ctrl(s0_out);
34    (...)
35    DAC.node_vin(node_vin_check);
36    DAC.v_tdf_in(vin_check);
37    DAC.v_out(comp_input_check);
38
39    comparator comp("Comparator");
40    comp.input(comp_input_check);
41    comp.output(comp_sig_check);
42
43    SAR DUT("DUT_SAR", fp_SAR);   // Instantiate Device Under Test
44    DUT.clk(clk);                 // Connect ports
45    DUT.start_up(start_up_pulse); // d1 = startup
46    DUT.comp_in(comp_sig_check);  // comp_in = comp_sig
47    DUT.q1(q1_out);   //q_out_1 = q1
48    (...)
49    DUT.s0(s0_out);   //s0_out = s0
50    sc_core::sc_start(0, SC_NS);   // start_up = 1
51    start_up_pulse = 1;
52    (...)
53    sc_core::sc_stop();   // makes the simulation stop
54    sc_core::sc_close_vcd_trace_file(fp_SAR); // close .vcd
55    sca_util::sca_close_vcd_trace_file(fp_C_RED);
56
57    system("twinwave SAR_12bit.vcd sar.sav + SAR_Chrg_redist.vcd chrgsc.sav");
58    return 0;
59 } // end of sc_main
```

Listagem 6.6: extrato do código do *testbench* do ADC SAR de 12 *bits*.

A implementação do conversor produziu cerca de 1060 linhas de código, adicionando-se cerca de 180 linhas para a criação do *testbench*.

A seguir, será mostrada a última etapa do desenvolvimento que é a criação do protótipo do conversor.

6.3 DESENVOLVIMENTO DO PROTÓTIPO (CI)

Seguindo a sequência da metologia, terminada a fase de criação dos modelos, procedeu-se ao desenvolvimento do protótipo em silício. Para tal, foram utilizadas as especificações para o processo CMOS 0,35 μm 3,3 V da X-FAB, conjuntamente com as ferramentas DFII da Cadence.

Para uma maior fluidez no desenvolvimento, o sistema foi segmentado em blocos que passaram pelas seguintes fases:

1. Criação do esquemático;
2. Criação do símbolo;
3. Verificação do funcionamento do bloco quanto ao atendimento a determinado requisito funcional (criação de um DUT);
4. Criação do *layout* correspondente e verificação do cumprimento das regras de desenho (DRC);
5. Produção do *extracted view*: extração da *netlist* do circuito a partir do *layout*;
6. Verificação da correspondência entre o *layout* (*extracted view*) e o esquemático (LVS);
7. Produção do *analog-extracted view*: versão aumentada do *extracted view* utilizada na simulação analógica no Spectre. O Spectre é similar ao SPICE em termos de simulação do comportamento analógico dos transistores (Brunvand, 2010); e
8. Verificação do funcionamento do bloco na versão do *layout* (*analog-extracted view*).

Nesta implementação prática, modificações no cir*cui*to foram necessárias para que o mesmo funcionasse a contento. O comparador, as chaves e capacitores, por exemplo, foram implementados de maneira a permitir o funcionamento adequado do circuito, bem como para minimizar os efeitos dos componentes parasitas. A figura 6.6 mostra a conformação do conversor implementada no DFII. As chaves SW_1 a S_6 e SW_7 a SW13 correspondem aos arranjos MSB e LSB dos capacitores, respectivamente. SW_0 corresponde à chave de ligação entre o nó *Vx* e a saída do comparador. O módulo de controle SAR é visto na figura como *12-bit SAR*.

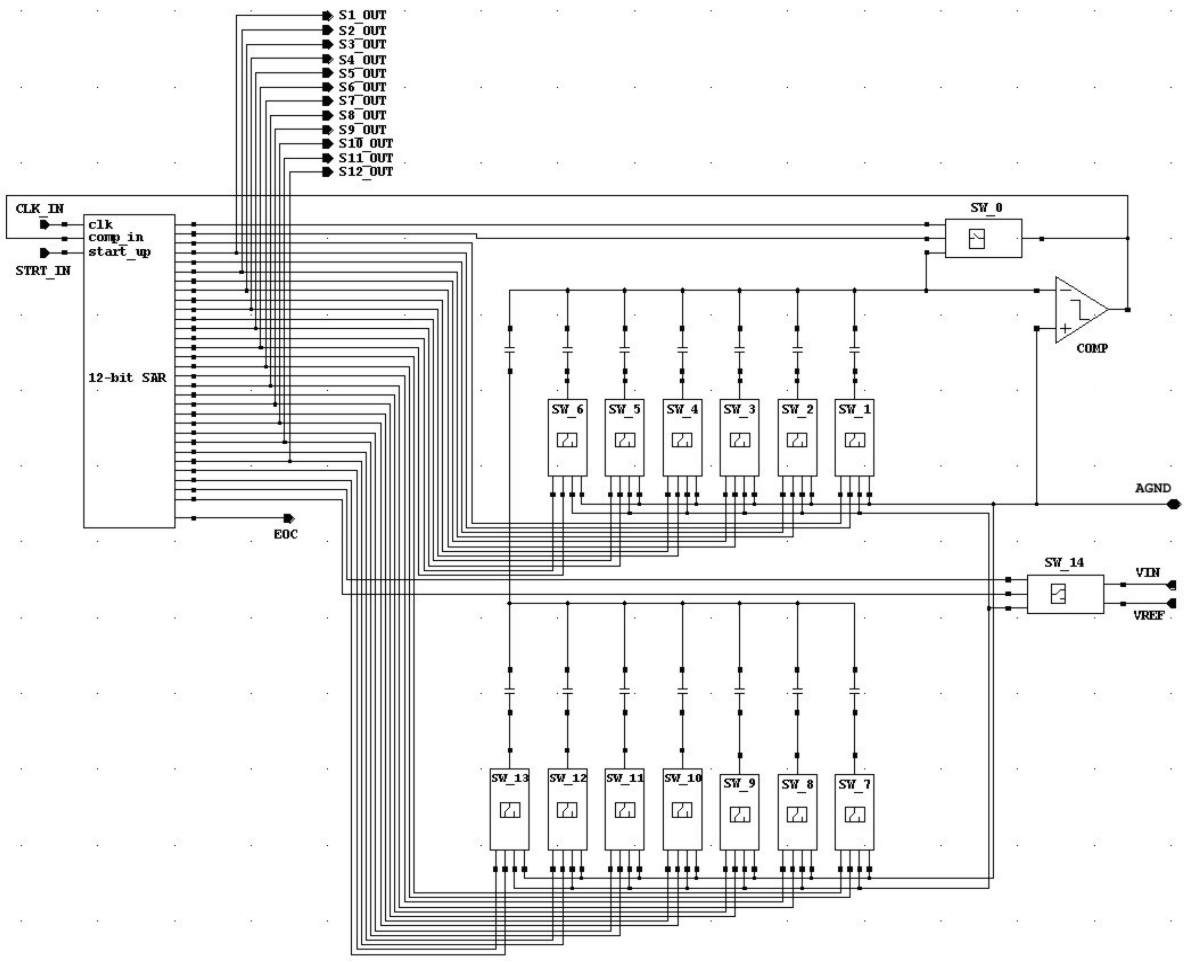

Figura 6.6: blocos interligados do ADC SAR de 12 *bits*.

Os blocos com função inerentemente digital (*SAR*) foram implementados utilizando-se de *standard cells* (esquemático e *layout*), ou células padrão, fornecidas pela X-FAB. Esse procedimento reduziu sensivelmente o tempo de desenvolvimento dessa porção do sistema. A figura 6.7 mostra o esquemático do controle digital *SAR* juntamente com o respectivo *layout*.

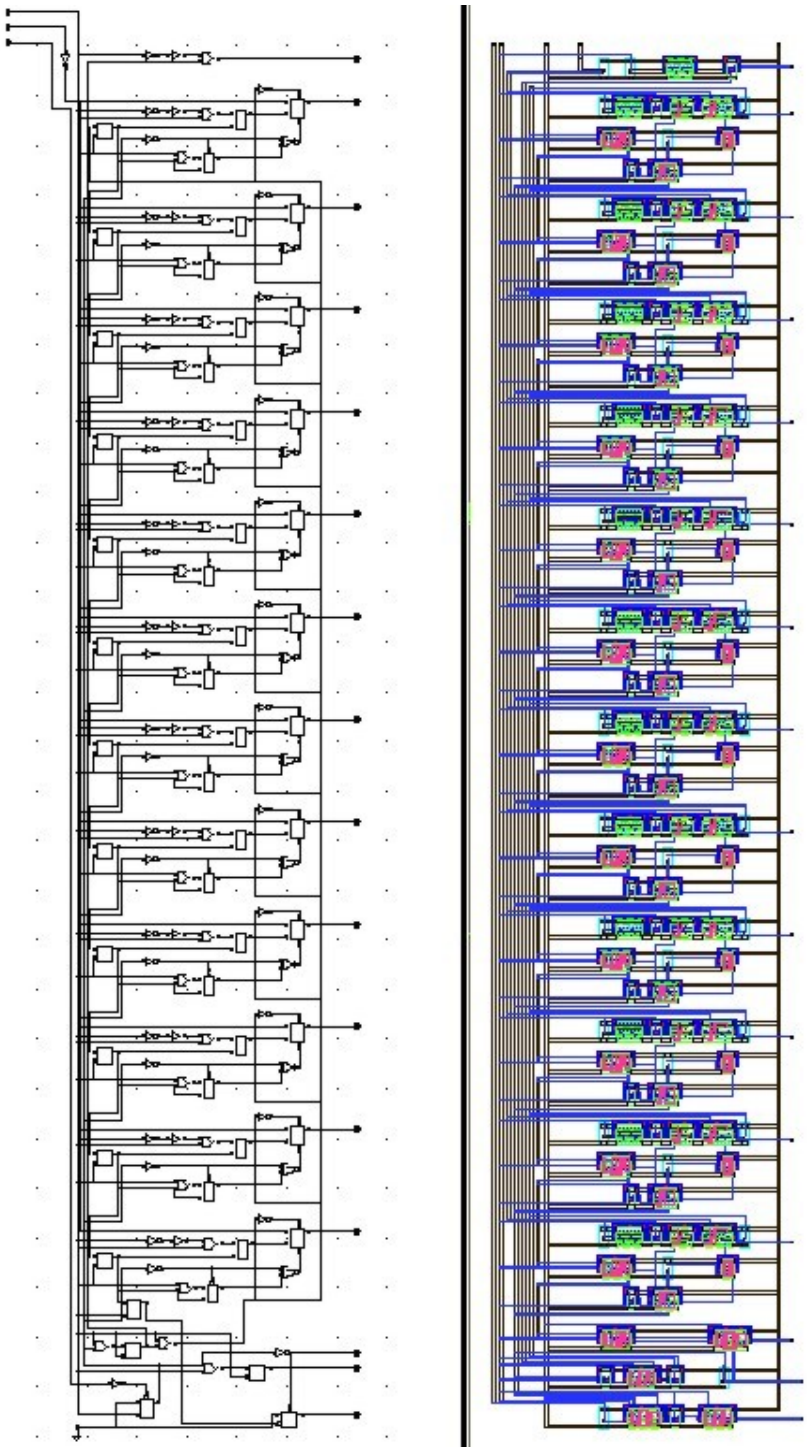

Figura 6.7: esquemático do controle digital SAR (esquerda) e respectivo *layout* (direita).

As chaves analógicas foram projetadas aplicando-se as técnicas usuais para redução dos efeitos de injeção de carga e de alimentação indireta por meio do sinal de *clock*. A chave SW_0 não foi implementada utilizando a configuração PMOS/NMOS em paralelo (TG); foi implementada utilizando um transistor PMOS em série com uma chave fictícia (PMOS com metade do tamanho da chave principal) devido às tensões negativas no nó *Vx* que produziriam níveis de corrente suficientemente elevadas para levar o transistor NMOS à ruptura, prejudicando o funcionamento da chave. As demais chaves (SW_1 a SW_14) foram implementadas como TGs ordinárias, com relação 2:1 entre a largura dos transistores PMOS e NMOS.

