Universidade Federal de Itajubá – UNIFEI

***Campus* Theodomiro Carneiro Santiago**

Instituto de Ciências Tecnológicas – ICT

Engenharia da computação

Circuito digital para controle do fator de qualidade de um filtro passa-banda ativo sintonizável

Autor: Alef de Oliveira Santos

Orientador: Dr. Dean Bicudo Karolak

Coorientador: Dr. Paulo Márcio Moreira e Silva

Itabira, MG

30 de março de 2024

Alef de Oliveira Santos

Circuito digital para controle do fator de qualidade de um filtro passa-banda ativo sintonizável

Monografia submetida ao curso de graduação em Engenharia da computação da Universidade federal de Itajubá, como requisito parcial para obtenção do Título de Bacharel em Engenharia da computação.

Universidade Federal de Itajubá – UNIFEI

Campus Theodomiro Carneiro Santiago

Instituto de Ciências Tecnológicas – ICT

Orientador: Dr. Dean Bicudo Karolak

Coorientador: Dr. Paulo Márcio Moreira e Silva

Itabira, MG

30 de março de 2024

Alef de Oliveira Santos

Circuito digital para controle do fator de qualidade de um filtro passa-banda ativo sintonizável

Monografia submetida ao curso de graduação em Engenharia da computação da Universidade federal de Itajubá, como requisito parcial para obtenção do Título de Bacharel em Engenharia da computação.

Trabalho _____. Itabira, MG, 01 de dezembro de 2023:

Dr. Dean Bicudo Karolak
Orientador

Dr. Paulo Márcio Moreira e Silva
Co-orientador

Dr. Rodrigo Aparecido da Silva Braga
Membro da banca

Dr. Diogo Leonardo Ferreira da Silva
Membro da banca

Itabira, MG
30 de março de 2024

Dedico este trabalho àqueles que lutam
para popularizar a ciência.

Agradecimentos

Agradeço ao professor Rodrigo por despertar em mim o interesse sobre o tema de microeletrônica. Aos professores Dean e Paulo pela orientação e paciência em ensinar nestes anos de pesquisa. Ao Maderson e à Cristina pelo suporte e confiança. À Larissa pela lista de inúmeras contribuições, particularmente, fora deste trabalho.

Resumo

O controle do fator de qualidade (Q) em um filtro passa banda é uma maneira direta de controlar a seletividade deste filtro, e utilizando um circuito ativo é possível controlá-lo. Para controlar o Q de um sistema ressonante com um circuito digital utiliza-se neste trabalho técnicas de computação numérica para aproximação de funções não-lineares, uma vez que os parâmetros do filtro *on-chip* não são manipuláveis além da corrente que controla o Q. Assim, o objetivo deste trabalho é projetar um circuito digital sintetizável capaz de: receber como entrada o Q desejado e medido de um filtro passa banda e controlar digitalmente esse valor de Q do filtro. Além disso, três diferentes métodos numéricos de controle serão estudados e implementados para comparação e escolha do melhor método de convergência.

Palavras-chave: Fator Q. Controle digital. Métodos numéricos. RTL. Verilog.

Abstract

Tuning the quality factor (Q) in a bandpass filter is a direct way to control the filter's selectivity, and this goal can be achieved by using an active circuit. In order to control the Q of a resonant system with a digital circuit, this work uses numerical computing techniques to approximate non-linear functions, since the parameters of the *on-chip* filter are not manipulable beyond the current that controls the Q. Thus, the aim of this work is to design a synthesizable digital circuit capable of: receiving as input the desired and measured Q of a bandpass filter and digitally and digitally adjust its bias current to achieve a Q value close to the desired one. Therefore, the system architecture is basically composed by digital circuits to determine the measured filter Q value, to compare it with the desired value and tune it as close as possible to the desired value. Regarding the Q adjustment block, three different numerical control methods are studied and implemented in order to compare and choose the best convergence method.

Key-words: Q factor. Digital control. Numerical methods. RTL. Verilog.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

*

Lista de abreviaturas e siglas

A/D	Analógico para Digital (conversão)
CI	Circuito Integrado
CC	*Clock Cycle*
D/A	Digital para Analógico (conversão)
DAC	*Digital-to-analog converter*
DFT	*Design for testability*
DUT	*Device Under Test*
EDA	*Electronic design automation*
GDSII	*Graphic Design System* II
HDL	*Hardware Description Language*
ITC	*Iterations-to-converge*
RF	Rádio-Frequência/*Radio-Frequency*
STA	*Static timing analysis*
TB	*Testbench(es)*
UVM	*Universal Verification Methodology*

Lista de símbolos

α	Alfa
Δ	Delta
β	Beta
ε	Épsilon (erro)
Q	Fator de Qualidade
$\hat{Q}$	Fator de qualidade aproximado por algoritmo
Q_d	Fator de qualidade desejado
Q_m	Fator de qualidade medido
γ	Gama
$\leftarrow$	Atribuição em algoritmos (recebe)
$\vee$	Disjunção
$\wedge$	Conjunção

Sumário

Parte I

Introdução

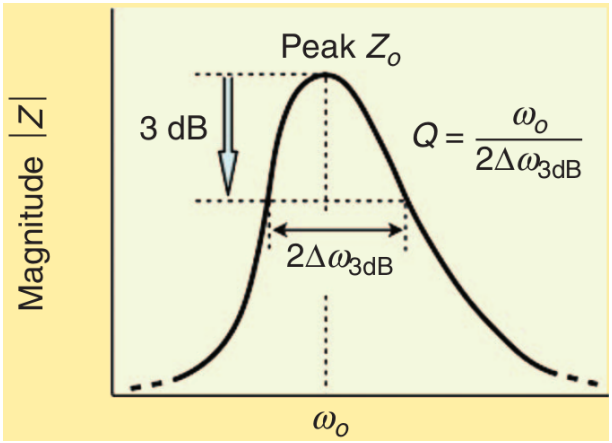
1 Introdução

Este trabalho trata de um sistema eletrônico que recebe e controla o fator de qualidade (Q) de um circuito eletrônico ressonante. O circuito proposto utiliza-se de técnicas de computação numérica para controlar o fator de qualidade através de uma corrente de referência injetada no sistema ressonante. Neste trabalho serão realizados circuitos digitais periféricos para a determinação do valor de Q medido de um filtro passa banda ativo, bem como, um circuito digital para controle e aproximação do Q desejado. Em relação ao controle e aproximação de Q, serão comparados os métodos numéricos da Bisseção, Secantes e Secantes com seleção de intervalo implementados. O sistema digital é projetado e implementado em Verilog tendo em mente a posterior fabricação em silício. Para efeitos de estudo e desenvolvimento, este projeto em ASIC utiliza uma tecnologia GPDK de 45nm.

1.1 Contexto e justificativa

A caracterização fator de qualidade (Q) é um ponto de partida para projetos de circuitos integrados de rádio-frequência de alto desempenho [1]. Ele está relacionado com a largura de banda de um filtro passa-banda e, por vezes, os projetistas de filtros necessitam de uma largura de banda estreita (isto é, alto Q) e em outras, uma banda maior (baixo Q) como ilustra a Figura 1:

Figura 1 – Q *versus* largura de banda

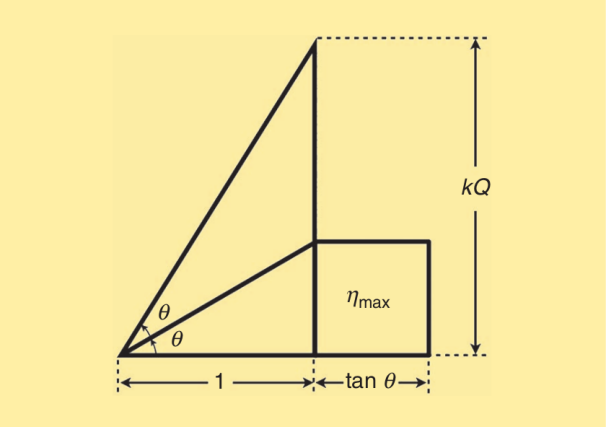


Fonte: [1]

Ainda de acordo com Ohira[1], o fator Q também está relacionado com o teorema da máxima transferência de potência em circuitos de RF. Para entregar o máximo de

potência de uma fonte para uma carga através de uma rede, um circuito de casamento de impedâncias é usado para alcançar a maior transferência de potência. A Figura 2 ilustra como o Q interfere na eficiência de um circuito:

Figura 2 – Q *versus* eficiência



Fonte: [1]

Num circuito com uma carga estática, basta calcular um Q que realize o casamento de impedâncias para a maior transferência de potência. Num circuito onde a carga é variável, um Q fixo não fornece o casamento de impedâncias necessário para a máxima transferência de potência em todos os cenários de carregamento, assim reduzindo a eficiência do circuito.

De fato Liao, Tan e Wang[2] retomam em seu trabalho as proposições de Ohira[1] e projetam uma rede de casamentos de impedância baseada no fator de qualidade, onde o mesmo é ajustado a fim de casar impedâncias de rede com uma carga variável. Chung[3] realiza um trabalho com a mesma ideia de controlar o Q usando outras técnicas para selecionar o valor ótimo.

Desta forma, fica evidente a necessidade de controlar o Q de um circuito eletrônico, tanto no contexto deste trabalho quanto em outros trabalhos com pouca similaridade, mas com a mesma necessidade de Q controlável/variável.

Em relação às diferentes implementações de circuitos eletrônicos para controle do Q, pode-se distingui-los em dois grupos, os circuitos analógicos e os digitais. Os circuitos analógicos têm a vantagem de ser mais simples, mais rápidos, como apresentado em [3]. Entretanto, eles não apresentam a mesma versatilidade e configurabilidade proposta por um sistema digital. Já os circuitos digitais, além de mais versáteis, apresentam uma implementação física mais simples com o uso de *standard-cells* para a construção de *layouts*, sendo inclusive, altamente assistidas por ferramentas de EDA pela característica programática. Em conjunto com a maior versatilidade, o circuito digital poderia ser estendido em aplicações onde o controle do Q deve ter alta precisão, uma vez que o cir-

cuito digital pode ser replicado com maior confiabilidade, portabilidade e reprodutibilidade.

Dessa forma, julga-se pertinente projetar este circuito eletrônico digital capaz de controlar o Q, para que seja possível obter circuitos mais versáteis e com aplicações mais amplas com menor complexidade em projeto analógico, além de promover inovação no desenvolvimento de circuitos para computação numérica.

2 Objetivos

Os objetivos principais deste trabalho para o TCC1 são, principalmente o fluxo de front-end, compreendido por:

1. Projetar a arquitetura capaz de controlar o fator de qualidade do circuito;
2. Codificar os blocos do sistema projetado em Verilog;
3. Comparar o desempenho dos métodos de controle prototipados *standalone*;
4. Validar a funcionalidade blocos projetados através de *testbenches* em Verilog/SystemVerilog.

Após finalizado o fluxo de front-end, seleciona-se o método de convergência com melhor desempenho com relação à todo o sistema e inicia-se o processo de verificação do sistema completo seguido do fluxo de back-end no TCC2, compreendido por:

1. Integrar e coordenar a operação dos blocos como um sistema completo;
2. Realizar a síntese lógica em RTL;
3. Simular o circuito sintetizado em RTL e checar a equivalência lógica;
4. Realizar etapas de posicionamento e roteamento;
5. Analisar o consumo;
6. Analisar desempenho do sistema por temporização estática (STA);
7. Construir layout.

