Implementação de Circuitos de Sincronismo Monofásicos Utilizando FPGA ⋆

Fabiano da Silva Catão * **Dayane Mendonça Lessa** **
Cleiton Magalhães Freitas * **Michel Pompeu Tcheou** **
Luís Fernando Corrêa Monteiro **

* *Departamento de Engenharia Elétrica, Universidade do Estado do Rio de Janeiro, RJ, (e-mails: catao.fabiano@graduacao.uerj.br, cleiton.freitas@uerj.br).*
** *Programa de Pós-graduação em Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, RJ, (e-mails: day_lessa@yahoo.com.br, mtcheou@uerj.br, lmonteiro@uerj.br).*

Abstract: PLLs are synchronism circuits used in different applications ranging from signal processing to real-time algorithms applied to active power conditioners. The PLLs used in single-phase circuits can be composed by the association of quadrature signal generating circuits, such as the second-order generalized integrator (SOGI) or the all-pass filter (APF). The association of single-phase PLLs to quadrature signal generators results in a similar performance in comparison to three-phase PLLs, with no transient double-frequency oscillation. In this context, this work exploits the digital implementation in FPGA of SOGI-PLL and APF-PLL circuits through an approach based on high level synthesis (implemented in C++). A fixed-point architecture was applied to implement the PLL algorithms in 2MHz interrupts. In this case, the developed IPs (Intellectual Property) represented each of the PLLs through the included tools in the Xilinx Software VIVADO 2020.2. In this paper, there are simulation and experimental results through the FPGA. In both cases, the proposed digital implementation tracked the input signal over a period of approximately 0.06s.

Resumo: Os PLLs são circuitos de sincronismo utilizados em diferentes aplicações que vão desde o processamento de sinais aos algoritmos em tempo real utilizados no acionamento dos condicionadores ativos de energia. Os PLLs usados em sistemas monofásicos podem ser compostos pela associação de circuitos geradores de sinal de quadratura, como o integrador generalizado de segunda ordem (SOGI) ou o filtro passa tudo (APF). A associação dos PLLs monofásicos aos geradores de sinal de quadratura faz com que estes tenham comportamento similar aos PLLs trifásicos, que não apresentam o problema da oscilação de dupla frequência. Neste contexto, o trabalho apresenta uma discussão a respeito da implementação digital em FPGA dos circuitos SOGI-PLL e APF-PLL utilizado uma abordagem baseada em síntese de alto nível (codificação feita em C++). Na implementação proposta, os algoritmos dos PLLs estudados são implementados utilizando uma arquitetura de ponto fixo, com 64 bits, que processa os algoritmos em interrupções de 2MHz. Neste caso, foram desenvolvidos IPs (Intellectual Property) para representar cada um dos PLLs através das ferramentas incluídas no Software Xilinx VIVADO 2020.2. Para validar a proposta de implementação em FPGA, são apresentados resultados de simulação e resultados experimentais. Em ambos os casos, a implementação digital proposta foi capaz de rastrear o sinal de entrada em um período de aproximadamente 0,06s.

Keywords: Phase-locked loop; Quadrature signal generator; Single-phase PLL; Second-order generalized integrator; All-pass filter; FPGA.

Palavras-chaves: Phase-locked loop; Gerador de sinal de quadratura; PLL monofásico; Integrador generalizado de segunda ordem; Filtro passa tudo; FPGA.

1. INTRODUÇÃO

Os circuitos de sincronismo do tipo PLL (Phase-Locked Loop) são essenciais para instrumentação, controle e processamento de sinais de sistemas relacionados às redes elétricas e integração de conversores eletrônicos de potência com ela. Entre os exemplos, podemos citar o controle de sistemas de geração distribuída, a identificação e a caracterização de fenômenos relacionados à qualidade de energia, a localização de pontos da rede onde ocorreram curto-circuitos, entre outros (Lessa, 2019).