Para melhorar a paridade entre os elementos no *layout*, cada chave foi ladeada por transistores fictícios (*dummy elements*) nos quais todos os terminais dos transistores PMOS foram ligados ao V_{DD} analógico e todos os terminais dos transistores NMOS foram ligados ao terra. As figuras 6.8 e 6.9 mostram, respectivamente, o esquemático e o *layout* da chave interligada à placa inferior de cada capacitor.

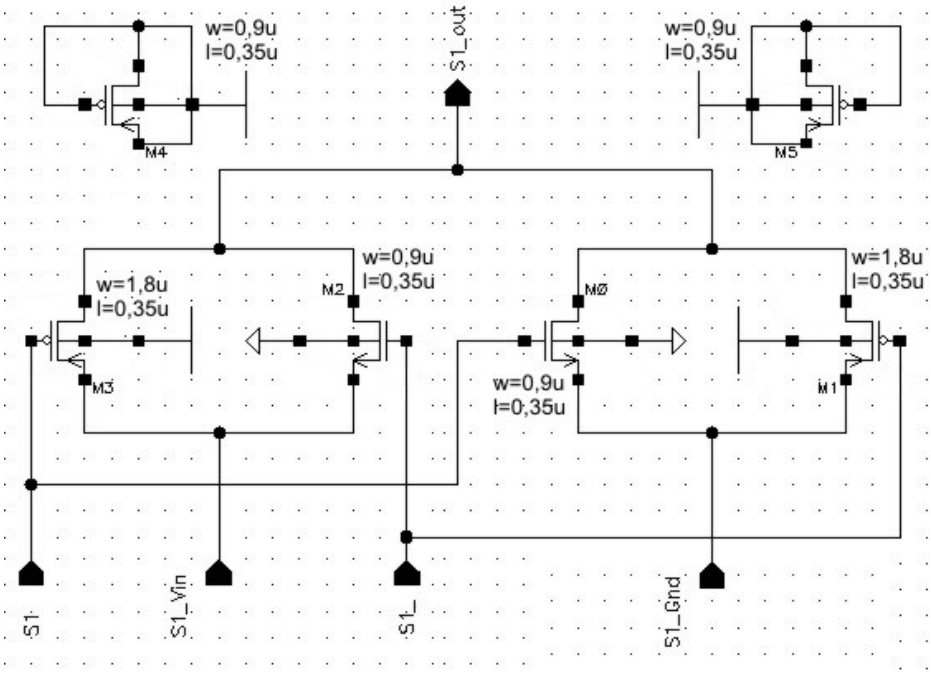

Figura 6.8: esquemático da chave interligada à placa inferior do capacitor.

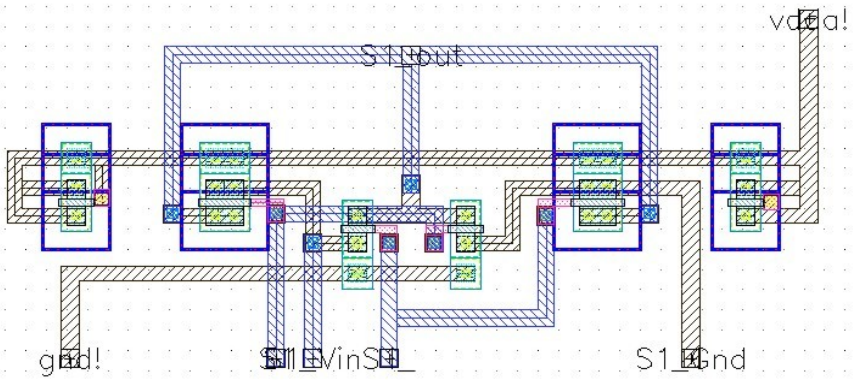

Figura 6.9: *layout* da chave interligada à placa inferior do capacitor.

Na figura acima, a largura dos transistores PMOS e NMOS segue uma relação 2:1, ou seja, W = 1,8 µm e L = 0,35 µm para o PMOS e W = 0,9 µm e L = 0,35 µm para o NMOS. Os transistores PMOS vistos nos lados são fictícios e de tamanho mínimo (W = 0,9 µm e L = 0,35 µm).

As figuras 6.10 e 6.11 mostram, respectivamente, o esquemático e o *layout* da chave que interliga o nó *Vx* à saída do comparador (SW_0).

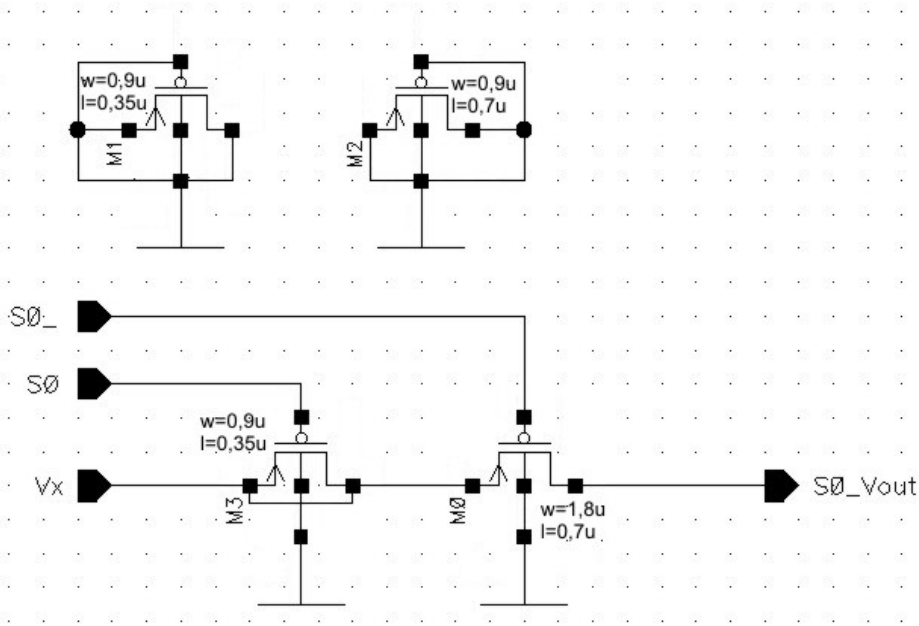

Figura 6.10: esquemático da chave correspondente ao controle digital *S0* (SW_0).

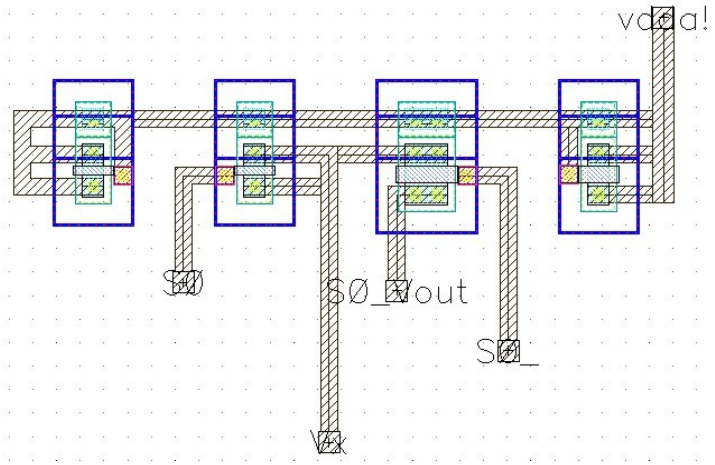

Figura 6.11: *layout* da chave correspondente ao controle digital *S0* (SW_0).

A chave SW_0, mostrada na figura 6.11, é formada por um transistor PMOS, com dimensões W = 1,8 µm e L = 0,7 µm, em série com um transistor (também PMOS) com dimensões mínimas (W = 0,9 µm e L = 0,35 µm) utilizado para mitigar os efeitos de injeção de carga. Ambos são ladeados por transistores fictícios de largura mínima (W = 0,9 µm).

As figuras 6.12 e 6.13 mostram, respectivamente, o esquemático e o *layout* da chave interligada às tensões de entrada e de referência (V_{IN} e V_{REF}).

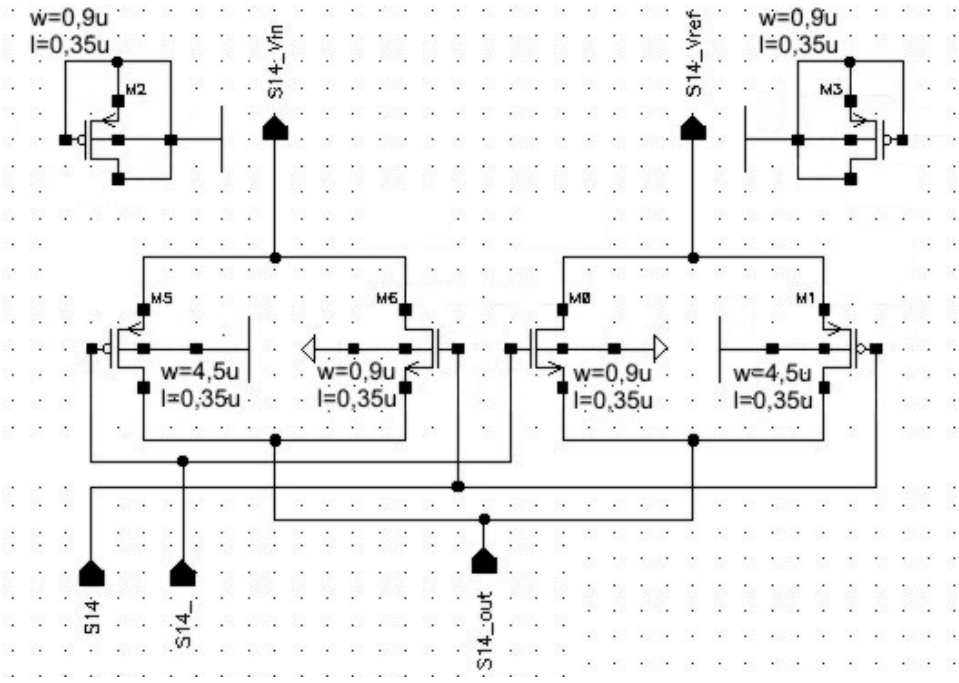

Figura 6.12: esquemático da chave interligada à tensão de entrada e de referência (V_{IN} e V_{REF}).

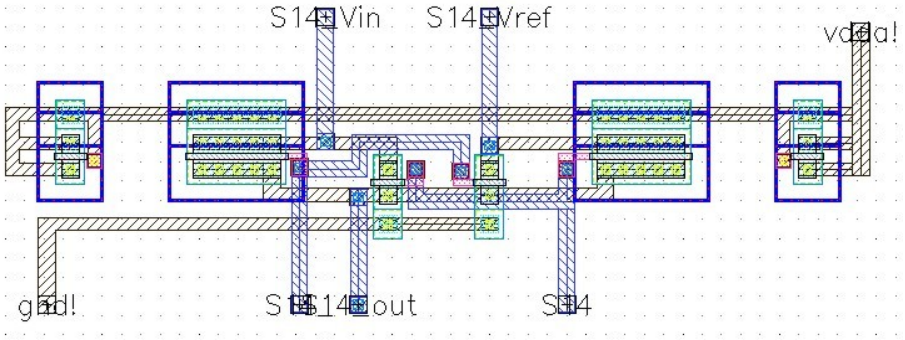

Figura 6.13: *layout* da chave interligada à tensão de entrada e de referência (V_{IN} e V_{REF}).

A chave SW_14, mostrada na figura 6.13, é constituída por transistores PMOS e NMOS cuja largura segue uma relação 5:1, ou seja, W = 4,5 µm e L = 0,35 µm para o PMOS e W = 0,9 µm e L = 0,35 µm para o NMOS. Os transistores PMOS vistos nos lados são fictícios e de tamanho mínimo (W = 0,9 µm e L = 0,35 µm).