Parte II

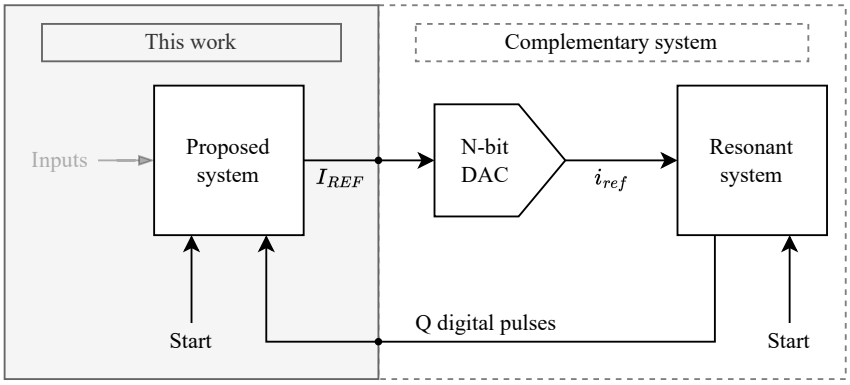
Desenvolvimento

3 Referencial teórico

3.1 Interface com o sistema completo e arquitetura simplificada

Antes de descrever a arquitetura completa desenvolvida neste trabalho, é necessário retomar que o sistema de controle é complementar à um sistema pré-existente composto pelo próprio sistema ressonante[1] (dentre outras estruturas) e um DAC. Assim sendo, algumas escolhas de arquitetura na verdade são requisitos de integração do projeto e comunicação. A Figura 3 ilustra a arquitetura simplificada do sistema completo.

Figura 3 – Arquitetura simplificada do sistema completo



Neste diagrama de blocos há 3 blocos principais. Cada bloco e sua respectiva responsabilidade é:

1. *Proposed System*: o próprio sistema digital desenvolvido responsável por controlar o Q fazendo uso dos métodos numéricos descritos na seção 3.2 – Métodos numéricos para aproximações de funções não-lineares. Também responsável por outras operações relevantes ao ajuste do Q, bem como a determinação do valor máximo alcançável de Q, recepção do valor de Q dentre outras estruturas. Este bloco é detalhado na seção 4.2 – Arquitetura do sistema.

2. *Resonant System*: Sistema ressonante controlado por corrente (i_{ref}), o qual apresenta um fator de qualidade Q proporcional à corrente de polarização e disponibiliza a medição do Q em um sinal serial *Q digital pulses*.

3. *Current Control* (N-bit DAC): Conversor D/A que recebe a palavra digital I_{REF} e a converte em um valor analógico i_{ref} que controla o oscilador do sistema ressonante.

[1] Baseado em [4]

Os blocos supracitados, com suas entradas, saídas, têm suas funcionalidades elencadas na Tabela 1:

Tabela 1 – Sinais, respectivos tipos e funcionalidades por bloco

Bloco	**Sinal**	**Tipo**	**Função referente ao bloco**
Proposed system	Start	Digital [1 bit]	Sinalizar que o oscilador enviará pulsos correspondentes ao valor de Q
Proposed system	I_{REF}	Digital [10 bits]	Enviar valor digital de corrente de referência para controlar o Q
N-bit DAC	I_{REF}	Digital [10 bits]	Receber valor digital de corrente de referência para conversão D/A
N-bit DAC	i_{ref}	Analógico	Equivalente analógico convertido de I_{REF}
Resonant system	Start	Digital [1 bit]	Indicar que o sistema pode injetar a corrente no oscilador LC interno
Resonant system	*Q digital pulses*	Digital Serial [1 bit]	Valor de Q medido e convertido em um trem de pulsos

As entradas descritas na Tabela 1 e Figura 3 são suficientes para a comunicação completa do sistema, detalhada na seção 4.2 – Arquitetura do sistema. As outras entradas relevantes do sistema proposto (representadas na Figura 3 com baixa opacidade) também serão elencadas no detalhamento da arquitetura na seção relativa.

3.2 Métodos numéricos para aproximações de funções não-lineares

De acordo com 5, há vários métodos para solução aproximada de funções não-lineares. Esses métodos visam encontrar o valor x de uma função tal $f(x) \approx 0$. Em outras palavras, os métodos de solução aproximada visam encontrar encontrar raízes para uma função à partir de técnicas de aproximação.

Para a busca de um valor em uma função, ou seja, $f(x) \approx a$ o mais comum é que, se use algum tipo de interpolação. De fato, interpolação em hardware e principalmente em FPGA é uma tarefa amplamente implementada e pesquisada onde o volume de dados é extremamente alto pelo paralelismo intrínseco das FPGA's como fizeram [6] e [7] em seus trabalhos. Entretanto, neste trabalho a interpolação não é uma opção que teria uma boa razão entre desempenho e precisão pelas operações feitas com números decimais, o que em hardware, não é trivial. Então, estuda-se a possibilidade de adaptar os métodos de encontrar raízes para que estes se tornem métodos de busca de valores mantendo as mesmas características de convergência e desempenho.

3.2.1 Método da bisseção

O método da bisseção é um dos mais antigos e simples já desenvolvido para a obtenção de raízes. Este método é também chamado de busca binária e baseia-se no teorema do valor intermediário [5]. Supondo que f seja uma função contínua definida no intervalo fechado [a, b], com $f(a)$ e $f(b)$ tendo sinais opostos, então existe um valor x entre a, b no qual $f(x) = 0$ pois a função teria, obrigatoriamente que cruzar o eixo x em algum momento, produzindo assim uma raiz.

O método consiste em particionar o intervalo $[a, b]$ sucessivamente até achar uma raiz. Supondo que c_i seja o ponto médio, a_i, b_i os pontos iniciais e finais a cada iteração i. Pode-se escrever c_i na Equação 3.1:

$$c_0 = a_0 + \frac{b_0 - a_0}{2} = \frac{a_0 + b_0}{2}. \tag{3.1}$$

À partir do cálculo de c_0 na primeira iteração com os valores a_0 e b_0, existem duas situações distintas para a busca de raiz:

1. Se $f(c) = 0$, então c é raiz.
2. Se $f(c) \neq 0$ então c não é raiz e tem o mesmo sinal de $f(a_0)$ ou de $f(b_0)$
   a) Caso $f(c)$ e $f(a_0)$ tenham o mesmo sinal, então a raiz está entre [b,c]
   b) Caso $f(c)$ e $f(b_0)$ tenham o mesmo sinal, então a raiz está entre [a,c]

Se o item 1 não é satisfeito, reparte-se o intervalo conforme o item 2 e recalcula-se c_{i+1} com os valores atualizados de a e b para a iteração seguinte. Com os passos elencados nos itens 1 e 2, constrói-se o algoritmo de acordo com Burden e Faires[5] no Algoritmo 1:

Algoritmo 1 Método clássico da bisseção

Input: Pontos iniciais e finais a, b ; Tolerância TOL; Número máximo de iterações N_0
Output: Solução c para $f(c) = 0$ ou erro

```
1:  i ← 1                                   ▷ flag de iterações é inicializada
2:  F_A ← f(a)                              ▷ Variável F_A recebe o valor da função no ponto a
3:  while i ≤ N_0 do
4:      c ← a + (b−a)/2
5:      F_C ← f(c)
6:      if F_C = 0 ∨ ((b−a)/2 < TOL) then
7:          return(c)
8:      end if
9:      i ← i + 1
10:     if F_A · F_C > 0 then               ▷ F_A e F_C têm o mesmo sinal
11:         a ← c
12:         F_A ← F_C
13:     else                                ▷ F_A e F_C têm sinais diferentes
14:         b ← c
15:     end if
16: end while
17: return "Method failed"
```

Fonte: Adaptado de [5]

O método da bisseção é particularmente interessante para a implementação em hardware pois além das operações simples, a divisão por 2 pode ser abstraída por um *shift*

aritmético – operação realizada com certa "facilidade" em hardware se comparada com uma divisão tradicional.

3.2.2 Método das secantes

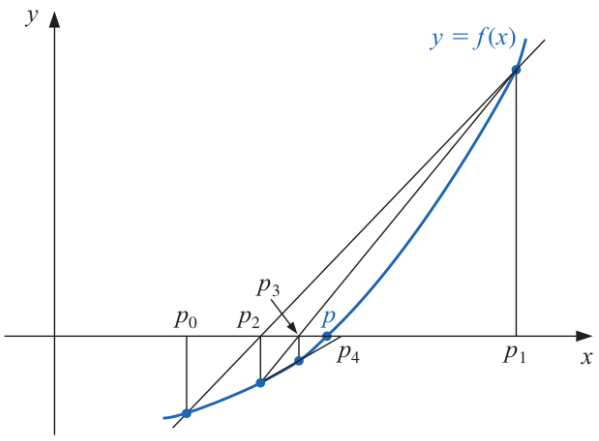
O método das secantes é um método de convergência baseado no método de Newton. Burden e Faires[5] citam que apesar do método de Newton ser um dos mais rápidos (com relação à ordem de convergência), conta com uma dificuldade: o cálculo da derivada da função à cada iteração. Calcular $f'(x)$ costuma ser mais trabalhoso e demorado do que calcular apenas o valor de $f(x)$.

O método das secantes faz uma aproximação da derivada justificando que o ponto c que se aproxima da raiz varia pouco à cada iteração (sub-escrito i): Essa simplificação é descrita na Equação 3.2

$$f'(c_{i-1}) \approx \frac{f(c_{i-1}) - f(c_{i-2})}{c_{i-1} - c_{i-2}} \quad \longleftrightarrow \quad c_i = c_{i-1} - \frac{f(c_{i-1}) - f(c_{i-2})}{c_{i-1} - c_{i-2}}. \tag{3.2}$$

Fazendo esta adaptação ao método de Newton é produzido o método das secantes, pois ao invés de traçar uma reta tangente, traçam-se retas secantes sucessivamente até atingir uma raiz, como ilustra a Figura 4:

Figura 4 – Representação gráfica do método das secantes



Fonte: [5]

Sintetizando o comportamento descrito com base no proposto por Burden e Faires[5] no Algoritmo 2:

Algoritmo 2 Método clássico das secantes

Input: Pontos iniciais e finais a, b ; Tolerância TOL; Número máximo de iterações N_0
Output: Solução c para $f(c) = 0$ ou erro

1: $F_A \leftarrow f(a)$ ▷ F_A recebe o valor da função no ponto
2: $F_B \leftarrow f(b)$
3: $i \leftarrow 2$ ▷ *flag* de iteração é inicializada
4: **while** $i \leq N_0$ **do**
5: $c \leftarrow b - F_B \cdot \frac{b-a}{F_B - F_A}$ ▷ Calcula o ponto de intersecção com a abcissa
6: $F_C \leftarrow f(c)$
7: **if** $|c - b| <$ TOL **then**
8: **return**(c)
9: **end if**
10: $i \leftarrow i + 1$
11: $a \leftarrow b$
12: $b \leftarrow c$
13: $F_A \leftarrow F_B$
14: $F_B \leftarrow f(c)$ ▷ Novo cálculo para a próxima reta secante
15: **end while**
16: **return** "Method failed"

Fonte: Adaptado de [5]

O método das secantes torna-se uma solução razoável no cenário de implementação em software e hardware onde não é possível calcular uma derivada analítica da função avaliada, como propõe o método de Newton. Uma vez que a secante oferece uma aproximação adequada da derivada no ponto por iteração, o ponto calculado caminha em direção à raiz.

3.2.3 Método das secantes com seleção de intervalo

O terceiro método estudado é na verdade uma adaptação do método das secantes. Este terceiro método, chamado de "Secante modificada" ou "Secante com seleção de intervalos" é apenas uma forma mais eficiente de selecionar o intervalo de busca baseando-se nas propriedades da curva de $Q \times I_{REF}$. A motivação para a formalização deste terceiro método veio enquanto testava-se o algoritmo desenvolvido do método das secantes e observou-se alta sensibilidade às condições iniciais. Algo que aconteceu de forma menos expressiva comparando-se com o método da bisseção.