Em circuitos trifásicos, os PLLs tradicionais são baseados na estrutura em referencial síncrono (SRF, do inglês *Synchronous Reference Frame*) obtida com a transformação de Park. Em circuitos monofásicos, no entanto, esta metodologia possui uma limitação, pois não há o sinal de quadratura em relação ao sinal de entrada. Este problema é resolvido com a utilização de circuitos de geração de sinal de quadratura (QSG do inglês *Quadrature Signal Generators*) (Han et al., 2016) possibilitando assim a implantação do PLL baseado em SRF. Os PLLs baseados em sinal de quadratura, QSG-PLLs, atuam com um comportamento similar ao comportamento dos PLLs trifásicos, fazendo com que a componente oscilante de segunda harmônica seja eliminada assim que a frequência do sinal de entrada é corretamente identificada.

Dentre as estruturas de geração de sinal de quadratura utilizadas em PLLs, o integrador generalizado de segunda ordem (SOGI, do inglês *Second-Order Generalized Integrator*) e o filtro passa-tudo (APF, do inglês *All-Pass Filter*) se destacam. O SOGI-PLL, proposto por Ciobotaru et al. (2006), utiliza em sua malha um SOGI cuja frequência é obtida a partir da malha de controle do PLL em SRF. Além da geração do sinal de quadratura, o SOGI neste PLL ainda atua filtrando o sinal da rede e assim reduzindo a influência de possíveis distorções harmônicas no comportamento do PLL. O APF-PLL, por outro lado, primeiramente apresentado por Jung et al. (2004) e analisado por completo em (Golestan et al., 2020), utiliza um circuito de geração de sinal de quadratura de primeira ordem, um filtro passa tudo. Este tipo de filtro linear permite a passagem de todas as frequências com o mesmo ganho, porém altera a relação de fase entre elas. Assim como o SOGI, ele gera dinamicamente o componente ortogonal. É importante mencionar que ambos os PLLs explorados neste trabalho ainda se encontram em estágio de contínuo estudo e aprimoramento, como pode ser comprovado por publicações recentes como (Prakash et al., 2021) e (Singh et al., 2022).

Analisando literatura científica de uma maneira mais ampla, encontramos trabalhos comparando os diferentes tipos de PLLs monofásicos (Xu et al., 2022), propondo melhorias nos PLLS apresentados em (Mohamadian et al., 2022), analisando-se a inclusão de filtros *Notch* (Lessa et al., 2021) ou a implementação utilizando sistemas de janelamento de sinal (Golestan et al., 2018). Também é de grande interesse o estudo da implementação digital desses PLLs, em especial a implementação utilizando circuitos digitais reconfiguráveis como FPGAs. Nesta linha, Cossutta et al. (2015) apresenta a implementação em FPGA do SOGI-PLL a partir da ferramenta *System Generator* do programa *Matlab - Simulink*. Em sua abordagem, o PLL é implementado considerando-se uma estrutura de 16 bits a uma taxa de amostragem de 500kHz. Carugati et al. (2011), por outro lado, propôs um PLL trifásico utilizando um filtro do tipo *Notch* para garantir imunidade à distorção harmônica de tensões da rede. A implementação digital também foi realizada em FPGA, mas neste caso, os autores desenvolveram o PLL utilizando uma linguagem de descrição de hardware, VHDL para ser mais preciso. Xu et al. (2013), por sua vez, apresentou implementações de duas estruturas de PLL monofásico em FPGA, entre elas o EPLL Karimi-Ghartemani and Iravani (2002). Sun et al. (2011) também trabalhou com implementação em FPGA de PLLs, neste caso trifásico, e visando suporte ao controle de filtros ativos. O PLL em questão é conhecido como PLL desacoplado em frequência dupla, em tradução livre, e garante imunidade tanto à distorção harmônica quanto ao desequilíbrio nas tensões. Jo et al. (2015) também trabalhou com implementação em FPGA de PLLs trifásicos, contudo o PLL estudado é baseado na técnica conhecida como cancelamento de sinal atrasado Wang and Li (2011), também em tradução livre.