No processo de escolha do comparador três tipos foram testados: comparador de três estágios diretamente acoplados, comparador com histerese e comparador de uso geral, linha-a-linha, com entrada em modo comum. Os dois primeiros tipos não apresentaram desempenho satisfatório, com baixo ganho e, consequentemente, baixa resolução. O último tipo apresentou um desempenho melhor, com ganho comparativamente maior. Além disso, não necessitava de sinais de controle adicionais e funcionava com fonte única (*single supply*).

O comparador utilizado no conversor foi o de uso geral, linha-a-linha, com entrada em modo comum. As figuras 6.14 e 6.15 mostram, respectivamente, o esquemático e *layout* do comparador.

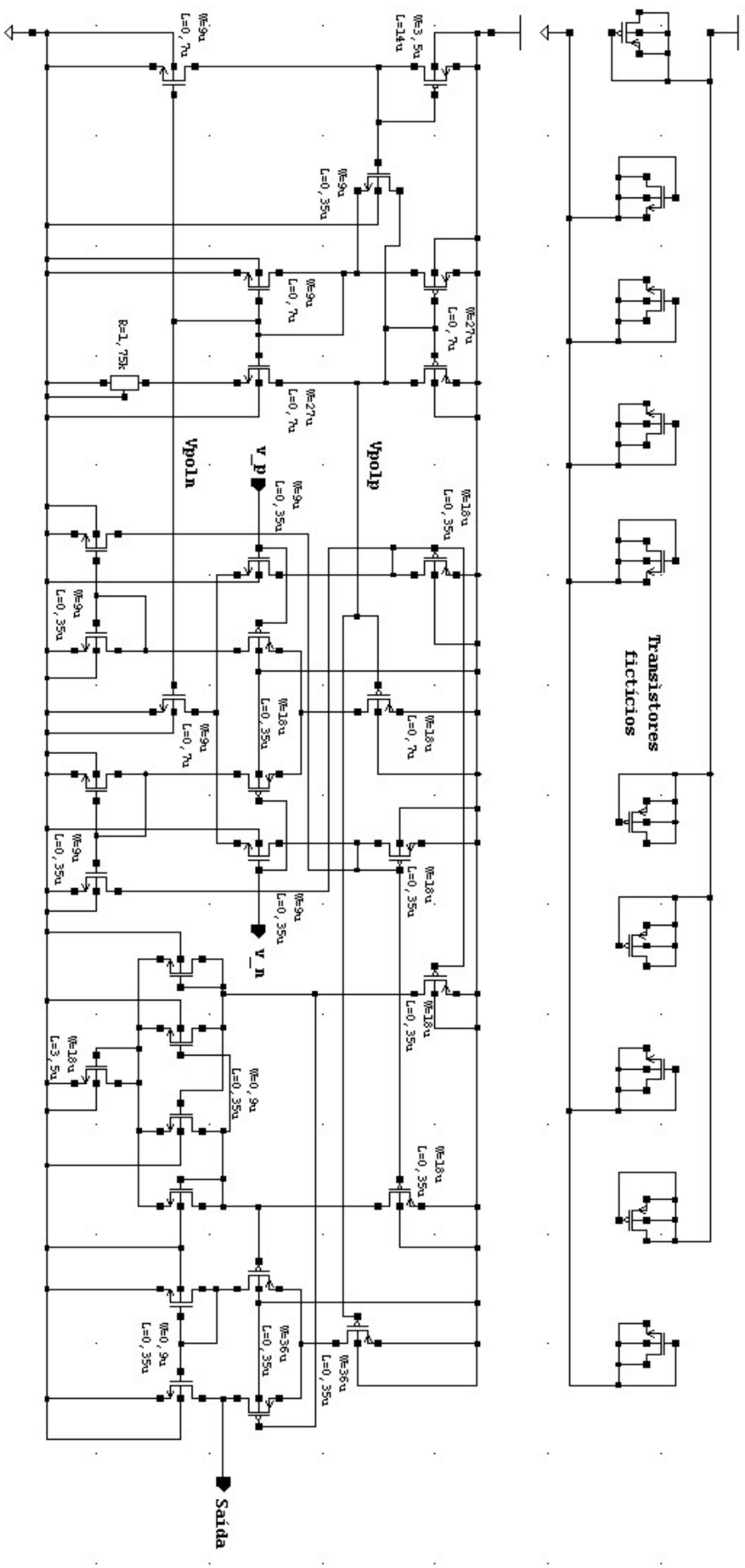

Figura 6.14: esquemático do comparador.

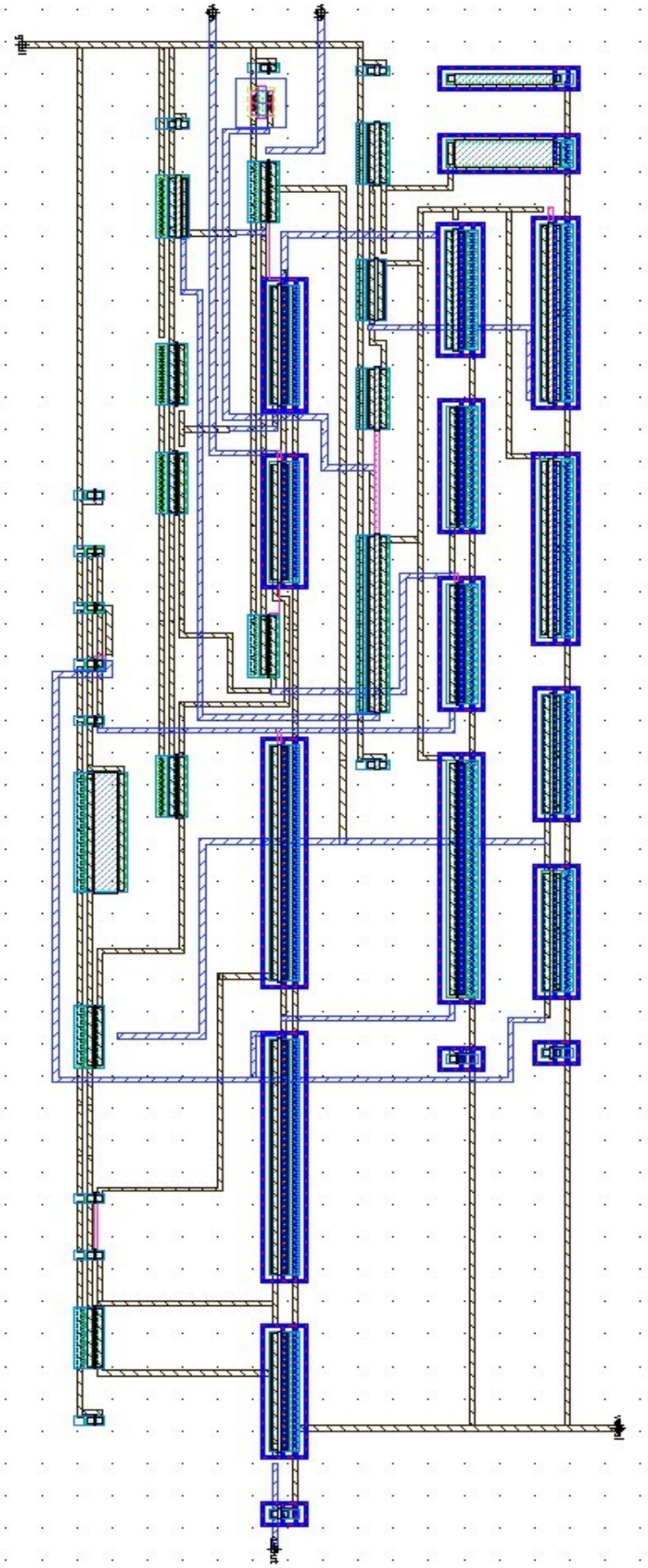

Figura 6.15: *layout* do comparador.

O banco de capacitores seguiu o formato de implementação de McCreary (McCreary et al 1975), com a aplicação da geometria centroide comum (common-centroid), de maneira a se obter as capacitâncias somadas de cada segmento (32C, 16C, 8C, 4C, 2C e 1C) a partir de capacitores unitários. A criação do *layout* com um centro comum ajuda a melhorar a paridade entre os capacitores. Da mesma maneira em que foram aplicados elementos fictícios nas chaves, tais elementos também foram aplicados ao banco de capacitores (Hastings, 2001).

A figura 6.16 mostra o *layout* do arranjo utilizando o esquema geométrico centroide comum. Os capacitores mais externos são capacitores fictícios.

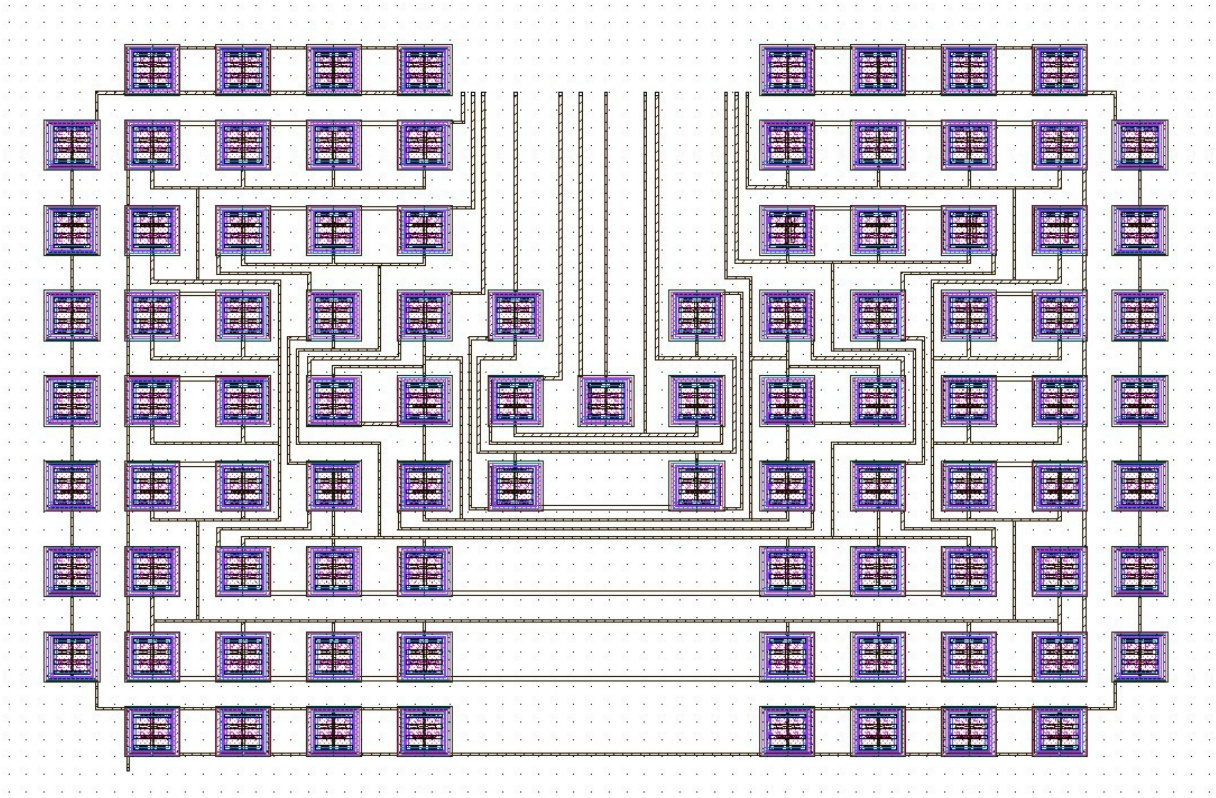

Figura 6.16: *layout* do banco de capacitores (1C a 32C).

De acordo com Johns (Johns *et al,* 1997), precauções em relação ao *layout* devem ser tomadas para ajudar a minimizar o ruído nos circuitos analógicos. Mudanças de estado nos componentes digitais injetam ruído na fonte de alimentação e no substrato circundante. A utilização de fontes separadas previne a contaminação dos circuitos analógicos por esse tipo de

ruído. Em um caso ideal, pinos separados são utilizados para as fontes de tensão e para aterramento em ambos os circuitos (digital e analógico). Outra precaução comum é posicionar os circuitos digital e analógico em seções diferentes no microcircuito. As duas seções devem ser separadas por anéis de guarda (*guard rings*) e poços conectados às fontes.