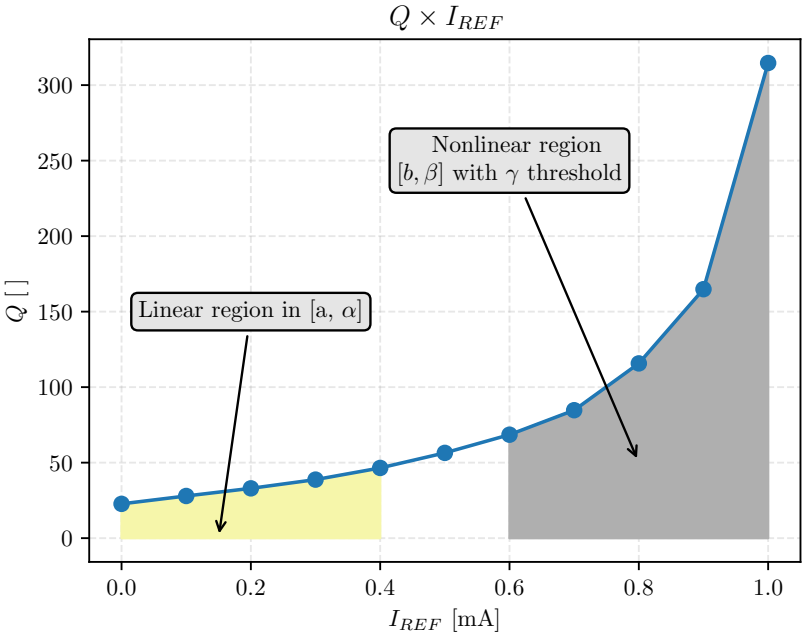
O método da secante com seleção de intervalo foi considerado uma formalização à parte ao invés de somente uma adaptação, pois como elencado por Burden e Faires[5], há métodos que seguem exatamente a mesma ideia de outros já formalizados e são considerados outro método por melhorarem a razão de convergência.

Alguns exemplos de métodos que aplicam alguma modificação em um método clássico são os métodos de Aitken Δ^2 e Steffensen, que visam construir uma sequência c_n que convirja mais rapidamente modificando o cálculo das secantes. Outros exemplos

tomam como base propriedades da função a ser aproximada (se é um polinômio, se há zeros complexos, etc) e reduzem a quantidade de iterações com base na especialização do método.

Dessa forma, introduz-se uma seleção de intervalos no método das secantes à partir da observação do comportamento da curva de $Q \times I_{REF}$ dando destaque às suas duas regiões na Figura 5.

Figura 5 – $Q \times I_{REF}$ com destaque para as regiões linear e não linear



Da Figura 5 dá-se destaque para duas regiões: Uma onde o Q varia de forma aproximadamente linear com o aumento de I_{REF} (região sombreada de amarelo entre 0 e 0.4mA) e outra onde o Q varia de forma mais abrupta se comparada com a região linear. Com este comportamento pode-se particionar a busca em dois intervalos distintos, um entre [a, α] e outro entre [β, b].

Existe uma faixa intermediária $[\alpha, \beta]$ que aparenta inicialmente não integrar a busca. De fato na primeira iteração não integra, mas, da segunda em diante, quando é calculado o ponto c, esta faixa passa a integrar o intervalo de busca. Essa faixa intermediária é ajustável de acordo com os parâmetros $[\alpha, \beta]$. O parâmetro γ ajusta o *threshold* da busca. A partir de $Q_d > \gamma$ é usada a região de busca $[b, \beta]$. Caso contrário usa-se $[a, \alpha]$.

4 Metodologia

4.1 Adaptação de métodos numéricos para busca de valores

Como citado anteriormente e descrito no Capítulo 3 – Referencial teórico, os métodos numéricos para aproximação de funções não-lineares são projetados originalmente para encontrar raízes. Entretanto, a adaptação para a busca de valores pode ser feita mantendo as características originais dos métodos modificando a maneira como alguns cálculos são feitos e suas condições de parada.

A primeira modificação feita é a inserção da variável a ser buscada (Q_d). Este é o valor de Q que o usuário almeja obter. Outra modificação é que agora o algoritmo não controla internamente o número de iterações[1] i.

A inicialização dos valores para o intervalo de busca também é feita de forma diferente pela condição de instabilidade da curva $Q \times I_{REF}$ como mencionado no Capítulo 3. O limite inferior de corrente sempre será 0mA enquanto o superior é passado como parâmetro para o algoritmo, sendo este I_{Stable}.

Outra modificação relevante é que o algoritmo agora se assemelha mais à um procedimento do que de uma função, pois não retorna valores, apenas retém os valores de corrente. Essa modificação foi implementada para alinhar o comportamento do algoritmo tanto em hardware quanto em software.

As subseções subseção 4.1.1 a subseção 4.1.3 descrevem as adaptações feitas para a busca de valores. É introduzida também a corrente de referência – I_{REF} para ilustrar chamadas ao bloco de medição.

4.1.1 Método da bisecção

No método da bisseção, as modificações mais importantes são a condição de parada e a seleção do próximo valor. No algoritmo original, é usado o teorema do valor intermediário para seleção do próximo intervalo [a,b] de busca [5]. Como na busca de valores pode não haver a troca de sinal, substitui-se o retorno em caso de $f(c) \approx 0 \vee f(c) < \text{TOL}$ pela diferença entre o Q medido e o desejado. Caso essa diferença seja menor que a tolerância, considera-se que o algoritmo convergiu.

Em caso de ainda não ter alcançado a convergência, a seleção dos novos valores é feita de forma similar mas sem usar o teorema do valor intermediário: caso o erro (ε) seja

[1] Nos testes do Capítulo 5 é usada uma variável auxiliar MAX_ITER apenas para restringir o testbench.

negativo, indica que o valor procurado está entre o intervalo [c, b]. Caso contrário está entre [a, c]. A transcrição do algoritmo adaptado desenvolvido é visível no Algoritmo 3.

Algoritmo 3 Método adaptado da bisseção

Input: TOL, Q_d, Q_m, I_{Stable}
Output: $I_{REF} \mid \varepsilon \leq \text{TOL}$

1: $a \leftarrow 0$
2: $b \leftarrow I_{Stable}$
3: Converged $\leftarrow$ **False**
4: **while** not Converged **do**
5: $\quad c \leftarrow \frac{a+b}{2}$
6: $\quad I_{REF} \leftarrow c$
7: $\quad Q_m \leftarrow Q_m\,(I_{REF})$ ▷ Realiza-se uma medição do Q com o novo valor de I_{REF}
8: $\quad \varepsilon \leftarrow Q_m - Q_d$
9: $\quad$ **if** $\varepsilon \leq$ TOL **then** Converged $\leftarrow$ **True**
10: $\quad$ **else**
11: $\quad\quad$ **if** $\varepsilon > 0$ **then**
12: $\quad\quad\quad a \leftarrow c$
13: $\quad\quad$ **end if**
14: $\quad\quad$ **if** $\varepsilon < 0$ **then**
15: $\quad\quad\quad b \leftarrow c$
16: $\quad\quad$ **end if**
17: $\quad$ **end if**
18: **end while**

A atribuição feita na linha 7 do Algoritmo 3 é feita apenas para simbolizar que há uma atualização do valor de Q_m assim que um novo I_{REF} é atribuído na linha anterior. Foi escolhido representar essa atribuição, pois em software ela é somente uma chamada de função que acontece instantaneamente, mas em hardware é uma operação de medição que requer alguns ciclos de *clock*, portanto, não é uma atribuição instantânea.

O método mantém as mesmas características de convergência do original, uma vez que foram modificadas apenas operações atômicas, mantendo sua ordem de convergência linear.

4.1.2 Método das secantes

Para o método das secantes, a estratégia de encontrar o próximo ponto c que intersecciona o eixo x é simples: translada-se a curva com o valor Q_d. Em outras palavras, c é calculado da mesma forma e diminui-se o valor de Q_d para que a raiz da curva seja $f(c) = Q_d$ ao invés de zero.

A atribuição de c é repartida em duas instruções diferentes. Primeiro calcula-se a inclinação da reta secante (Linha 6 do Algoritmo 4) e depois calcula-se o ponto de intersecção com a reta transladado pelo valor de Q desejado (Linha 8 do Algoritmo 4).

As instruções foram separadas pois percebe-se que os valores da inclinação podem pequenos, resultando em divisão por zero e potenciais estouros de memória em hardware. Verificando o valor da variável antes de calcular e atribuir à c, pode-se evitar valores indefinidos.

Para a condição de convergência basta calcular se o erro (ε) entre o valor medido e o desejado está aceitável conforme a tolerância escolhida. Neste caso é calculado o valor absoluto pois a posterior atribuição dos novos intervalos não necessita do sinal de ε como no Método adaptado da bisseção. O método adaptado desenvolvido é visível no Algoritmo 4.

Algoritmo 4 Método adaptado das secantes

Input: TOL, Q_d, Q_m, I_{Stable}
Output: $I_{REF} \mid \varepsilon \leq \text{TOL}$

1: $a \leftarrow 0$
2: $b \leftarrow I_{Stable}$
3: Converged $\leftarrow$ **False**
4: **while** not Converged **do**
5:
6: slope $\leftarrow \dfrac{Q_m(b) - Q_m(a)}{b - a}$ ▷ $Q_m(a), Q_m(b)$ são novas medições
7:
8: $c \leftarrow b - \dfrac{Q_m(b) - Q_d}{\text{slope}}$
9: $I_{REF} \leftarrow c$
10: $Q_m \leftarrow Q_m\,(I_{REF})$ ▷ Medição do Q com o novo valor de I_{REF}
11: $\varepsilon \leftarrow |Q_m - Q_d|$
12: **if** $\varepsilon \leq$ TOL **then** Converged $\leftarrow$ **True**
13: **else**
14: $a \leftarrow b$
15: $b \leftarrow c$
16: **end if**
17: **end while**

Diferentemente da busca por raiz, na busca por valores não é necessário que os valores de medição fossem reatribuídos em caso de não-convergência, como faz o Algoritmo 2 nas linhas 12-13.

Como comentado no próprio código, $Q_m(a)$ é uma nova chamada de medição com $I_{REF} = a$. Nota-se que no método das secantes há três chamadas desse tipo para o bloco de medição em cada iteração ($Q_m(a), Q_m(b), Q_m(c)$). Como a operação de medição é a operação que mais leva ciclos de *clock* para completar, considera-se um ponto de atenção ao comparar-se o desempenho dos algoritmos desenvolvidos em HDL.

4.1.3 Método das secantes com seleção de intervalo

O método das secantes com seleção de intervalo segue as mesmas considerações feitas adaptando o Algoritmo 2 para o Algoritmo 4. Durante a execução, a determinação dos valores de coeficientes (α, β, γ) que selecionam a região de intervalo é feita com base na curva de $Q \times I_{REF}$ conforme mostrado no Capítulo 3, Figura 5. O método adaptado desenvolvido é visível no Algoritmo 5:

Algoritmo 5 Método adaptado das secantes com seleção de intervalo desenvolvido

Input: TOL, Q_d, Q_m, I_{Stable}
Output: $I_{REF} \mid \varepsilon \leq \text{TOL}$

Converged $\leftarrow$ **False**
$Q_{max} \leftarrow Q_m(I_{Stable})$
if $Q_d > \gamma \cdot Q_{max}$ **then** ▷ Seleção na região linear
    $a \leftarrow \alpha \cdot I_{Stable}$
    $b \leftarrow I_{Stable}$
else ▷ Seleção na região não-linear
    $a \leftarrow 0$
    $b \leftarrow \beta \cdot I_{Stable}$
end if
while not Converged **do**

$$\text{slope} \leftarrow \frac{Q_m(b) - Q_m(a)}{b - a}$$ ▷ $Q_m(a), Q_m(b)$ são novas medições

$$c \leftarrow b - \frac{Q_m(b) - Q_d}{\text{slope}}$$

$I_{REF} \leftarrow c$
$Q_m \leftarrow Q_m\left(I_{REF}\right)$ ▷ Medição do Q com o novo valor de I_{REF}
$\varepsilon \leftarrow |Q_m - Q_d|$
if $\varepsilon \leq$ TOL **then** Converged $\leftarrow$ **True**
else
    $a \leftarrow b$
    $b \leftarrow c$
end if
end while