Neste contexto, o trabalho proposto consiste em apresentar a implementação digital em FPGA de PLLs monofásicos dos tipos SOGI-PLL e APF-PLL. A abordagem utilizada se baseia na criação de blocos IPs (do inglês *Intelectual Property* (Khvatov and Zheleznikov, 2021)) que implementam um sistema de interrupções e computam as variáveis de estado a partir dos respectivos modelos matemáticos. O artigo apresenta resultados de simulação utilizando o software Xilinx VIVADO 2020.2, e resultados experimentais, onde os *hardwares* sintetizados a partir dos IPs desenvolvidos foram implantados na placa de desenvolvimento *Xilinx Artix-7 FPGA AC701 Evaluation Kit*. Em todos os casos, os PLLs apresentados conseguiram rastrear a fase, frequência e amplitude do sinal de entrada em um período inferior a três ciclos de onda referente ao sinal de entrada.

2. MODELO DINÂMICO DO PLL

A Figura 1 contém os diagramas de blocos dos geradores de sinais ortogonais representados pelo APF na Figura 1(a) e pelo SOGI na Figura 1(b). A Figura 1(c) consiste no diagrama de blocos do PLL utilizado, com a inclusão de uma malha para extrair o valor da amplitude da tensão $v_{in}(t)$. Considerando que o sinal de entrada é composto tão somente pela componente fundamental, as saídas do APF e do SOGI ficam atrasadas de 90° em relação ao sinal de entrada. No trabalho de Lessa et al. (2021) estão as equações dinâmicas destes geradores de sinais ortogonais. Assim, assumindo que o PLL extraiu corretamente o valor da frequência angular da tensão $v_{in}(t)$, os sinais $v_\alpha(t)$ e $v_\beta(t)$ são dados por:

$$\begin{cases} v_\alpha(t) = v_{in}(t) = V \cdot sen(\theta_1) \\ v_\beta(t) = V \cdot sen(\theta_1 - 90°) \end{cases} ; \quad (1)$$

sendo $\theta_1 = \omega_1 t$, onde ω consiste na frequência angular da tensão de entrada $v_{in}(t)$. O PLL utilizado con-

⋆ O trabalho foi financiado com fundos do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Processo 422792/2016-0) e com bolsas de mestrado (CAPES) e iniciação científica (PIBIC-UERJ).

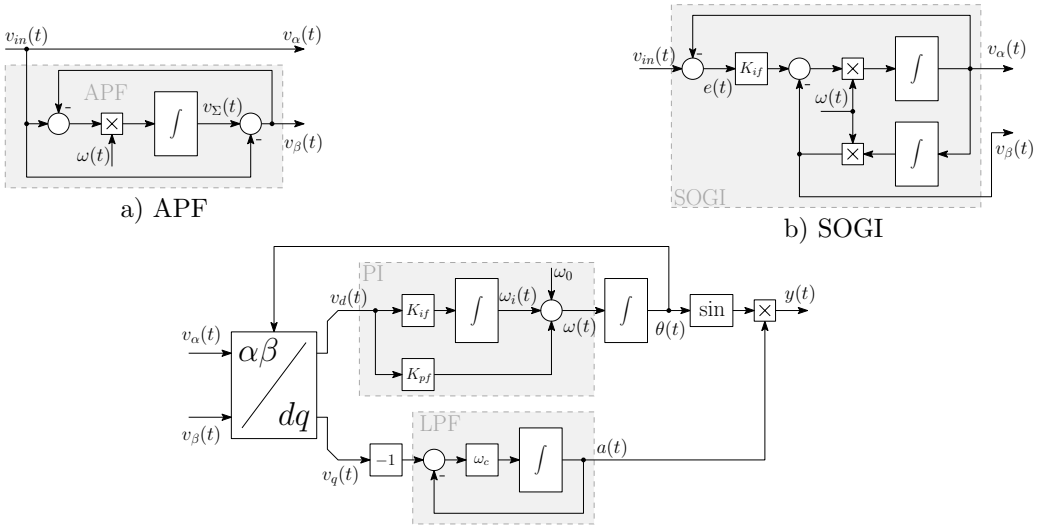


Figura 1. Diagrama esquemático dos PLLs utilizados neste trabalho.

tém malhas de fase e frequência, bem como de amplitude, independentes entre si, onde os sinais de entrada destas malhas são determinados pela transformada *dq*-modificada. É importante comentar que trata-se de um circuito de sincronismo diferente do E-PLL original apresentado por Karimi-Ghartemani et al. (2004), uma vez que não faz uso do sinal de saída, $y(t)$, para os cálculos dos erros utilizados tanto na malha de fase e frequência quanto na malha de amplitude. A transformada *dq* utilizada é dada na forma:

$$\begin{bmatrix} v_d(t) \\ v_q(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} v_\alpha(t) \\ v_\beta(t) \end{bmatrix}. \tag{2}$$