Conforme mostrado na figura 6.17, os anéis de guarda foram implementados com um poço *n* central com regiões n^+ conectadas ao V_{DD} e regiões p^+ laterais conectadas ao terra. A conexão de regiões p^+ ao terra ajudam a manter um caminho de baixa impedância entre o substrato e o terra. A utilização do poço *n* entre as conexões p^+ ajuda a aumentar a impedância do subtrato entre as regiões analógica e digital devido à dopagem gradual do subtrato.

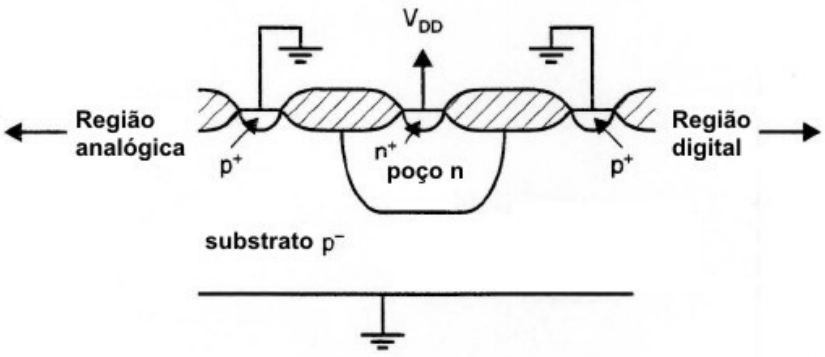

Figura 6.17: separação de áreas analógica e digital com anéis de guarda.

As figuras 6.18 e 6.19 mostram, respectivamente, o mapa de pinos e o correspondente *layout* com a conformação final do ADC SAR de 12 *bits*.

Os pinos de alimentação foram separados em VDD_D/GND_D (digital) e VDD_A/GND_A (analógico). CLK e STRT são os pinos correspondentes aos sinais de *clock* e de início de conversão, respectivamente. V_IN e V_REF são os pinos de entrada das tensões de referência e do sinal a ser convertido, respectivamente.

EOC corresponde ao pino com a saída indicadora de conversão em curso ou fim de conversão. Os pinos S1 a S12 são os pinos de saída da palavra oriunda da conversão.

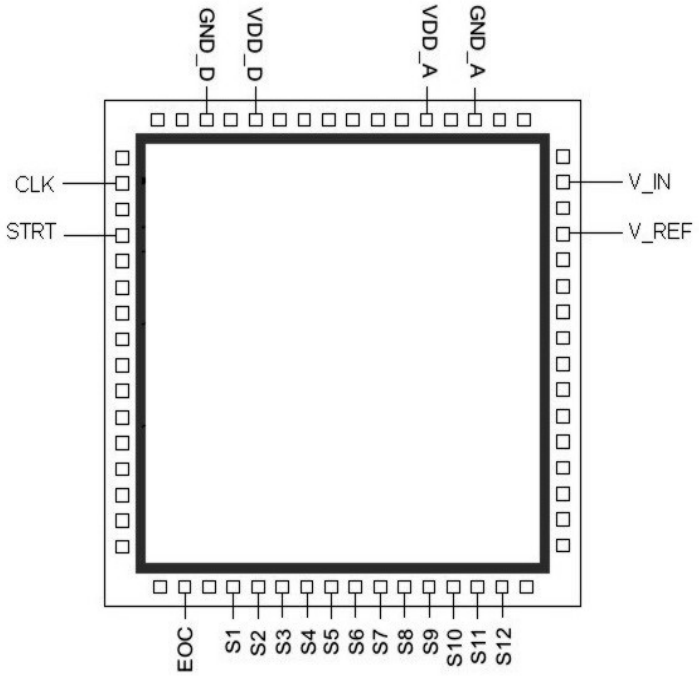

Figura 6.18: mapa de pinos.

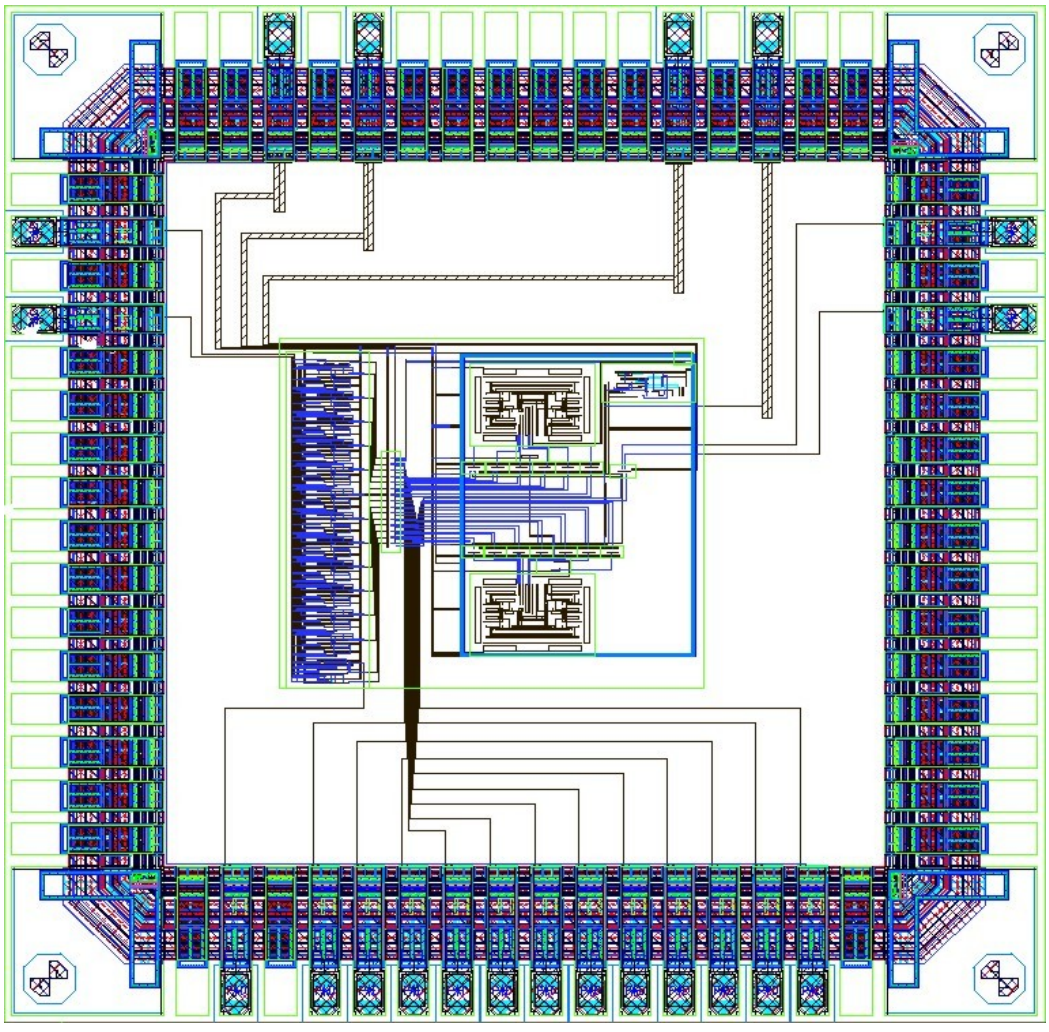

Figura 6.19: *layout* do *chip*.

7 RESULTADOS

7.1 MODELO STATEFLOW/SIMSCAPE

O modelo implementado utilizando o Stateflow e o Simscape se mostrou condizente com o comportamento desejado do circuito. Utilizando uma tensão de referência (V_{REF}) de 3,3V, obtém-se o tamanho do degrau (teórico) de aproximadamente $V_{REF}/2^{12}$ = 0,0008056641V. A figura 7.1 mostra a saída do conversor mediante tensões de entrada de 1,650 V e 2,350 V.

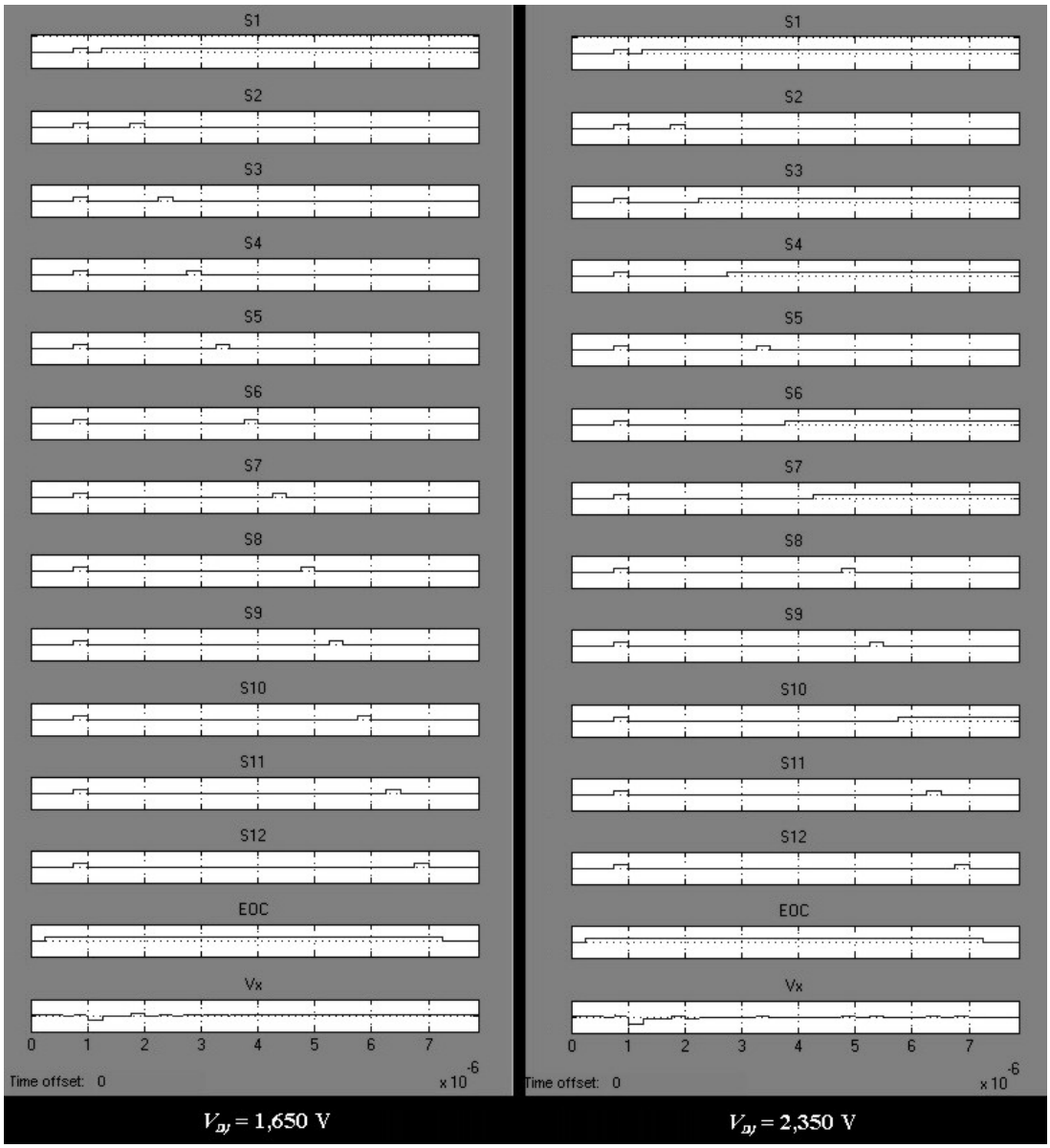

Figura 7.1: saída do conversor para valores de entrada V_{IN} = 1,650 e 2,350 V.