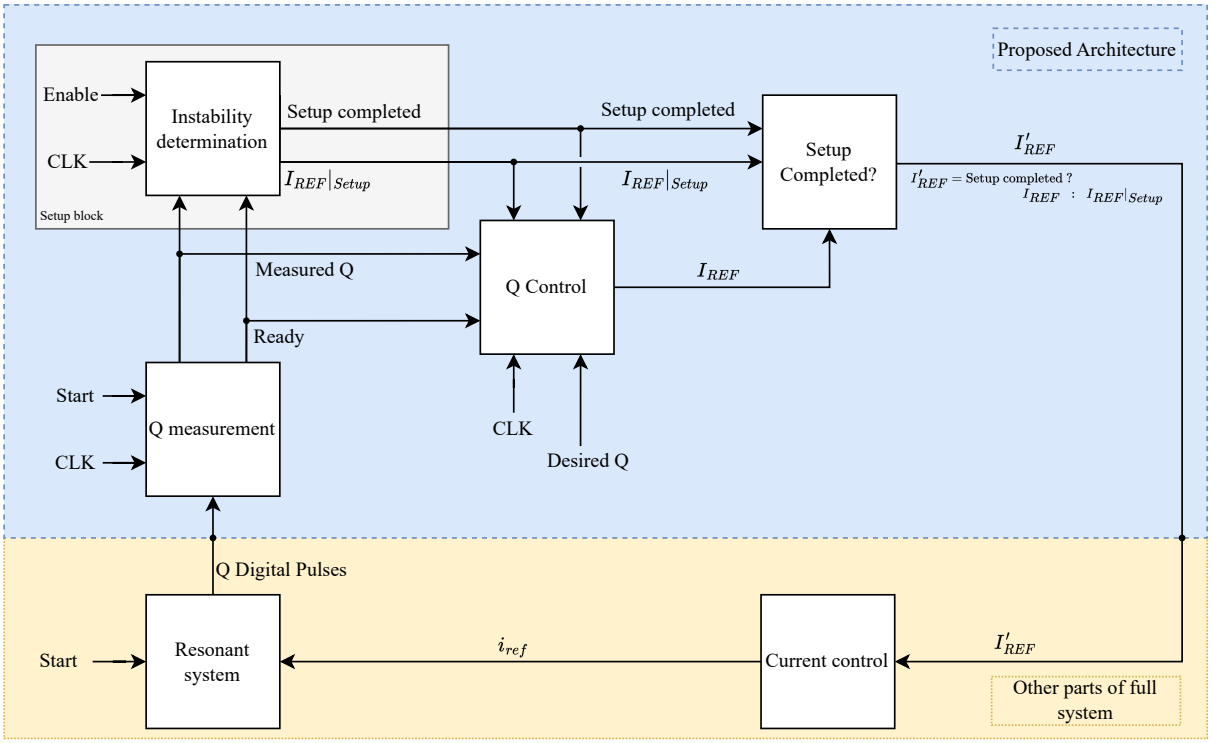
Além das considerações sobre as chamadas feitas ao bloco de medição e as mesmas considerações relativas ao método original, o método modificado é baseado em experimentação. Dessa forma, sempre haverá margem para otimização na escolha dos coeficientes. Um problema de otimização à parte.

4.2 Arquitetura do sistema

Em linhas gerais, a arquitetura deve responder à um único requisito funcional: O usuário entra com um valor decimal para o Q desejado e o sistema deve retornar o valor de

corrente referente à este valor de Q desejado. Figura 6 ilustra a arquitetura desenvolvida com todas[2] as entradas, saídas e blocos correspondentes ao circuito proposto:

Figura 6 – Arquitetura desenvolvida para o sistema



O cenário base de funcionamento em um ciclo completo desde a inicialização até ter atingido o valor de Q desejado é elencado da seguinte forma:

1. O sistema inicializa variáveis internas e prossegue para a determinação do valor máximo de corrente de referência de configuração ($I_{REF|Setup}$) conforme justificado posteriormente na subseção 4.2.2 – Determinação de limite de instabilidade: *Instability determination*.

2. O bloco de determinação de instabilidade manipula a corrente de configuração do sistema até determinar onde é o ponto máximo de estabilidade. O valor da corrente é atualizado sempre que recebe uma nova medição do bloco de medição do Q à cada ciclo de *clock* até atingir o ponto de estabilidade.

3. Com o limite superior de corrente determinado, este valor é passado ao bloco de controle como valor máximo de corrente a ser injetada no sistema ressonante. Neste momento o bloco *Q control* está livre para manipular a corrente I_{REF} para alcançar o fator-Q desejado, logo, habilita-se o bloco de controle do Q. O valor de corrente

[2] Todos os blocos têm um *reset* assíncrono, não representado para simplificação

injetada é atualizado sempre que recebe uma nova medição do bloco de medição do Q.

4. Após o alcançar o Q desejado, o sistema mantém o valor de corrente para o bloco que a controla, mantendo este estado até receber uma nova solicitação de um valor de Q desejado.

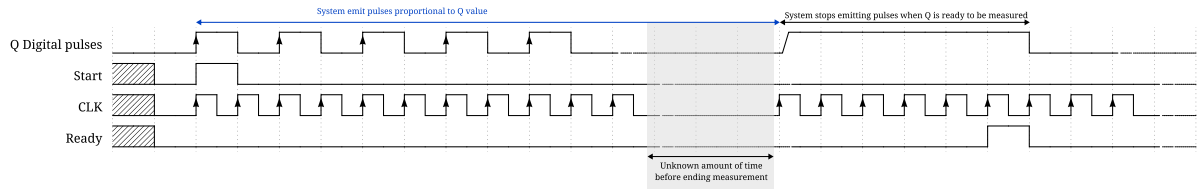
As partes subsequentes (subseção 4.2.1 – subseção 4.2.4) descrevem o funcionamento bloco a bloco individualmente e no que diz respeito à comunicação entre blocos.

Vale ressaltar que, apesar de serem desenvolvidos três métodos de controle do fator de qualidade, apenas um é testado por vez (e posteriormente o projeto avançará apenas com o método de melhor desempenho com relação à todo o sistema).

4.2.1 Medição do Q: Q *measurement*

A medição do Q é um passo fundamental que acaba por coordenar os demais blocos. Essa responsabilidade de coordenação dá-se pela característica de que os pulsos emitidos pelo sistema ressonante (Q *digital pulses*) não "avisam" que a medição do Q terminou, como ilustra a Figura 7. Sendo assim, este bloco é responsável por temporizar a operação dos blocos subsequentes pois faz interface direta com o sistema ressonante. O diagrama de temporização da Figura 7 ilustra o comportamento do bloco que mede o valor de Q:

Figura 7 – Diagrama de tempo da operação do bloco de medição.



Fonte: o autor

O sistema ressonante emite n pulsos continuamente por um período de tempo indeterminado[3] proporcional ao Q até detectar que a medição está completa. Neste momento, para-se de emitir pulsos e congela-se o valor lógico alto indicando que a medição terminou. Como não há nenhuma *flag* emitida pelo sistema ressonante para indicar que a medição está pronta, implementa-se uma estratégia de contador que aciona a *flag ready* gerada pelo bloco de medição.

Para detectar que a medição está completa e levar à nível alto a *flag* "*ready*" incrementa-se um contador interno à cada Ciclo de *clock*. Caso no ciclo seguinte o valor do

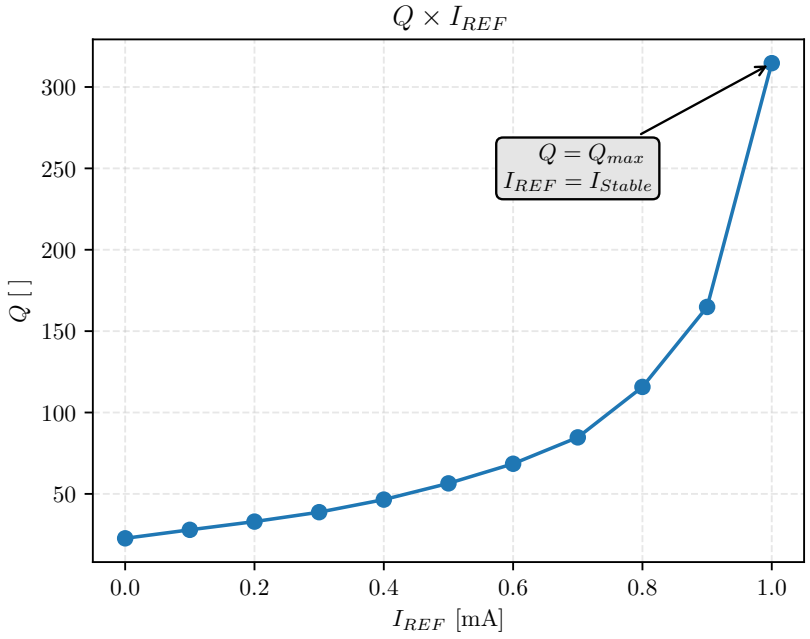
[3] O período mínimo e máximo são conhecidos uma vez que os valores mínimos e máximos de Q são medidos

pulso esteja alto, incrementa-se esse contador. Caso seja baixo, reseta-se. A *flag ready* é setada como "1" quando o valor do contador atinge o valor decimal 5. Assim, este contador age como um temporizador que "espera" os pulsos acabarem por 5-CC para indicar que a medição terminou e setar a *flag ready*. Este valor foi obtido empiricamente, abrindo margem para futura otimização.

4.2.2 Determinação de limite de instabilidade: *Instability determination*

O comportamento da curva $Q \times I_{REF}$, conta com uma característica na qual a partir de certo ponto de corrente o Q tende ao infinito e cai abruptamente a partir dese ponto com o aumento da corrente. Desta forma é necessário determinar o ponto máximo de estabilidade do sistema à fim de não ultrapassar a região de estabilidade no momento de fazer a busca do fator de qualidade desejado. Este ponto está destacado na Figura 8:

Figura 8 – Curva $Q \times I_{REF}$ com destaque para o ponto máximo de estabilidade



Fonte: o autor

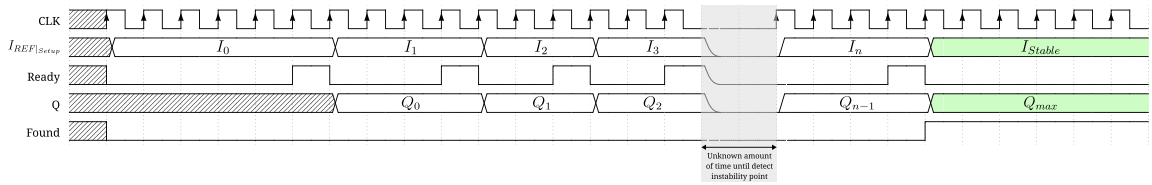
Para encontrar limite de instabilidade, sendo o ponto $p = (I_{Stable}, Q_{Máx})$ o bloco em questão opera conforme o algoritmo: Algoritmo 6:

Algoritmo 6 Algoritmo de determinação de instabilidade

Input: $\Delta Q, \Delta I_{REF}, Q_m$
Output: I_{Stable}
1: $I_{Stable} \leftarrow I_{MAX}$ ▷ $I_{MAX} = 1\text{mA}$
2: Found $\leftarrow$ **False**
3: **while** not Found **do**
4: $\quad Q_i \leftarrow Q_m(I_{Stable})$
5: $\quad I_{Stable} \leftarrow I_{Stable} - \Delta I_{REF}$
6: $\quad Q_{i+1} \leftarrow Q_m(I_{Stable})$
7: $\quad$ **if** $(Q_{i+1} - Q_i \geq \Delta Q)$ **then**
8: $\quad\quad$ Found $\leftarrow$ **True**
9: $\quad$ **end if**
10: **end while**
11: **return** I_{Stable}
Fonte: o autor

No domínio do tempo, vale lembrar que a medição do fator de qualidade não é instantânea e depende da *flag ready* do bloco de medição. O diagrama de temporização da Figura 9 ilustra o comportamento:

Figura 9 – Diagrama de tempo da operação do bloco de determinação de instabilidade



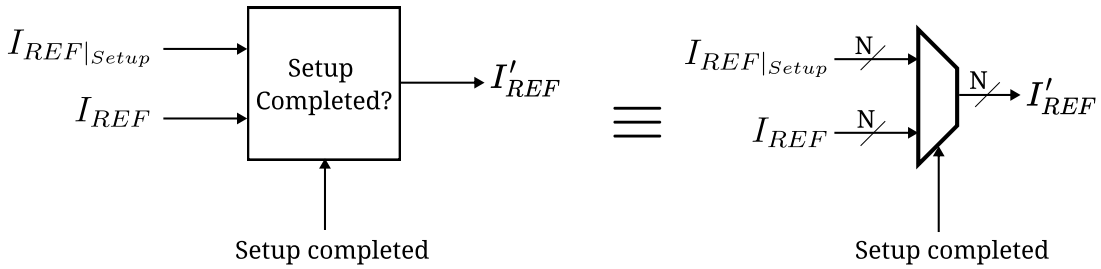
Fonte: o autor

Alinhando Algoritmo 6 com a Figura 9, tem-se que na corrente inicial I_0 (corrente máxima do sistema) é decrementado o valor ΔI_{REF} passado como parâmetro do algoritmo até que o decremento de corrente produza um ΔQ maior ou igual ao parâmetro especificado, atingindo assim o ponto $p = (I_{Stable}, Q_{Máx})$. Quando este ponto é alcançado um registrador interno `Found` é setado para 1, indicando que o ponto está determinado e a operação de configuração finalizada. Neste momento o bloco de controle do Q está livre para atuar.