O valor de tensão correspondente ao meio da escala (2048_{10}) é igual a 1,650 V. Assim, alimentando a entrada do circuito com uma tensão V_{IN} = 1,650 V, foi obtido o valor 100000000000_2 (2048_{10}). Para um valor aleatório V_{IN} = 2,350 V, equivalendo a 2916_{10}, foi obtido o valor 101101100100_2 (2916_{10}).

A figura 7.2 mostra o processo de busca binária em *Vx*, nó ligado às placas superiores dos capacitores e na entrada do comparador, para a tensão de entrada V_{IN} = 1,650 V. A tensão armazenada no arranjo de capacitores, no momento da retenção (*hold*), é mostrada como $Vx = -V_{IN}$ = -1,650 V. No processo de redistribuição de carga, o valor de V_{IN} é "comparado" com os diferentes pesos binários dos respectivos capacitores. Desta forma, uma vez que somente S1 (MSB) permanece em nível alto (vide figura 7.1), os demais *bits* são testados apresentando em *Vx* as suas respectivas tensões, ou seja, $V_{REF}/4$, $V_{REF}/8$, etc.

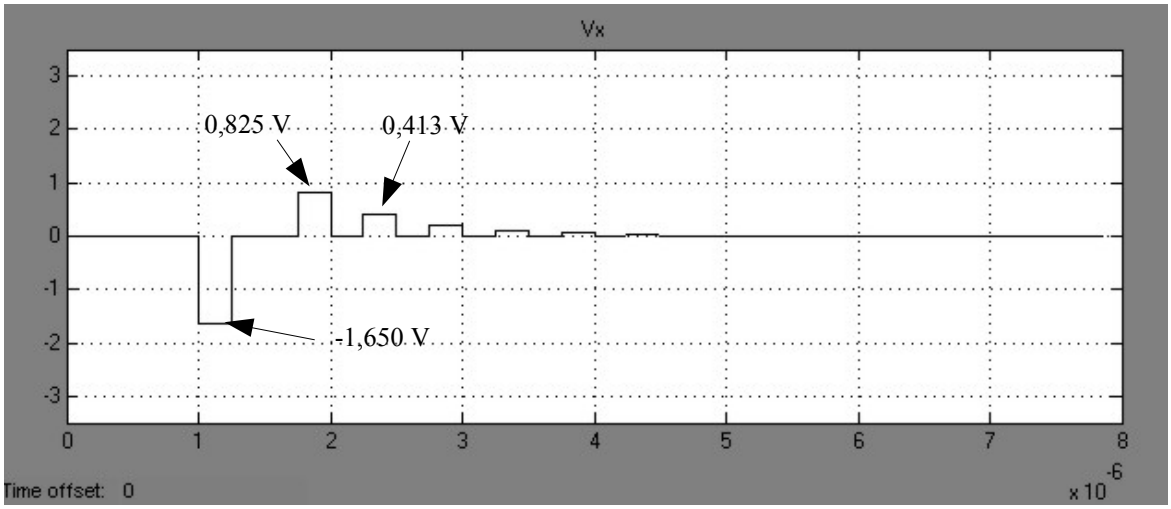

Figura 7.2: busca binária no nó *Vx*.

Nesse modelo não houve introdução de fatores como capacitâncias parasitas de transistores e chaves, bem como a resistência dos condutores, sensibilidade do comparador, etc. Entretanto, o modelo permitiu o entendimento e a verificação do funcionamento dos circuitos e da interação entre os diferentes blocos componentes do sistema, o que foi essencial para a decisão acerca da viabilidade do projeto.

7.2 Modelo SystemC/SystemC-AMS

O modelo implementado em SystemC/SystemC-AMS permitiu, além da verificação da validade do modelo produzido em Simulink (Stateflow/Simscape), a produção de módulos que podem ser incorporados em sistemas com complexidade maior por meio do reuso. Assim, o desenvolvimento de um determinado SoC que necessite de um conversor A/D, com as características do SAR, pode incorporar os módulos já implementados.

Na verificação do funcionamento do modelo, inicialmente foram aplicadas as mesmas tensões de entrada V_{IN} = 1,650 e 2,350 V, resultando nas formas de onda das figuras 7.3 e 7.4, respectivamente. Igualmente, é possível acompanhar o sinal no nó *Vx* (nominado *node_vx*) à medida que ocorre a busca binária. Podem ser verificados também, os processos iniciais de cancelamento da tensão de compensação do comparador, amostragem, retenção e o tempo de conversão com duração aproximada de 7 μs.

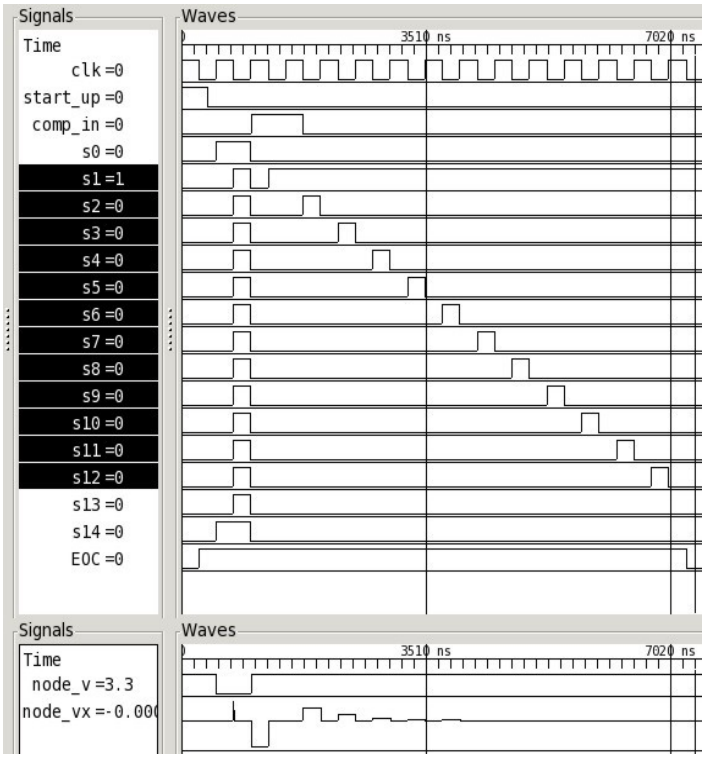

Figura 7.3: saída do conversor para o valor de entrada V_{IN} = 1,650 V.

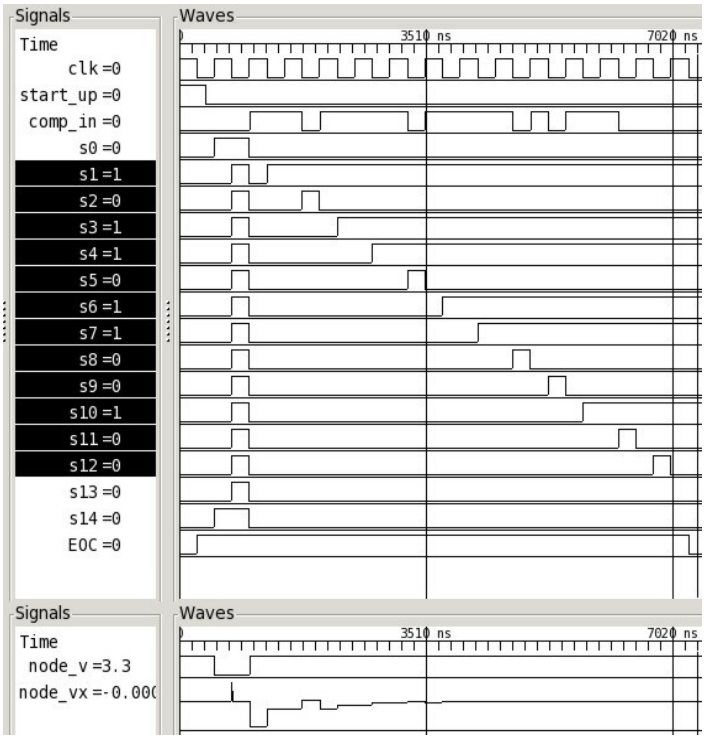

Figura 7.4: saída do conversor para o valor de entrada V_{IN} = 2,350 V.

A figura 7.5 mostra a saída de uma conversão, com treze amostras, mediante a entrada de um sinal em rampa, variando de 0 a 3,3 V.

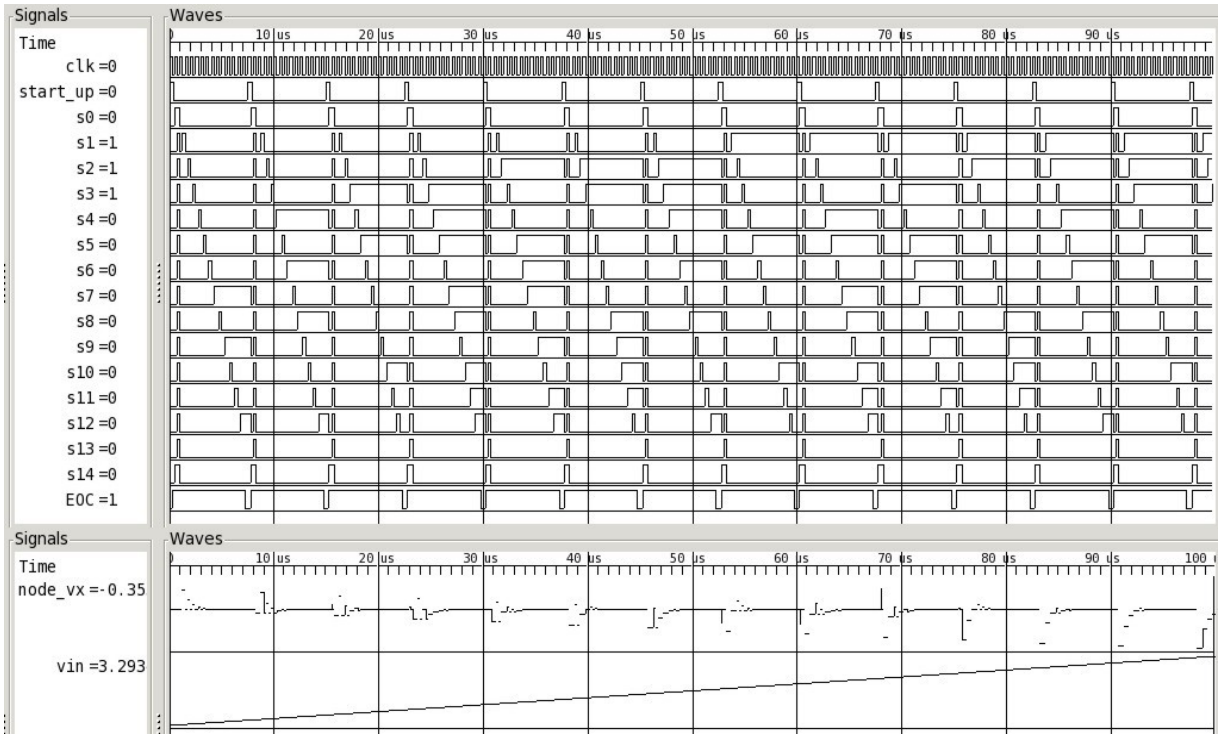

Figura 7.5: saída do conversor com treze amostras para uma entrada em rampa (0 a 3,3 V).

O modelo produzido em SystemC/SystemC-AMS se mostrou mais flexível e com abrangência maior do que o produzido em Stateflow/Simscape, permitindo também a criação de módulos IP (*intellectual property*). Assim, a incorporação desse modelo de conversor em sistemas mais complexos pode se mostrar natural, beneficiando-se das vantagens do ambiente SystemC.

7.3 DESENVOLVIMENTO DO PROTÓTIPO (CI)

O desenvolvimento do protótipo demandou um maior cuidado com a escolha dos componentes por envolver elementos de degradação do funcionamento do circuito, como capacitâncias parasitas, bem como limitações acerca dos níveis máximos de corrente e tensão ditados pela tecnologia utilizada (X-FAB). Valores máximos e mínimos das dimensões dos transistores também tiveram que ser considerados.

As simulações do circuito no Spectre foram feitas utilizando-se a *analog-extracted view* por esta conter as componentes parasitas do circuito, incluindo-se as originárias da interação entre os condutores. As capacitâncias parasitas limitam o desempenho geral do sistema não somente no aspecto da frequência de operação como também se mostram críticas em um conversor que utiliza o arranjo de capacitores como DAC. Isso implica que, aos valores dos próprios capacitores do arranjo, são adicionados os valores das capacitâncias parasitas, havendo assim, interferência na resposta do conversor.

O comparador se mostrou um componente crítico pois, de fato, representa o elemento decisor do conversor. Na sua caracterização foram verificados o desempenho em corrente contínua (CC), resposta ao transitório e ganho. As figuras 7.6 e 7.7 mostram o desempenho CC e o ganho, respectivamente.

Na caracterização do desempenho em corrente contínua foi feita uma varredura na entrada positiva (*vp*) do comparador de 0 a V_{DD} (3,3 V), enquanto a entrada negativa (*vn*) variou de 0 a 3,3 V, em incrementos de 330 mV, para mostrar a variação completa em modo comum.

O ganho do comparador foi medido com a entrada negativa mantida em 1,650 V, enquanto a entrada positiva variou de 1,649 a 1,651 V. A derivada da curva de transferência mostrou um ganho 26.200 ou aproximadamente 88,4 dB.

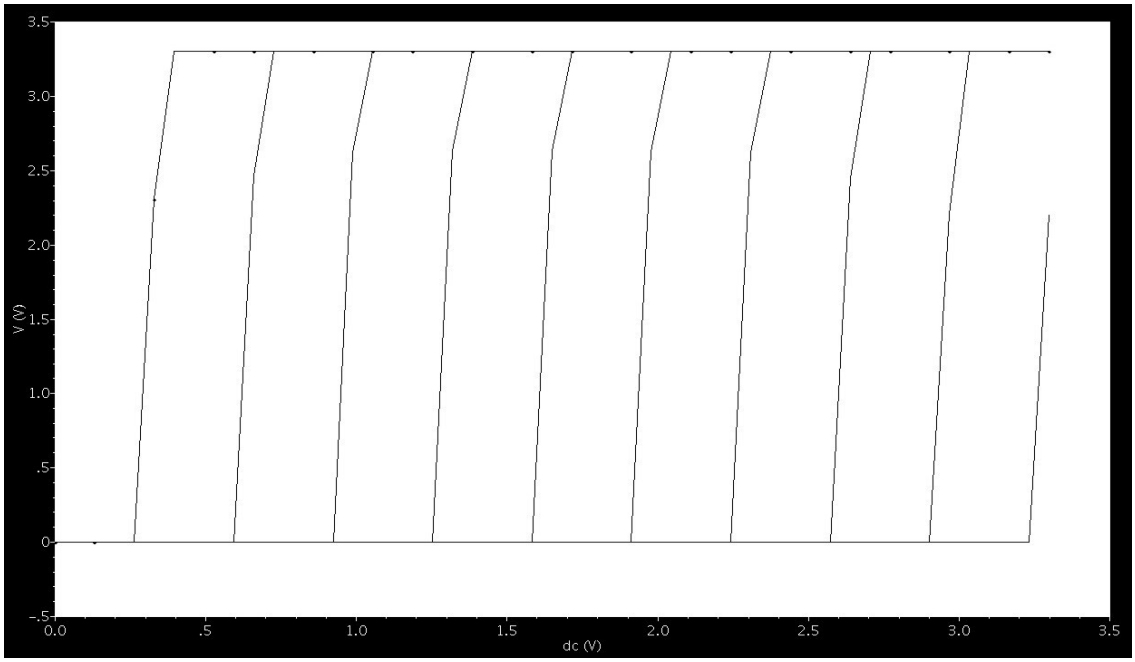

Figura 7.6: desempenho CC do comparador.

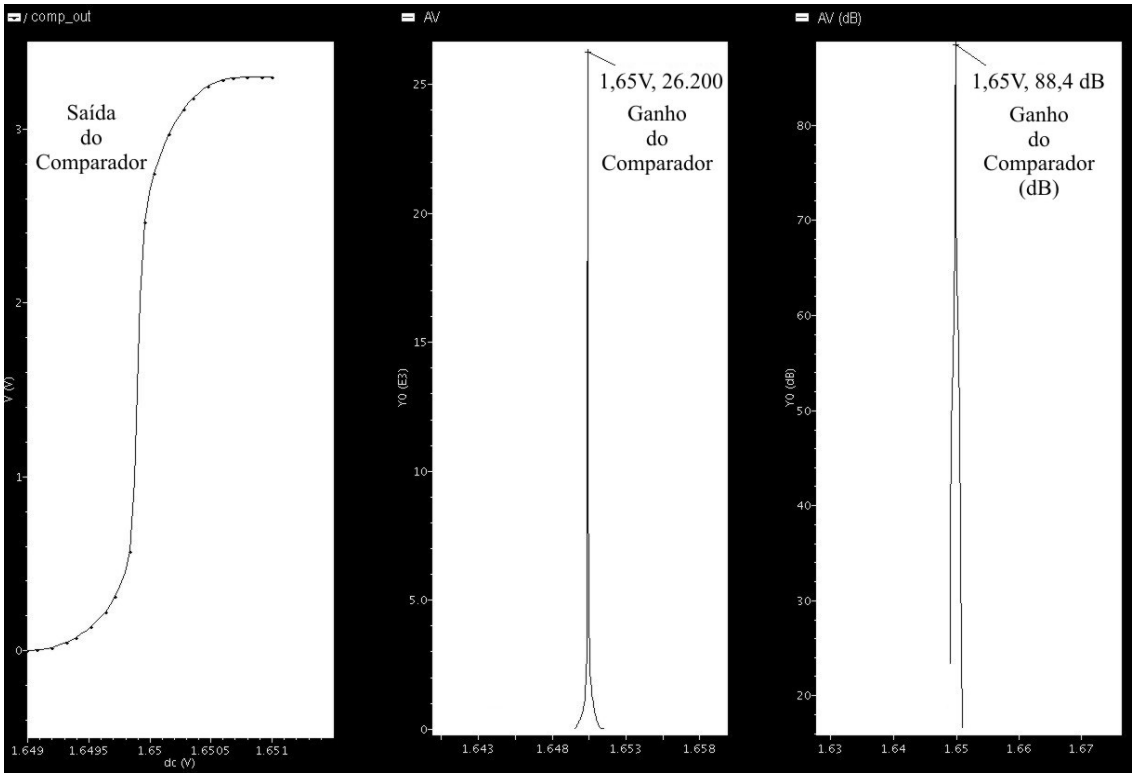

Figura 7.7: ganho do comparador.

A resposta ao transitório do comparador, mostrada na figura 7.8, foi obtida com *vn* = 1,650 V e *vp* um pulso variando de 1,600 a 1,700 V (50 mV *overdrive*) e largura de 10 ns. Na figura é possível verificar um atraso de 5 ns que corresponde, aproximadamente, ao tempo de resposta do comparador.

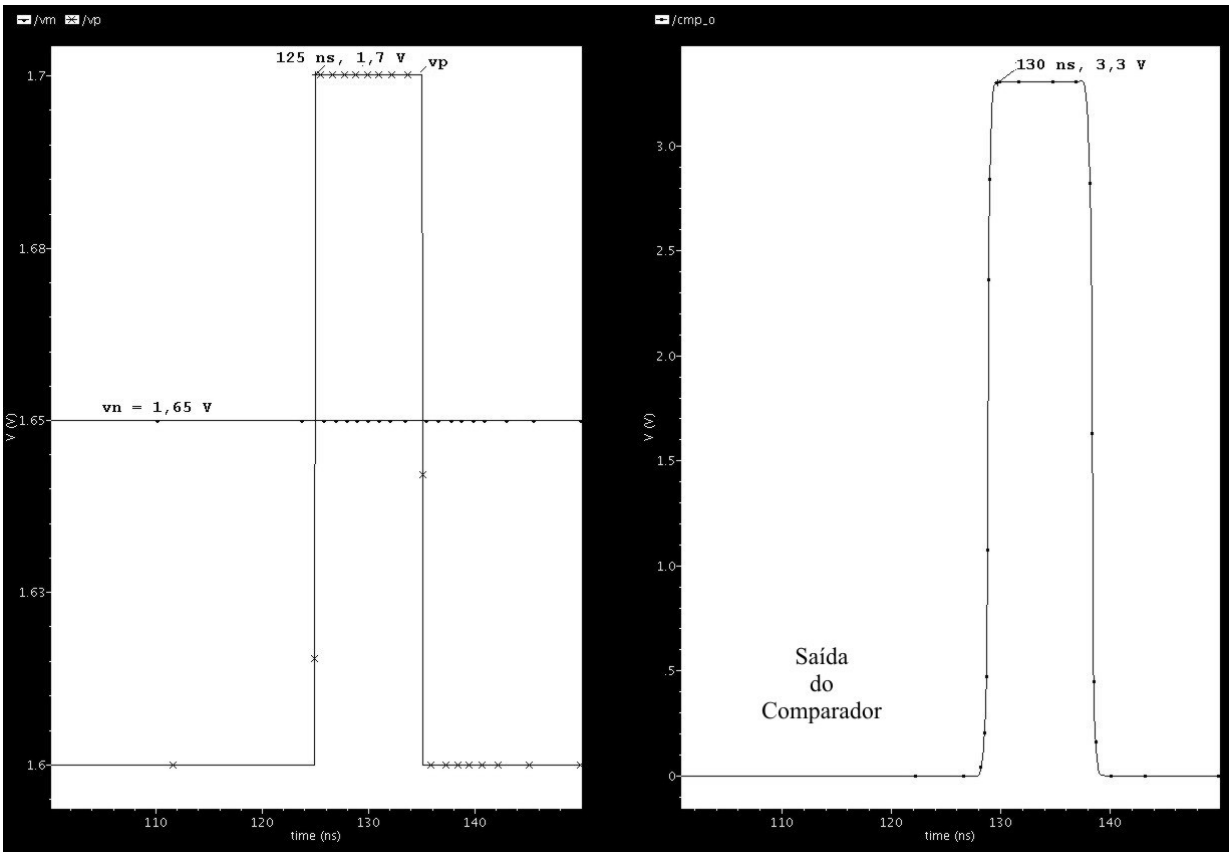

Figura 7.8: resposta transiente do comparador.

Para verificação do funcionamento do circuito, foram aplicadas inicialmente as tensões de entrada V_{IN} = 1,650 e 2,350 V, resultando nas formas de onda das figuras 7.9 e 7.10, respectivamente.

A figura 7.9 mostra uma palavra 100000000000_2 (2048_{10}), ou o equivalente a 1,650 V, indicando que, nesse caso, a conversão ocorreu sem erros. É possível verificar também, que a forma de onda correspondente ao nó *Vx* é semelhante às apresentadas nas figuras 7.2 e 7.3.

A figura 7.10 mostra uma palavra 101101100011_2 (2915_{10}), indicando que houve o erro de 1 LSB na conversão.