4.2.3 Seleção de correntes: *Setup completed*

Este bloco é essencialmente um multiplexador 2×1 com $N = 10$ bits em cada canal de entrada (e saída) na via principal de dados usados no sistema. A Figura 10 ilustra a simplificação deste bloco:

Figura 10 – Representação do bloco intertravamento ou seleção de correntes

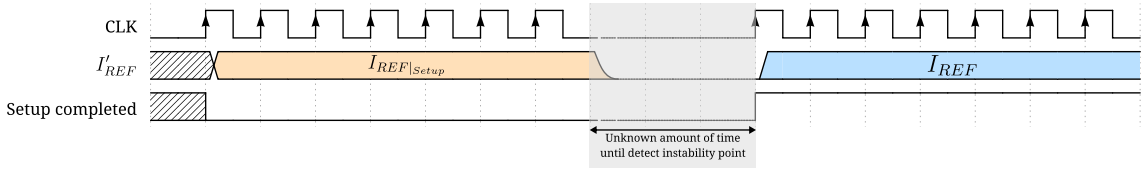


Esse bloco garante que apenas uma das correntes seja injetada ao mesmo tempo no sistema ressonante. Esta seleção entre $I_{REF}|_{setup}$ e I_{REF} se faz necessária devido a necessidade de determinar primeiramente a corrente máxima do circuito ressonante ($I_{REF}|_{setup}$). Após esta determinação, o circuito digital poderá efetuar a busca pelo Q desejado sendo $I_{REF}|_{setup}$ o limite superior de corrente, pois após este ponto, o sistema se encontrará em instabilidade.

O bloco *Setup Completed?* possui uma variável seleção de 1 bit chamada *Setup completed* e faz a multiplexação dos sinais $I_{REF}|_{setup}$ e I_{REF} para a saída I'_{REF}. Enquanto esta variável de seleção estiver em nível lógico baixo, o bloco *Instability Determination* buscará o valor máximo de Q e, portanto, faz com que a saída I'_{REF} receba a informação de $I_{REF}|_{setup}$. A variável de seleção somente ficará em nível lógico alto quando o valor máximo de Q for determinado, assim, permitindo que a saída $I_{REF}|_{setup}$ receba a informação de I_{REF} para a convergência do Q desejado através do bloco *Q Control*. A sua saída (I'_{REF}) É resultado da variável de seleção (*Setup completed*).

As formas de onda ilustradas na Figura 11 exemplificam um ciclo de funcionamento da fase de configuração para a fase de controle e manipulação da corrente:

Figura 11 – Diagrama de tempo da operação do bloco de seleção de corrente



Fonte: o autor

Da Figura 11 infere-se que há um primeiro momento onde a corrente de configuração é injetada até que seja determinado um valor máximo de corrente para manter a estabilidade. Quando isto ocorre a *flag Setup completed* recebe o valor "1" e então o sistema está apto a controlar a corrente para receber o Q desejado.

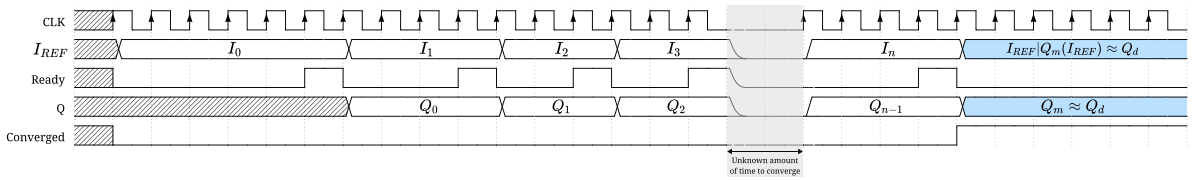
4.2.4 Controle do Q: *Q Control*

Quando o sistema já tem determinado qual o limite superior de corrente a fim de não atingir a instabilidade (subseção 4.2.2), o bloco *Q control* é ativado para controlar a corrente Iref e buscar o Q desejado inserido pelo usuário.

Retomando o que foi mencionado nos Objetivos, neste projeto foram implementados 3 métodos principais mencionados na seção 4.1. O bloco de controle em si contém e executa apenas um método por vez, não sendo possível alternar entre métodos uma vez compilado o código e executadas as simulações. O sistema é projetado desta forma pois o objetivo é selecionar o método com melhor desempenho acerca de todo o sistema e prosseguir para o fluxo de fabricação.

O diagrama de temporização deste bloco (Figura 12) é bastante similar ao do bloco de determinação de instabilidade (Figura 9).

Figura 12 – Diagrama de tempo da operação do bloco de controle do Q.



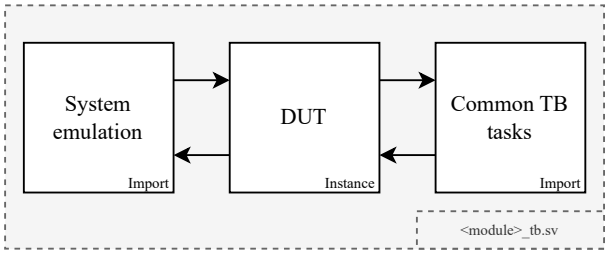
Fonte: o autor

Para cada método de controle o diagrama de temporização pode variar brevemente no que diz respeito ao número de ciclos de *clock* para terminar uma operação e no *overhead* inicial. Por exemplo, o método das secantes necessita de 3 medições de Q para a primeira comparação enquanto o da bisseção necessita somente de duas.

4.3 Arquitetura de testes do sistema

A arquitetura de testes foi desenvolvida para validar individualmente cada módulo do sistema. Como há comunicação entre os módulos desenvolvidos e o sistema ressonante, o *testbench* foi abstraído em 3 elementos, representados na Figura 13:

Figura 13 – Arquitetura dos *testbenches*



Da Figura 13 há 3 elementos presentes em cada arquivo de *testbenching* que diz respeito à cada módulo `<module>_tb.sv`, sendo eles:

1. DUT – Módulo: O módulo a ser testado com respectivas conexões
2. *System emulation* – Package: É um pacote a ser importado no `<module>_tb.sv` com as funcionalidades referentes à emulação do comportamento do sistema ressonante. Em sua maioria é apenas a geração dos *Q digital pulses* à partir de um start, como ilustrado na Figura 3 – Arquitetura simplificada do sistema completo
3. *Common TB tasks* – Package: É outro com funcionalidades comuns à tesbenches: geração de *clock*, monitoramento de variáveis, entrada e saída de dados para arquivos.

A arquitetura de testes foi projetada desta forma para testar os módulos desenvolvidos na Arquitetura desenvolvida para o sistema de forma independente e promover o reuso de funções em comum. Com essa estrutura é possível testar todos os módulos desenvolvidos, adequando apenas os cenários de teste, adequando apenas o arquivo principal `<module>_tb.sv`.

Nesta fase do projeto todos os módulos foram testados seguindo a estrutura mencionada ao longo desta seção. Em um segundo momento, como mencionado no Capítulo 2 – Objetivos os blocos serão testados de forma conjunta, verificando a sincronia e os e a coerência dos resultados obtidos do sistema acoplado *versus* blocos independentes.

5 Resultados e análise

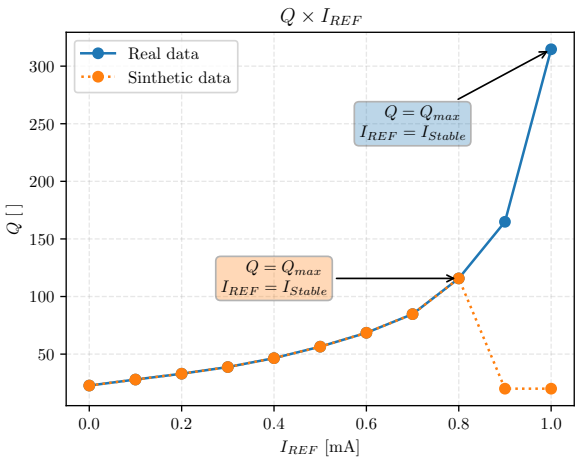
Nesta fase do projeto, foram implementados e validados individualmente dois dos blocos principais: bloco de controle do Q e bloco de determinação de instabilidade.

No bloco de determinação de instabilidade a implementação foi feita diretamente em HDL. Para os métodos de aproximação foram primeiro prototipados em alto nível e na sequência transcritos para HDL, comparando-se o resultado entre métodos e entre implementações

5.1 Resultados do bloco de determinação do limite de estabilidade

Para o teste de determinação do ponto máximo de estabilidade foi modificada a curva padrão de $Q \times I_{REF}$ utilizada em todo trabalho de forma sintética, isto é, foram alterados os dados para que o ponto de instabilidade estivesse localizado no ponto (índice) 850 de $I_{REF} = 0.83\text{mA}$. A curva modificada consta na Figura 14:

Figura 14 – Dados teóricos e sintéticos utilizados para o teste de determinação de instabilidade



Fonte: o autor

Os dados sintéticos foram usados em virtude de que, se fossem usados os dados reais, o algoritmo encontraria o ponto logo na primeira operação, uma vez que o ponto onde $I_{REF} = 1\text{mA}$ é o ponto máximo de estabilidade.

Testa-se o bloco com os parâmetros da Tabela 2 e os resultados são visíveis na Tabela 3:

Tabela 2 – Parâmetros passados ao algoritmo de determinação de instabilidade

Parâmetro	**Valor**	**Descrição**
ΔQ	50 [ ]	Variação de Q considerada suficiente para sair da região instável
ΔI_{REF}	10 [bits]	Valor de decremento da corrente de configuração
$p = (I_{Stable}, Q_{max})$	(850 [bits], 113 [ ])	Ponto (definido) de estabilidade
MAX_ITER	50	Numero máximo de iterações. (neste caso são ciclos de clock)

Fonte: o autor

Tabela 3 – Resultados da busca por ponto máximo de estabilidade.

Time	I_{REF} [bits]	Q_m	I_{REF} [mA]
2	1013	23	0.989258
7	993	23	0.969727
12	963	23	0.940430
17	943	23	0.920898
22	913	23	0.891602
27	893	23	0.872070
32	863	23	0.842773
36	843	113	0.823242

Fonte: o autor

Da Tabela 3 verifica-se que o ponto de estabilidade é determinado como 843 enquanto o esperado é 850 após 36 ciclos de *clock*. Como os parâmetros de $\Delta I_{REF}, \Delta Q$ foram ajustados priorizando a proximidade com o ponto definido (850). Pode-se priorizar a rapidez da busca em troca de obter um Q_{max} menor ajustando os parâmetros.

5.2 Resultados do bloco de controle do Q

O bloco de controle do Q foi testado em dois cenários: o padrão, onde a tolerância é coerente com os valores de Q produzidos; o sintético, onde a tolerância é baixa. Foram testados os 3 métodos em alto nível e o método da bisseção em HDL.