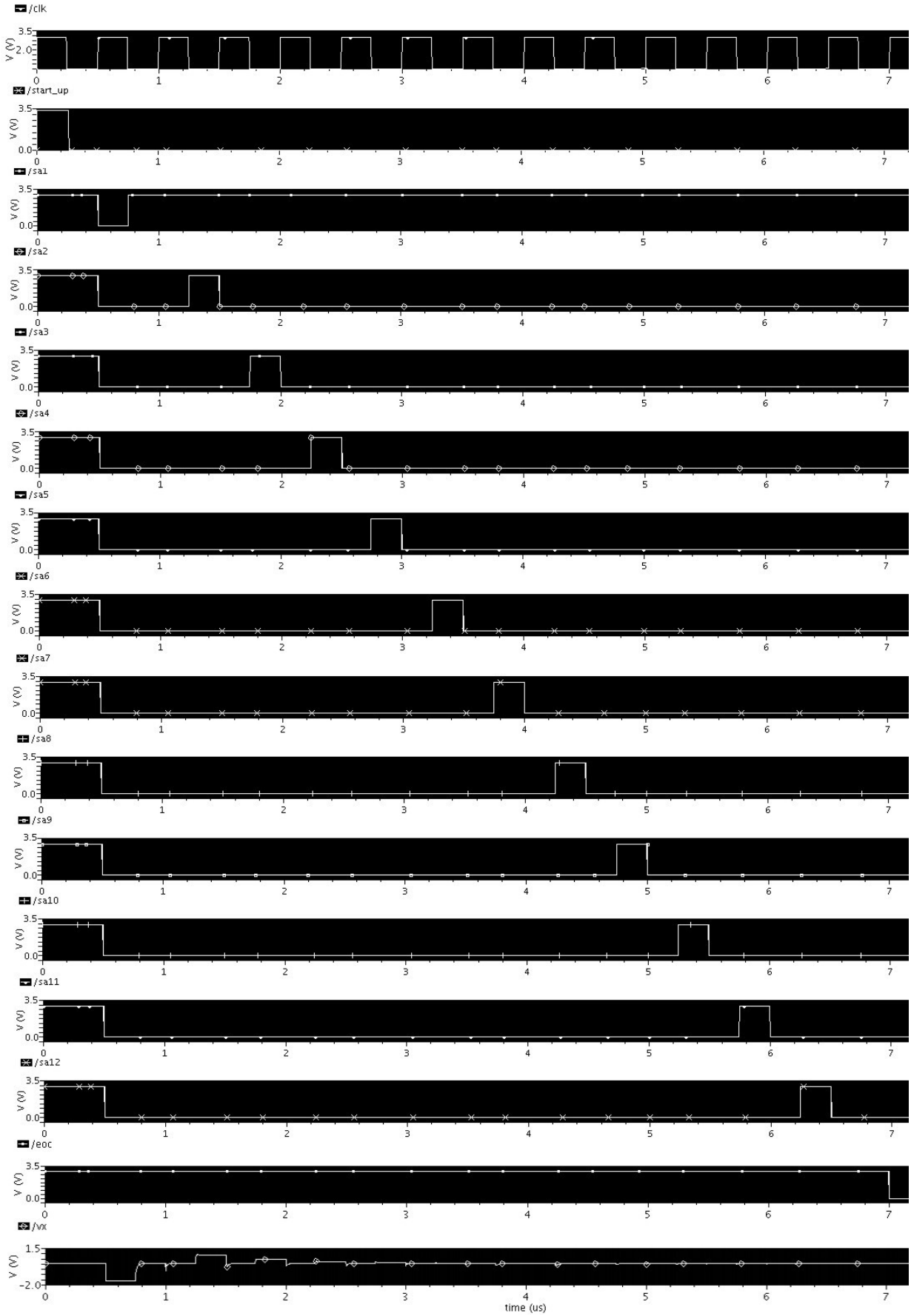

Figura 7.9: saída do conversor para o valor de entrada V_{IN} = 1,650 V.

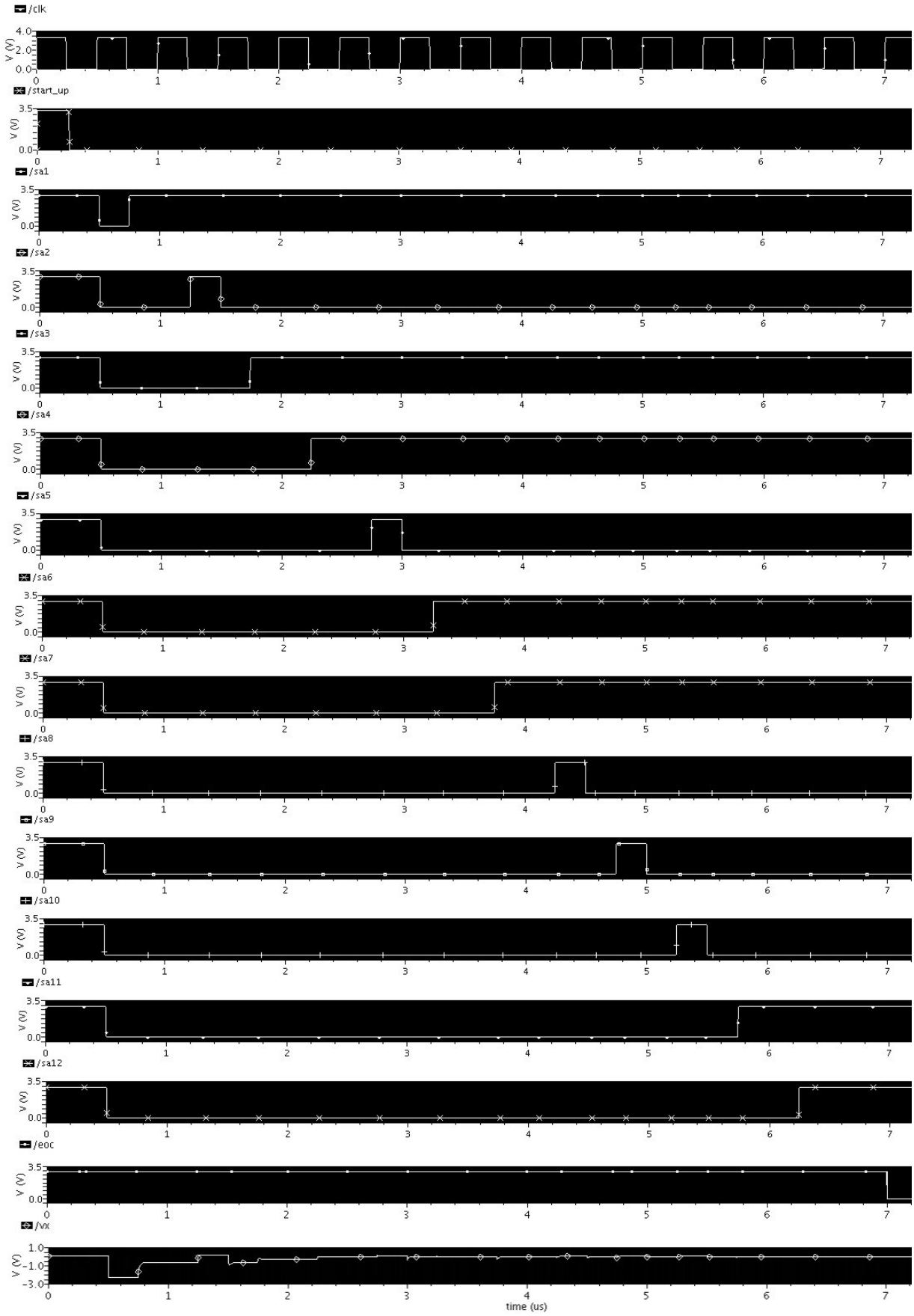

Figura 7.10: saída do conversor para o valor de entrada V_{IN} = 2,350 V.

Uma conversão completa ocorre em aproximadamente 7 μs, resultando em uma frequência de amostragem de 142 kSPS.

Os valores de tensão obtidos da entrada em rampa (treze amostras com V_{IN} variando de 0 a 3,3 V) mostrada na figura 7.5 para o modelo implementado em SystemC/SystemC-AMS foram aplicados à entrada do conversor para efeito de comparação entre os valores ideais (modelo) e os obtidos do CI. A figura 7.11 mostra os valores de saída das treze amostras para o conversor ideal (modelo) e para o conversor implementado (Spectre).

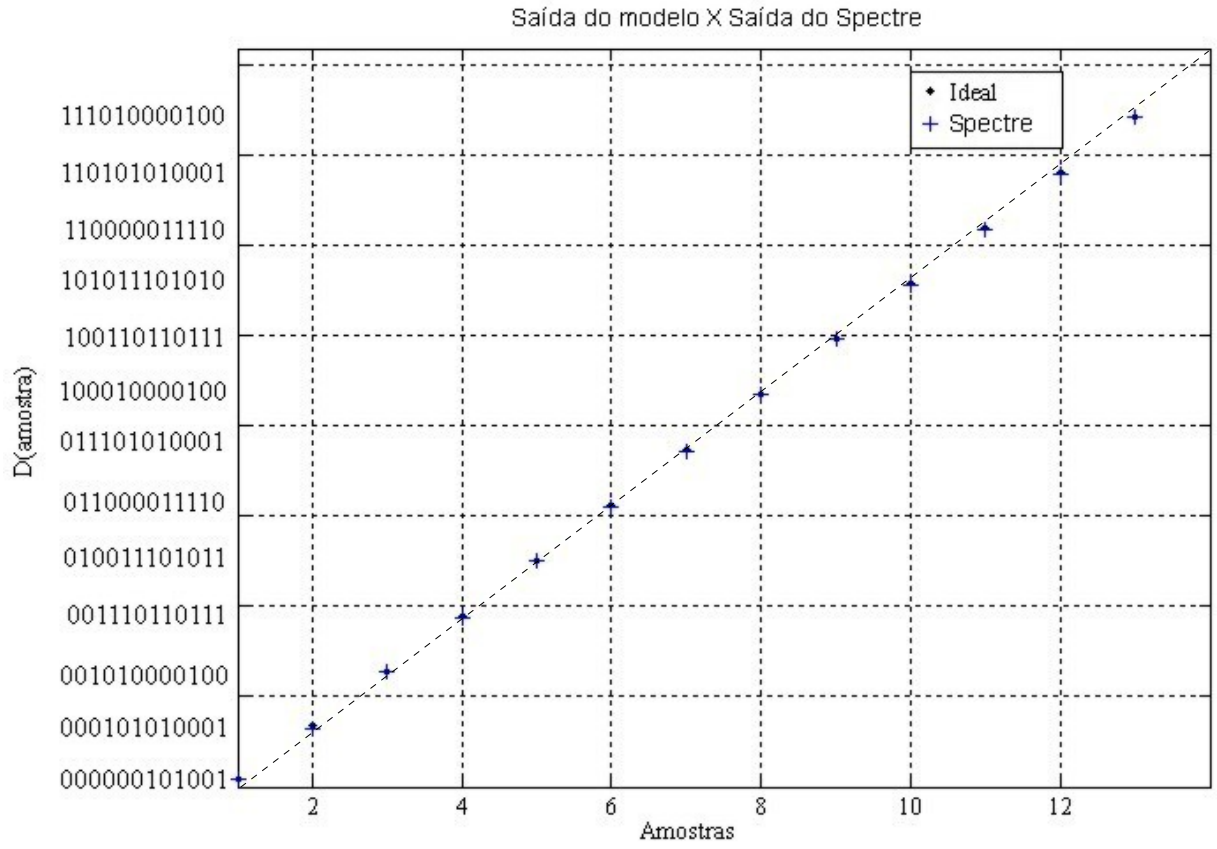

Figura 7.11: resposta do circuito mediante 13 amostras do sinal de entrada.

Foram medidos os erros de linearidade integral (INL) e diferencial (DNL), utilizando-se as amostras da figura 7.11, resultando em INL/DNL = -1,00 / +1,25 LSB, implicando em uma relação sinal-ruído (SNR) igual a 63,28 dB. Disso, o número efetivo de *bits* (ENOB) conseguido para o conversor foi de 10 *bits*. Os valores de INL/DNL informados correspondem

ao pior caso (máximos) e foram medidos na adjacência de cada amostra. Assim, o INL/DNL da amostra 100010000100_2 (2180_{10}), por exemplo, foi medido por meio das transições de 100010000011_2 (2179_{10}), e para 100010000101_2 (2181_{10}). Apesar de o número reduzido de amostras utilizadas, estas foram suficientes para sinalizar o correto funcionamento do conversor, uma vez que abrangeram toda a faixa de operação. O comparador é uma elemento chave para a redução ocorrida no valor do ENOB; podendo ser mitigada com a aplicação de melhorias no referido componente.