5.2.1 Algoritmos em alto nível

Primeiro, realiza-se um teste busca de um valor único seguindo as especificações da Tabela 4. Os valores elencados na Tabela 4 foram selecionados de forma que o teste representasse um cenário real de uma busca única. Os valores mínimos e máximos de corrente foram definidos de acordo com as capacidades do sistema ressonante, o valor de tolerância de acordo com a resolução do sistema de medição e o número máximo de iterações fixado em 32 para o caso de algum método não alcançar convergência.

Tabela 4 – Especificações do teste de busca por valor único

Parâmetro	**Valor**	**Descrição**
a	0.0	Limite inferior inicial de I_{REF}
b	1.0	Limite superior inicial de I_{REF}
TOL	± 30	Tolerância ou erro máximo admissível
Q_d	110	Valor de Q desejado
MAX_ITER	32	Numero máximo de iterações.

Fonte: o autor

Executando o teste com os parâmetros da Tabela 4 para cada método desenvolvido em alto nível, têm-se os resultados para cada método na Tabela 5 em suas respectivas sub-tabelas.

Tabela 5 – Resultados de execução do teste de busca única para cada método, $Q_d = 110$.

(a) Bisseção

Iteração	a	b	c	$f(c)$	ε
1	0.0	1.0	0.50	56.743276	-53.256724
2	0.5	1.0	0.75	98.171202	-11.828798

(b) Secante

Iteração	a	b	c	$f(c)$	ε
1	0.000000	1.000000	0.299006	38.839856	-71.160144
2	1.000000	0.299006	0.479866	54.497986	-55.502014
3	0.299006	0.479866	1.120945	1623.460030	1513.460030
4	0.479866	1.120945	0.502544	57.028494	-52.971506
5	1.120945	0.502544	0.523456	59.379481	-50.620519
6	0.502544	0.523456	0.973731	248.961601	138.961601
7	0.523456	0.973731	0.643685	74.313694	-35.686306
8	0.973731	0.643685	0.711124	87.462432	-22.537568

(c) Secante modificada

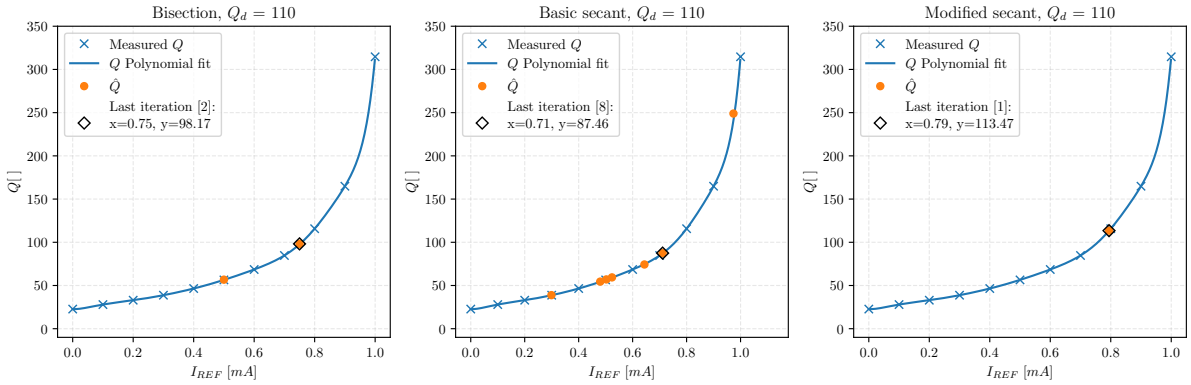
Iteração	a	b	c	$f(c)$	ε
1	0.8	1.0	0.7943	113.472012	3.472012

Fonte: o autor

Dos dados obtidos com uma busca de valor percebe-se que a secante modificada e a secante clássica comportam-se de forma diferente apesar de serem baseados no mesmo algoritmo, simplesmente por escolher o intervalo de busca de maneira diferente. Assim, o algoritmo original apresentou o pior desempenho dos três, sendo o da bisseção o com

desempenho intermediário. Sintetizando essa busca em um gráfico , na Figura 15 vê-se a aproximação do Q para o valor desejado:

Figura 15 – Pontos obtidos por iteração e curva $Q \times I_{REF}$ dos métodos de busca em alto nível



Fonte: o autor

Para um teste mais abrangente foi feito um *sweeep* de valores de Q desejado (Q_d) e medido quantas iterações foram necessárias para que o algoritmo viesse a convergir. As especificações do teste são visíveis na Tabela 6:

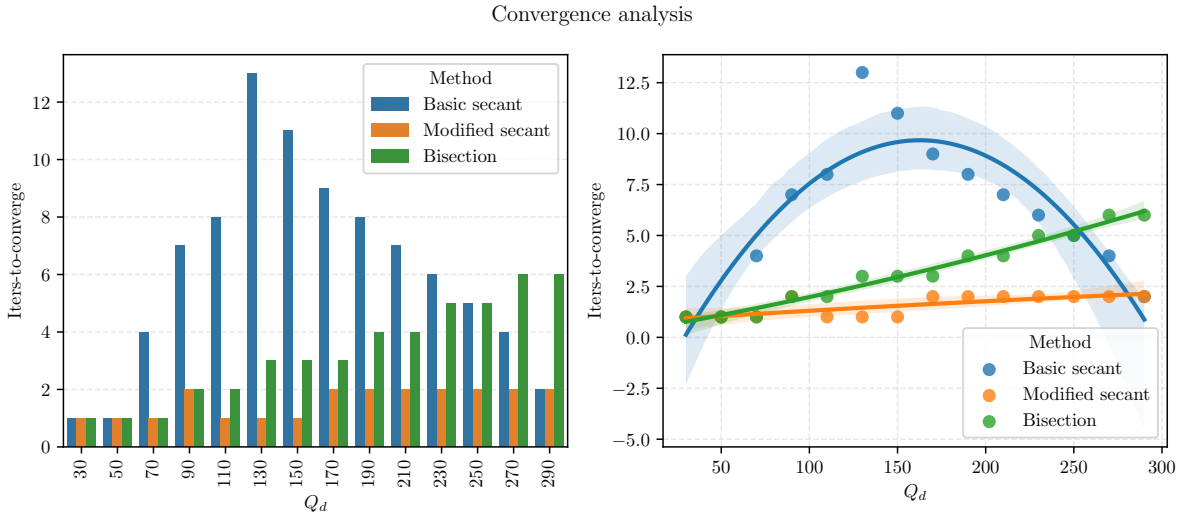
Tabela 6 – Especificações do teste de busca em *sweep*

Parâmetro	**Valor**	**Descrição**
a	0.0 [mA]	Limite inferior inicial de I_{REF}
b	1.0 [mA]	Limite superior inicial de I_{REF}
TOL	± 30	Tolerância ou erro máximo admissível
Q_d	30 até 300 com passo de 20	Valor de Q desejado
MAX_ITER	32	Numero máximo de iterações.

Fonte: o autor

Os resultados do teste com as especificações da Tabela 6 são visíveis na Figura 16 em conjunto com um ajuste polinomial de $Q_d \times$ ITC:

Figura 16 – Número de iterações para convergência e valor desejado de Q com alta tolerância



Fonte: o autor

Na Figura 16 verifica-se um desempenho razoável para todos os métodos, porém, o método da secante com seleção de intervalo apresentou melhor desempenho, enquanto o da secante básica, o pior.

Um comportamento que se destaca pelas linhas de tendência do gráfico é que tanto a secante modificada quanto o método da bisseção têm sua ordem de convergência aproximadamente linear enquanto a secante básica apresenta uma característica parabólica negativa. Em outras palavras, a secante básica demora mais para convergir quando o Q_d está localizado no ponto central da curva de $Q \times I_{REF}$. A bisseção e a secante modificada demoram mais enquanto o ponto está mais próximo do valor máximo atingível. A Tabela 7 apresenta algumas estatísticas sobre cada método.

Tabela 7 – Tabela com mínimo, máximo, média, desvio e variância de ITC por método

	Min ITC	Max ITC	Mean ITC	STD ITC	VAR ITC
Basic secant	1	13	6.142857	3.591810	12.901099
Bisection	1	6	3.285714	1.772811	3.142857
Modified secant	1	2	1.571429	0.513553	0.263736

Fonte: o autor

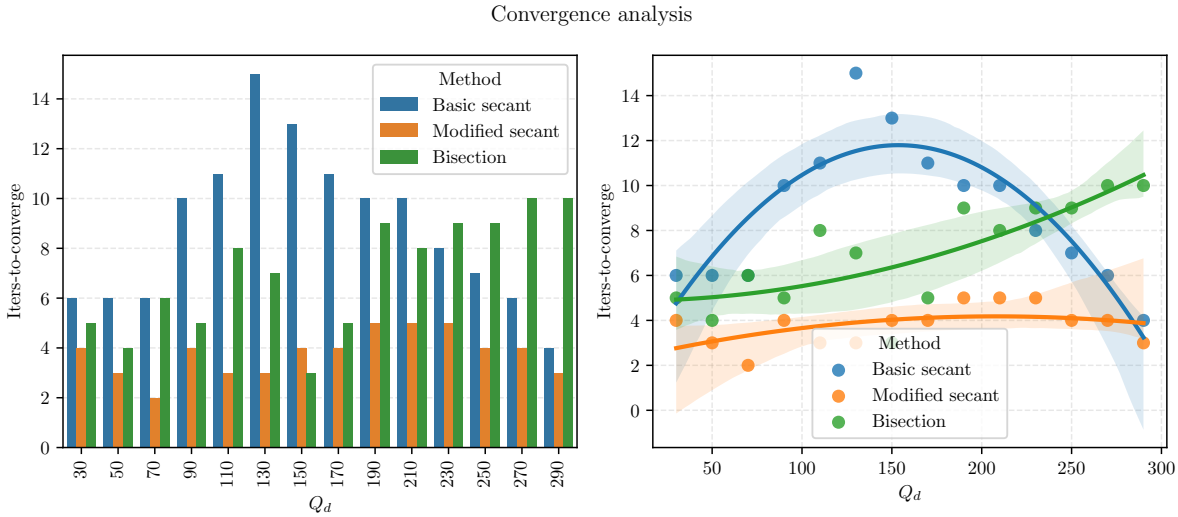
Na média, a secante básica requer o dobro de iterações para convergência se comparado com o da bisseção, que por sua vez também requer aproximadamente o dobro da secante modificada. Os valores máximos seguem o mesmo padrão. Os valores mínimos

acabam por ser iguais devido a alta tolerância. O desvio segue a tendência da média indicando baixa dispersão.

5.2.1.1 Cenário sintético com baixa tolerância

O teste em *sweep* foi repetido com menor tolerância, para investigar se os métodos são capazes de alcançar a convergência em um cenário sintético onde o Q_m pudesse ser medido com maior resolução. As especificações são essencialmente as mesmas da Tabela 6 com exceção da tolerância que agora é ± 1 Os resultados do teste sintético são visíveis na Figura 17 em conjunto com um ajuste polinomial de $Q_d \times$ ITC:

Figura 17 – Número de iterações para convergência e valor desejado de Q com baixa tolerância



Fonte: o autor

Na Figura 17 verifica-se o mesmo padrão encontrado no cenário realista de testes (Resultados da Figura 16), onde a tolerância é mais alta. O método da secante com seleção de intervalo manteve-se com o melhor desempenho, entretanto, com uma característica que agora mais se assemelha à uma parábola com concavidade negativa. O método da bisseção continua a sendo o método intermediário mas agora com um formato parabólico positivo. O método da secante básica manteve exatamente as mesmas características. Na Tabela 8 estão resumidas algumas estatísticas dos métodos:

Tabela 8 – Tabela com mínimo, máximo, média, desvio e variância de ITC por método

	Min ITC	Max ITC	Mean ITC	std ITC	var ITC
Basic secant	4	15	8.785714	3.142233	9.873626
Bisection	3	10	7.000000	2.320477	5.384615
Modified secant	2	5	3.785714	0.892582	0.796703

Fonte: o autor

Agora, na média, a secante básica não requer mais o dobro de iterações para convergência se comparado com o da bisseção: a secante básica é 1.25× mais lenta. No caso da bisseção com a secante modificada, a diferença ainda é alta, sendo aproximadamente 1.85× mais lenta. Os valores mínimos são relativamente próximos enquanto os máximos apresentam diferença de 1.5× e de 2× comparando o melhor método com o intermediário, e o intermediário com o pior, respectivamente. O desvio padrão e a variância são mais altos nos casos da secante básica e da bisseção se comparados com a secante modificada, indicando alta dispersão.

5.2.2 Algoritmos em HDL

Para os algoritmos em HDL testa-se somente o cenário sintético, uma vez que sendo capazes de alcançar convergência com baixa tolerância, também serão capazes de alcançar com menor exigência de precisão.

Os dois métodos das secantes não foram testados num cenário abrangente, pois, apesar de terem sido desenvolvidos, apresentaram problemas de convergência quando implementados em HDL. Estes problemas ocorreram no cálculo da inclinação da reta secante e posterior atribuição do valor de c.

Se retomarmos o cálculo da inclinação da reta dos algoritmos 4 e 5, percebe-se que este pode envolver valores próximos de 0. No momento que $0 \leq slope < 1$ essa variável recebe o valor de 0 pois a resolução usada é de 1 bit. Esse arredondamento é um problema neste contexto pois gera divisão por zero, então o método atribui um valor indefinido ('x' em hardware) para c, que faz com que o método fique estagnado em um valor de corrente.

Para tentar contornar o problema, no algoritmo desenvolvido foi usada uma atribuição condicional para caso a inclinação recebesse o valor de 0. Apesar de convergir para alguns valores, o método torna-se instável e não converge para outros valores. Desta forma, mostra-se nos resultados um exemplo onde foi possível atingir a convergência e outro onde não foi.

5.2.3 Resultados do método da bisseção

Repete-se o teste sintético com os parâmetros da Tabela 4 para a busca de valor único exceto pela tolerância de ± 1 para um $Q_d = 110$. O resultado dessa é visto na Tabela 9.

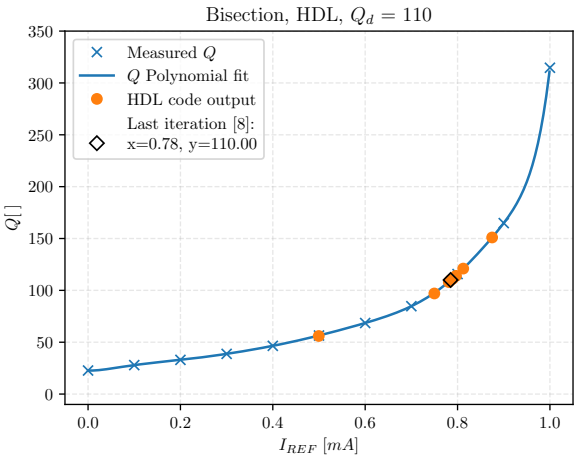
Tabela 9 – Pontos obtidos por iteração no método da bisseção

I_{REF} [bits]	I_{REF} $[mA]$	Q_m
511	0.499511	56
767	0.749756	97
895	0.874878	151
831	0.812317	121
799	0.781036	108
815	0.796676	114
807	0.788856	111
803	0.784946	110

Fonte: o autor

Construindo um gráfico com os pontos da Tabela 9 verifica-se na Figura 18 que o método converge para mais perto do valor desejado à cada iteração até atingir o ponto onde $TOL \leq Q_m - Q_d$.

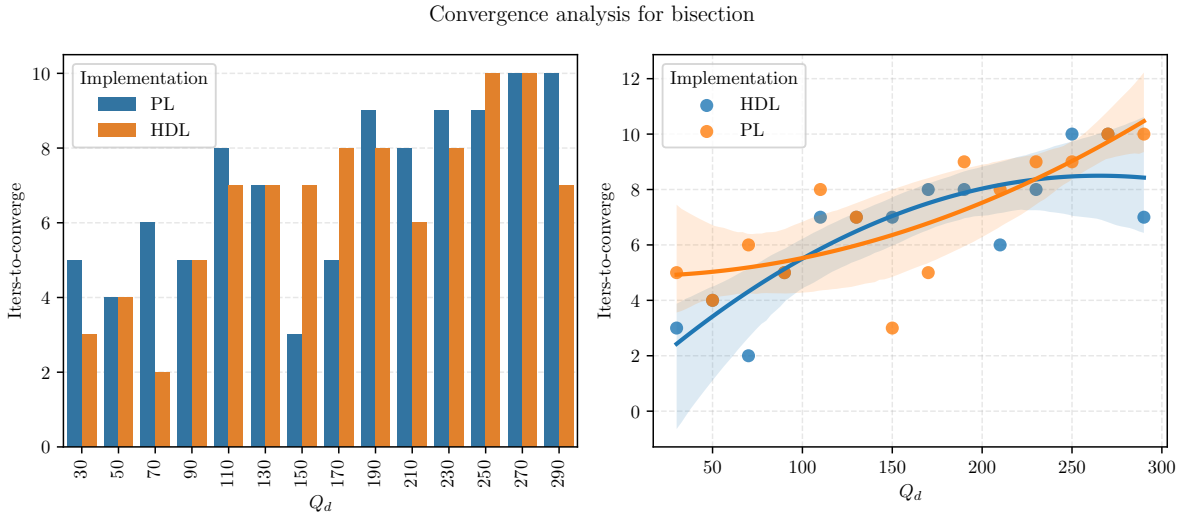
Figura 18 – Gráfico dos pontos obtidos por iteração no método da bisseção



Fonte: o autor

Realiza-se também um *sweep* de valores assim como feito para os algoritmos em alto nível. Neste teste compara-se o desempenho do método da bisseção implementado em linguagem de programação (PL) *versus* sua versão em linguagem de descrição de hardware (HDL). Esses resultados são visíveis na Figura 19.

Figura 19 – Número de iterações para convergência *versus* Q_d do método da bisseção em de HDL *versus* PL



Fonte: o autor

Verifica-se na Figura 19, que os algoritmos implementados em hardware e software têm seu desempenho similar. As diferenças surgem pelos arredondamentos e aritmética de ponto flutuante: em software são usados 6 dígitos após a virgula para valores de $c(I_{REF})$ resultando em $\pm 1nA$ de precisão – de fato exagerado. Enquanto em hardware, em conversão direta são usados valores inteiros num range de 0 a 1023, resultando em $\approx 0.97\mu$A de precisão. Essa variação de precisão por vezes fez com que o método implementado em PL fosse mais lento com que sua versão em HDL. Em linhas gerais, o desempenho é o mesmo. Resumindo as estatísticas da busca em *sweep* na Tabela 10.

Tabela 10 – Tabela com mínimo, máximo, média, desvio e variância de ITC por implementação

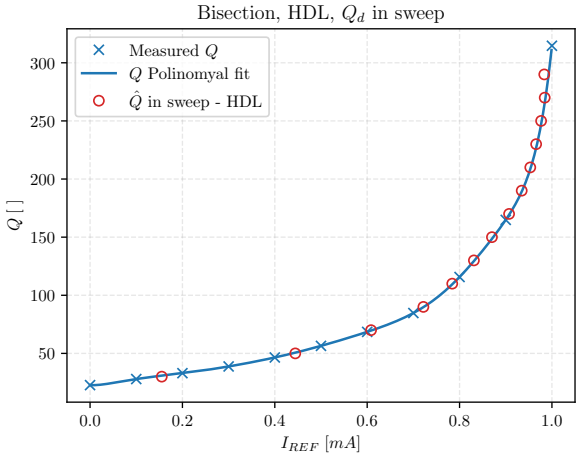
Implementation	Min ITC	Max ITC	Mean ITC	STD ITC	VAR ITC
HDL	2	10	6.571429	2.376626	5.648352
PL	3	10	7.000000	2.320477	5.384615

Fonte: o autor

Na Tabela 10 percebe-se estatísticas similares por implementação. Enquanto os valores máximos são iguais, os valores mínimos e médios são menores para a versão em HDL dados pela menor precisão. Por sua vez, desvio e variância são ligeiramente menores em PL.

Por último, representa-se a busca em sweep feita para o método da bisseção em HDL na Figura 20. Neste gráfico, espera-se que os valores produzidos pelo algoritmo (Circulos vermelhos – $\hat{Q}$) em sweeep reconstruam a forma da curva.

Figura 20 – Busca em sweep dos valores de Q no método da bisseção



Fonte: o autor

Percebe-se que os pontos aproximados estão suficientemente próximos do Q_d com a alta precisão requerida ao algoritmo.

5.2.4 Resultados do método das secantes

Como comentado no início deste capítulo, o método das secantes não apresentou bons resultados. Na Tabela 11 foi realizado um teste para a busca de $Q_d = 250$ com tolerância $TOL = \pm 20$.

Tabela 11 – Busca de $Q_d = 250$ pela secante com alta tolerância

I_{REF} [bits]	I_{REF} [mA]	Q_m
1022	0.998047	314
826	0.806641	119
1022	0.998047	314
794	0.775391	106
794	0.775391	106
694	0.677734	80
694	0.677734	80
694	0.677734	80
1002	0.978516	255

Para a baixa tolerância, o método consegue convergir. Porém, é visível que há algum problema na implementação uma vez que os valores de corrente se repetem ao longo

de algumas iterações. Ressalta-se que para outros valores com alta tolerância a erro o método também não conseguiu convergir.

Na Tabela 12 refaz-se o mesmo teste com o valor de tolerância $TOL = \pm 1$.

Tabela 12 – Busca de $Q_d = 250$ pela secante com baixa tolerância

I_{REF} [bits]	I_{REF} [mA]	Q_m
1022	0.998047	314
826	0.806641	119
1022	0.998047	314
794	0.775391	106
794	0.775391	106
694	0.677734	80
694	0.677734	80
694	0.677734	80
1002	0.978516	255
1002	0.978516	255
994	0.970703	238
994	0.970703	238
994	0.970703	238
536	0.523438	59
536	0.523438	59
764	0.746094	96
764	0.746094	96
764	0.746094	96
703	0.686523	82
703	0.686523	82
711	0.694336	83
711	0.694336	83
711	0.694336	83
878	0.857422	142

O mesmo padrão de repetição de valores de correntes visualizado na Tabela 11 ocorre na busca com menor tolerância. Verifica-se na Tabela 12 que mesmo após 24 iterações o método não converge para o valor de $Q_d = 250$. Cenário esse que não aconteceu na implementação em alto nível, onde o valor máximo de iterações necessárias foi 14, e para o valor em questão foram feitas 7 iterações.

Desta forma, é necessário estudar uma nova forma de implementar este método (e por consequência a secante com seleção de intervalos) a fim de analisar comparativamente o desempenho dos métodos em HDL.

Porém, como visto no funcionamento dos métodos em alto nível, possivelmente o método da bisseção é o melhor método com relação ao tempo total de funcionamento do sistema pois realiza menos medições por iteração. Proposição este que deve ser confirmada

com a readequação dos métodos baseados em secantes.

Parte III

Conclusões

6 Problemas encontrados e melhorias futuras

O principal problema encontrado foi descrito no Capítulo 5 – Resultados e análise, no que diz respeito ao método das secantes. Uma vez que a variável que armazena o valor da inclinação da reta secante pode ser um valor decimal entre 0 e 1, requer atenção para que o método possa funcionar da maneira adequada. Uma alternativa que pode ser razoável para resolver este problema é empregar uma estrutura de dados de ponto flutuante no hardware.

Outra parte em que não se considera um problema, mas sim uma melhoria é a otimização dos parâmetros ΔQ e ΔI_{REF} do algoritmo de determinação de instabilidade. Apesar de neste método haver um compromisso entre desempenho e precisão, não foi feito um ajuste fino para alcançar um bom balanço entre os dois. Sabe-se que neste trabalho a prioridade é que o ajuste do Q seja feito de forma rápida, mas abrir mão da precisão (especialmente na corrente) pode limitar o Q_{max} do circuito em um fator considerável, já que pequenas variações de corrente produzem alta variação do Q na região não linear da curva $Q \times I_{REF}$.