O que se verifica na figura 7.11 é que, apesar da influência de capacitâncias parasitas e da existência de erros de linearidade integral e diferencial, o CI projetado apresentou resultados próximos aos obtidos no modelo (ideal), o que demonstra que as técnicas aplicadas para diminuição dos problemas comuns aos conversores são de fato eficazes.

Seguramente, o componente que demandou um maior cuidado, revelando-se crítico no funcionamento do conversor, foi a chave analógica. As técnicas empregadas para redução dos efeitos de degradação no funcionamento das chaves produziram sensíveis melhorias nos resultados do conversor. Foram necessários numerosos testes para se conseguir o acerto necessário e o tipo de chave mais adequada a um determinado ponto do circuito. Por esse motivo, a chave conectada à saída do comparador e ao nó *Vx* foi implementada de maneira diferente da chave conectada às tensões de entrada e de referência. Bem assim, as chaves conectadas às placas inferiores dos capacitores são também diferentes das anteriores.

O consumo máximo medido durante uma conversão completa foi de 1,9 mW, sendo a porção analógica a demandante de quase a sua totalidade. É importante ressaltar que não foram introduzidos nesse conversor mecanismos ativos de entrada (ou operação) em modo de baixa energia (*power down mode*) e, ainda assim, a dissipação de potência se mostrou em níveis aceitáveis.

8 CONSIDERAÇÕES FINAIS

O presente trabalho mostrou a sequência de desenvolvimento de um ADC por aproximações sucessivas de 12 *bits*. O que se depreende desse projeto é que o desenvolvimento de sistemas de sinais mistos se mostra complexo, propenso a falhas e requer atenção, por parte do projetista, a inúmeros detalhes; o que desperta admiração pelos pioneiros desenvolvedores desses sistemas. Da concepção do sistema à criação do protótipo, há um longo caminho de refinamentos sucessivos, modificações e correções que demandam intensa atividade laboral que muitas vezes incorre em resultados que remetem o projetista ao reinício do processo.

A revisão bibliográfica obviamente se mostrou essencial ao desenvolvimento do conversor. Os princípios da conversão, as variantes físicas envolvidas no processo, as melhorias, os benefícios e malefícios ao se aplicar determinado bloco ou componente, bem como o resultado esperado de uma conversão advém da contribuição valorosa daqueles que se dedicaram à pesquisa, ao estudo e ao desenvolvimento desses artefatos essenciais ao mundo digital, os conversores A/D.

A metodologia preencheu lacunas e serviu de balizamento durante todo o desenvolvimento do conversor. Os modelos iniciais proporcionaram a verificação da viabilidade do projeto, bem como a produção de resultados que foram comparados ao que preconizava a bibliografia, a fim de corroborar ou reprovar. De fato, a concepção em alto nível de abstração permitiu o desenvolvimento relativamente rápido, assim como a correção de inconsistências ou falhas na produção dos resultados esperados. E, apesar de haver uma lacuna entre a transição do modelo e a implementação prática, não houve, sobretudo, desabono do método de desenvolvimento por esse permitir o entendimento dos princípios gerais do funcionamento do sistema e de servir de suporte para a criação do protótipo.

Durante o desenvolvimento do conversor, inúmeras decisões concernentes aos circuitos tiveram que ser tomadas. O formato de conversor apresentado na bibliografia não implica

necessariamente em um circuito pronto e funcional, ou seja, a aplicação não é direta. A implementação em si, mesmo a de um modelo ideal, envolve o surgimento de soluções que não estão contempladas na bibliografia. Desta forma, o tipo de chave a ser utilizada, o ganho requerido pelo comparador, o detalhamento dos circuitos que compõem o controle digital, dentre outros fatores, não estão explicitados, tampouco estão disponíveis para o desenvolvimento direto do conversor. Essas variantes tiveram que passar por um minucioso processo de implementação, teste e correção para que se conseguisse um funcionamento satisfatório.

Durante o processo de desenvolvimento, versões iniciais com um número menor de *bits* foram utilizadas para permitir o entendimento quanto ao funcionamento dos blocos, bem como do sistema como um todo. Uma vez verificado o atendimento aos requisitos funcionais, a versão final em 12 *bits* foi então produzida. Esse tipo de procedimento foi adotado em cada uma das fases do desenvolvimento, ou seja, tanto na criação dos modelos Simulink e SystemC, quanto na criação do protótipo (CI). Uma vantagem obtida foi a redução nos tempos de simulação e a rápida verificação da resposta do circuito.

Neste trabalho não se buscou o desenvolvimento do estado da arte em conversores A/D por aproximações sucessivas, mas o desenvolvimento de um produto inicial que permitisse o entendimento e a produção do conhecimento necessário ao refinamento e melhoria de futuras versões. Não obstante, como o próprio nome diz, o protótipo, ou primeiro exemplar, está destinado a teste e aperfeiçoamento.

Melhoramentos nesse projeto podem ser direcionados ao roteamento dos sinais na porção analógica, ao comparador, à diminuição da contribuição das capacitâncias parasitas, dentre outros. Além disso, a simples modificação do elemento de conversão unipolar para bipolar já expande a sua gama de aplicações, o que não implica necessariamente em alterações profundas na arquitetura utilizada.

O método de desenvolvimento empregado poderia ser melhorado por meio da inclusão

de linguagens que já se encontram contempladas nas ferramentas de desenvolvimento do circuito integrado como, por exemplo, o Verilog-AMS. Isso poderia trazer uma melhora na redução do tempo do processo de desenvolvimento como um todo.

Os pontos supracitados surgem como sugestão para continuidade ou surgimento de futuros trabalhos.

REFERÊNCIAS

(Allen, 2002) ALLEN, P. E.; HOLBERG, D. R. *CMOS Analog Circuit Design*. 2nd ed. New York: Oxford University Press, 2002.

(Baker, 2008) BAKER, J. R. *CMOS Circuit Design, Layout, and Simulation*. 2nd ed. Piscataway: Wiley-Interscience, 2008.

(Barna, 1973) BARNA, A.; PORAT, D. I. *Integrated circuits in digital electronics*. New York: John Wiley & Sons, 1973.

(Bennet, 1948) BENNET, W. R. Spectra of Quantized Signals. *Bell System Technical Journal*, Vol. 27, julho 1948, pp. 446-471.

(Bhasker, 2002) BHASKER, J. *A SystemC Primer*. Allentown: Star Galaxy Publishing, 2002.

(Black, 2004) BLACK, D. *SystemC From The Ground Up*. New York: Kluwer Academic Publishers, 2004.

(Brunvan, 2010) BRUNVAND, E. *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*. Boston: Addison-Wesley, 2010.

(Geiger, 1990) GEIGER, R. L.; ALLEN, P. E.; STRADER, N. R. *VLSI Design Techniques for Analog and Digital Circuits*. New York: McGraw-Hill, 1990.

(Gregorian, 1999) GREGORIAN, R. *Introduction to CMOS OP-AMPs and comparators*. New York: Wiley-Interscience, 1999.

(Grötker, 2002) GRÖTKER, T. *System Design with SystemC*. New York: Kluwer Academic Publishers, 2002.

(Gustavsson, 2002) GUSTAVSSON, M.; WIKNER, J. J.; TAN, N. N. *CMOS data converters for communications*. New York: Kluwer Academic Publishers, 2002.

(Hartley, 1928) HARTLEY, R. V. L. Transmission of Information. *Bell System Technical Journal*, Vol. 7, julho 1928, pp. 535-563.

(Hastings, 2001) HASTINGS, A. *The Art of Analog Layout*. Upper Saddle River: Prentice Hall, 2001.

(Hayes, 1999) HAYES, M. H. *Schaum's Outline: Digital Signal Processing*. New York: McGraw-Hill, 1999.

(Hnatek, 1976) HNATEK, E. R. *A user's handbook of D/A and A/D converters*. New York: John Wiley & Sons, 1976.

(Inose, 1962) INOSE, H.; YASUDA, Y.; MURAKAMI, J. A Telemetering System by Code Modulation: Δ-Σ Modulation. *IRE Transactions on Space Electronics Telemetry*, Vol. SET-8, setembro 1962, pp. 204-209.

(Inose, 1963) INOSE, H.; YASUDA, Y. A Unity Bit Coding Method by Negative Feedback. *Proceedings of the IEEE*, Vol. 51, novembro 1963, pp. 1524-1535.

(Johns, 1997) JOHNS, D.; MARTIN, K. *Analog Integrated Circuit Design*. New York: John Wiley & Sons, 1997.

(Kester, 2005) KESTER, W. *The Data Conversion Handbook*. Boston: Newnes, 2005.

(Kundert, 2010) KUNDERT, K. S. Principles of Top-Down Mixed-Signal Design. Disponível em http://www.designers-guide.org. Acesso em: 19 de julho de 2010.

(Malvino, 1997) MALVINO, A. P. *Electronic principles*. 4th ed. New York: Pearson Makron Books, 1997.

(Mathworks, 2010) MATHWORKS, Inc. Simulink®, Stateflow® e Simscape™. Disponível em http://www.mathworks.com. Acesso em: 21 de outubro de 2010.

(McCreary, 1975) MCCREARY, J. L.; GRAY, P. R. All-MOS Charge Redistribution Analog-to-Digital Conversion Techniques – Part I. *IEEE Journal of Solid State Circuits*, Vol. SC-10, No. 6, dezembro de 1975.

(Moscovici, 2001) MOSCOVICI, A. *High Speed A/D Converters: Understanding Data Converters Through SPICE*. New York: Kluwer Academic Publishers, 2001.

(Müller, 2003) MÜLLER, W. *SystemC Methodologies and Applications*. Dover: Kluwer Academic Publishers, 2003.

(Nyquist, 1924) NYQUIST, H. Certain Factors Affecting Telegraph Speed. *Bell System Technical Journal*, Vol. 3, abril 1924, pp. 324-346.

(Nyquist, 1928) NYQUIST, H. Certain Topics in Telegraph Transmission Theory. *A.I.E.E. Transactions*, Vol. 47, abril 1928, pp. 617-644.

(OSCI SystemC Lang Ref, 2003) Open SystemC Initiative. SystemC 2.0.1 Language Reference Manual Rev. 1.0 (2003). Disponível em http://www.systemc.org. Acesso em: 31 de janeiro de 2011.

(OSCI SystemC AMS Lang Ref, 2010) Open SystemC Initiative. SystemC AMS extensions Language Reference Manual (2010). Disponível em http://www.systemc-ams.org e http://www.systemc.org. Acesso em: 7 de fevereiro de 2011.

(OSCI SystemC User's Guide, 2010) Open SystemC Initiative. SystemC AMS extensions User's Guide (2010). Disponível em http://www.systemc-ams.org e http://www.systemc.org. Acesso em: 7 de fevereiro de 2011.

(Oppenheim, 2010) OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-Time Signal Processing*. 3rd ed. Upper Saddle River: Prentice Hall, 2010.

(Plassche, 2003) PLASSCHE, R. van de. *CMOS integrated analog-to-digital and digital-to-analog converters*. 2nd ed. Boston: Kluwer Academic Publishers, 2003.

(Razavi, 1995) RAZAVI, B. *Principles of Data Conversion System Design*. Piscataway: IEEE Press, 1995.

(Razavi, 2001) RAZAVI, B. *Design of Analog CMOS Integrated Circuits*. Boston: McGraw Hill, 2001.

(Shannon, 1949) SHANNON, C. E. Communication in the Presence of Noise. *Proceedings of the Institute of Radio Engineers (IRE)*, Vol. 37, No. 1, janeiro 1949, pp. 10-21.

(Weste, 2005) WESTE, N. H. E.; HARRIS, D. *CMOS VLSI Design: A Circuits and System Perspective*. 3rd ed. Boston: Pearson Education, 2005.