No bloco de determinação de instabilidade também pode ser inclusa uma condição na qual, caso o Q_{max} obtido seja próximo o suficiente do Q_d na tolerância especificada, o sistema mantém-se nesse estado e não permite a operação do bloco do controle do Q. Porém, esta modificação requer uma nova análise se a arquitetura projetada consegue promover esta nova funcionalidade.

Ademais, vistas as considerações de melhorias e soluções de problemas feitas ao longo deste capítulo, parte-se para a implementação do fluxo de back-end mencionadas nos objetivos.

Referências

1 OHIRA, T. What in the world is Q? – distinguished microwave lecture. *IEEE Microwave Magazine*, v. 17, n. 6, p. 42–49, 2016. Citado 2 vezes nas páginas 25 e 26.

2 LIAO, R.; TAN, J.; WANG, H. Q-based design method for impedance matching network considering load variation and frequency drift. *Microelectronics Journal*, v. 42, n. 2, p. 403–408, 2011. ISSN 0026-2692. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0026269210002284>. Citado na página 26.

3 CHUNG, B. Q-based design method for t network impedance matching. *Microelectronics Journal*, v. 37, n. 9, p. 1007–1011, 2006. ISSN 0026-2692. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0026269206000310>. Citado na página 26.

4 SILVA, P. M. M. e. *Contributions on the automatic tuning of LC networks using on- chip circuits*. Tese (Doutorado) — UFSC, 2017. Disponível em: <https://repositorio.ufsc.br/handle/123456789/185002>. Citado na página 33.

5 BURDEN, R.; FAIRES, J. *Numerical Analysis*. 10^{th}. Cengage Learning, 2010. (The Prindle, Weber and Schmidt Series in Mathematics). ISBN 9780538733519. Disponível em: <https://books.google.com.br/books?id=zXnSxY9G2JgC>. Citado 5 vezes nas páginas 34, 35, 36, 37 e 39.

6 SAFINAZ, S.; KUMAR, A. V. R. Vlsi realization of lanczos interpolation for a generic video scaling algorithm. In: *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*. [S.l.: s.n.], 2017. p. 17–23. Citado na página 34.

7 PIAO, Z.-Y. et al. Interpolation circuit design for xrf systems using fft. In: *2011 3rd Asia Symposium on Quality Electronic Design (ASQED)*. [S.l.: s.n.], 2011. p. 236–239. Citado na página 34.

Anexos

ANEXO A – Códigos em linguagem de programação de propósito geral

Código A.1 – Métodos numéricos para busca de valores em Python

```
1  import numpy as np
2
3  def secant(a: float,
4             b: float,
5             SEEK: int,
6             TOL: float,
7             f: callable,
8             MAX_ITER: int,
9             ):
10     """
11     # Description
12     Computes SEEK = f(x) for a given function and a given set of values using a m
13
14     # Parameters
15     @param a: initial lower bound
16     @param b: initial upper bound
17     @param SEEK: Value to be found
18     @param TOL: Error tolerance
19     @param f: The function that best describes the system
20     @param MAX_ITER: Maximum number of operations allowed before convergence
21     """
22
23     converged = False
24     iter = 1
25
26     while (not converged and iter <= MAX_ITER):
27         slope = (f(b) - f(a))/(b - a)
28         c = b - (f(b) - SEEK)/slope
29
30         a, b = b, c
31
```

```
32          converged = not (np.abs(f(c) - SEEK) > TOL)
33          iter += 1
34
35      return c
36
37
38  def mod_secant(a: float,
39                 b: float,
40                 SEEK: int,
41                 TOL: float,
42                 f: callable,
43                 MAX_ITER: int,
44                 alpha: float,
45                 beta: float,
46                 gamma: float
47                 ):
48      """
49      # Description
50      This implementation relies in the specific case of the Q versus I_ref curve. It ca
51
52      # Parameters
53      @param a: initial lower bound in linear region
54      @param b: initial upper bound
55      @param SEEK: Value to be found
56      @param TOL: Error tolerance
57      @param f: The function that best describes the system
58      @param MAX_TER: Maximum number of operations allowed before convergence
59      @param alpha: linear region upper bound
60      @param beta: non-linear region lower bound
61      @param: gamma: Q_d theshold
62      """
63      # Q (SEEK) is in nonlinear region
64      if SEEK > gamma * SEEK:
65          a = beta
66          b = 1.0
67      ## Q (SEEK) is in linear region
68      else:
69          a = 0.0
70          b = alpha
```

```
71
72      return secant(a, b, SEEK, TOL, f, MAX_ITER)
73
74
75  def bisection(a: float,
76                b: float,
77                SEEK: int,
78                TOL: float,
79                f: callable,
80                MAX_ITER: int,
81                ):
82      """
83      # Description
84      Computes SEEK = f(x) for a given function and a given set of values using a m
85
86      # Parameters
87      @param a: initial lower bound
88      @param b: initial upper bound
89      @param SEEK: Value to be found
90      @param TOL: Error tolerance
91      @param f: The function that best describes the system
92      @param MAX_TER: Maximum number of operations allowed before convergence
93      """
94
95      converged = False
96      iter = 1
97
98      while (not converged and iter <= MAX_ITER):
99
100         c = (a+b)/2
101         f_c = f(c)
102
103         if np.abs(SEEK - f_c) <= TOL:
104             converged = True
105         elif (SEEK - f_c) > 0:
106             a = c
107         elif (SEEK - f_c) < 0:
108             b = c
109         else:
```

```
110             converged = True
111
112         iter += 1
113
114     return c
```

ANEXO B – Códigos em linguagem de descrição de hardware

Código B.1 – Módulo de controle do Q: Método das secantes em Verilog

```
module secant #(
  parameter WIDTH = 10, // bus width
  parameter TOL   = 30  // minimum acceptable value of difference between measured
) (
  input                   clk        , // Clock
  input                   rst        , // Async reset
  input                   ready      , // Flag for measurement done
  input  wire [WIDTH-1:0] desired_q  , // Desired Q value
  input  wire [WIDTH-1:0] measured_q , // Measured Q value
  input  wire [WIDTH-1:0] i_ref_setup, // upper bound
  output reg  [WIDTH-1:0] i_ref        // produces Q
);

  // State is necessary for retaining value of 3 initial measurements
  reg [2:0] state;

  // Initial lower and upper bounds and midpoint respectively;
  reg [WIDTH-1:0] a, b, c;

  // Variables for f(x) == Q at x value of I_{REF}
  reg [WIDTH-1:0] f_a, f_b, f_c;

  // [Warning] Register type is not adequate in current implementation
  reg [WIDTH-1:0] slope;

  // flag for achieving convergence
  reg converged;


  // should be SIGNED and WIDTH+1 bit long
  reg signed [WIDTH:0] error;
```

```
32
33  // state updates
34  always @(posedge clk) begin : state_updates
35    if (ready) state <= state+1;
36
37    if (error < TOL) converged <= 1'b1;
38    else converged <= 1'b0;
39
40  end
41
42  always @(state) begin: current_updates
43    case (state)
44      0 : begin
45        i_ref <= a;
46      end
47
48      1 : begin
49        i_ref <= b;
50        f_a   <= measured_q;
51      end
52
53      2 : begin
54        i_ref <= c;
55        f_b   <= measured_q;
56      end
57
58      3 : begin
59        f_c <= measured_q;
60      end
61
62      4 : begin : new_bounds
63        a     <= b;
64        b     <= c;
65        state <= -1;
66      end
67    endcase
68
69  end
70
```

```
71    always @(state) begin : blocking_updates
72      if (state == 3) begin
73
74        slope = (f_b - f_a)/ (b-a);
75        // slope could be zero if is less than 1 in binary.
76        if (!slope) begin
77          c = b - (f_b - desired_q);
78        end
79        else
80          c = b - (f_b - desired_q) / slope;
81      end
82    end
83
84    always @* begin
85      // calculating error (\epsilon) and taking the absolute value
86      error = f_c - desired_q;
87      error = (error > 0) ? error : -error;
88    end
89
90  endmodule
```

Código B.2 – Módulo de controle do Q: Método da bisseção em Verilog

```
1   module bisection #(
2     parameter WIDTH = 10, // bus width
3     parameter TOL   = 30  // minimum acceptable value of difference between measured
4   ) (
5     input                  clk        , // Clock
6     input                  rst        , // Async reset
7     input                  ready      , // Flag for measurement done
8     input  wire [WIDTH-1:0] desired_q  , // Desired Q value
9     input  wire [WIDTH-1:0] measured_q , // Measured Q value
10    input  wire [WIDTH-1:0] i_ref_setup, // upper bound
11    output reg  [WIDTH-1:0] i_ref        // produces Q
12  );
13
14    // Initial lower and upper bounds and midpoint respectively;
15    reg [WIDTH-1:0] a, b, c;
16
```

```
17  // flag for achieving convergence
18  reg converged;
19
20  // should be SIGNED and WIDTH+1 bit long
21  reg signed [WIDTH:0] error;
22
23  always @(posedge clk or posedge rst) begin: bisection
24    if (rst) begin
25      a         = 0;
26      b         = 2**WIDTH-1;
27      converged = 1'b0;
28    end
29
30    if(!ready) c <= 0;
31    else if (!converged) begin
32
33      // finding midpoint
34      c <= (a+b)/2;
35
36      if (error < TOL) converged <= 1'b1;
37      // desired value is between a and c
38      else if (desired_q > measured_q) a <= c;
39      // desired value is between b and c
40      else if (desired_q < measured_q) b <= c;
41      // convergence achieved
42      else converged <= 1'b0;
43    end
44  end
45
46  always @* begin
47    // updating reference current with midpoint
48    i_ref <= c;
49  end
50
51  always @* begin
52    // calculating error (\epsilon) and taking the absolute value
53    error = measured_q - desired_q;
54    error = (error > 0) ? error : -error;
55  end
```

```
56
57 endmodule
```

Código B.3 – Módulo de determinação da instabilidade em Verilog

```
1  module instability_detect #(
2    parameter WIDTH      = 10 , // Bus Width
3    parameter DELTA      = 100, // $\Delta Q $
4    parameter IREF_DELTA = 50   // $\Delta I_{REF} $
5  ) (
6    input                  clk        , // Clock
7    input                  rst        , // Async reset
8    input                  ready      , // Flag for measurement done
9    input      [WIDTH-1:0] q_measured , // from Q measurement block
10   output reg [WIDTH-1:0] i_ref_setup  // upper bound for Q control module
11 );
12
13   // flag for finding the instability point
14   reg found = 1'b0;
15
16   // registers to store Q values
17   reg [WIDTH-1:0] curr_q, last_q;
18
19   // asynchronous setup/reset
20   always @ (posedge rst) begin: async_setup
21     if (rst) i_ref_setup = 2**WIDTH-1; // set all bits
22     else i_ref_setup = 0;              // clear all bits
23   end
24
25
26   //nonblocking assignments for $Q$ measurement scheduled updates
27   always @(posedge clk) begin: q_updates
28     if (ready) begin
29       if (!found) begin
30         last_q <= curr_q;
31         curr_q <= q_measured;
32       end
33     end
34
```

```
35    end
36
37    // blocking assignments for $\Delta Q$ calculations
38    always @(negedge clk) begin : delta_q_calc
39      if ((curr_q - last_q) > DELTA) begin
40        found = 1'b1;
41      end
42      else found = 1'b0;
43
44      // update current value on falling edge
45      if (!found)  i_ref_setup = i_ref_setup - IREF_DELTA;
46      // retain $I_{REF}|_{Setup}$ value for Q control block upper bound
47      else i_ref_setup = i_ref_setup;
48
49      // end
50    end
51
52    //TODO: Implement some optimization strategy to obtain the maximum Q
53
54
55  endmodule
```