A malha de fase e frequência utiliza um controlador do tipo PI para a determinação da frequência angular $\omega(t)$. A sua entrada consiste no sinal $v_d(t)$ representado na forma:

$$v_d(t) = V \cdot [cos(\theta) \cdot sen(\theta_1) + sen(\theta) \cdot sen(\theta_1 - 90°)] \ ; \tag{3}$$

onde valor médio de $v_d(t)$ decai para zero à medida que o sinal $\theta(t)$ converge para θ_1. Por outro lado o sinal $v_q(t)$ obtido na malha de amplitude é dado por:

$$v_q(t) = V \cdot [-sen(\theta) \cdot sen(\theta_1) + cos(\theta) \cdot sen(\theta_1 - 90°)] \ . \tag{4}$$

À medida que θ converge para θ_1, $v_q(t) = -V \cdot \left[sen^2(\theta) + cos^2(\theta)\right] = -V$. Assim, a malha de amplitude pode ser simplificada na forma:

$$a(t) = \int (V - a) \cdot (\omega_c) \ dt. \tag{5}$$

O valor de $a(t)$ é dinamicamente ajustado até a condição de $(V - a) = 0$. O parâmetro ω_c foi utilizado para reduzir o tempo de convergência da malha de amplitude, que pode ser entendida como um filtro do tipo passa-baixas a partir da sua função de transferência:

$$A(s) = V \cdot \frac{\omega_c}{s + \omega_c} \ . \tag{6}$$

Desta forma, o PLL atinge a condição de regime permanente somente após o sinal A convergir para a amplitude do sinal de entrada V. Em regime permanente, o sinal de saída, $y(t)$, é dado por $V \cdot sen(\theta)$, que corresponde à tensão de entrada, $v_{in}(t)$.

Conforme descrito na Introdução, uma característica importante do PLL utilizado neste trabalho, na condição da tensão $v_{in}(t)$ não conter distorções harmônicas, consiste na sua capacidade de mitigar oscilações transitórias de segundo harmônico. Estas oscilações ocorrem em circuitos de sincronismos monofásicos, mesmo quando a tensão de entrada não contém distorções harmônicas.

3. METODOLOGIA DE IMPLEMENTAÇÃO EM FPGA

Antes de discutir a metodologia de implementação, precisamos entender como o FPGA realiza computação em geral. Para isso, considere o Algoritmo 1 e a sequência de passos que o FPGA segue para implementá-lo mostrada na Figura 2. A primeira instrução exige dois ciclos de máquina, uma para computar $x_1 + y_1$ e armazenar o resultado em um registrador temporário R_1 e outra para atualizar o valor de x_1. A segunda instrução, $z_1 \leftarrow x_1 + 1$, só é executada após a realização completa da instrução anterior. Sendo assim, o FPGA consome três ciclos de máquina nas duas primeiras instruções do algoritmo. A terceira instrução não depende de variáveis presentes nas outras e, diferentemente do que ocorre em um microcontrolador, pode ser executada em paralelo com as outras instruções sem alterar o tempo total de computação do FPGA. Ainda assim, o FPGA necessita de um intervalo de $3\Delta t$ para executar o algoritmo, sendo Δt o período do *clock* utilizado. É importante mencionar que operações mais complexas, como implementação de funções trigonométricas, exigem um número maior de ciclos e aumentam consideravelmente o tempo de computação do algoritmo.

Tendo em vista a explicação do parágrafo anterior, foi adotada uma arquitetura baseada em interrupções, como mostrado na Figura 3. Neste caso, os sinais *clk* e clk_{it} representam respectivamente o *clock* interno do FPGA e o *clock* utilizado para interrupção. O primeiro, *clk*, indica o intervalo Δt em que os ciclos de instruções simples são executados e o segundo, clk_{it}, a frequência em que a função PLL é chamada. Para evitar sobreposição de cálculos e manter a taxa de processamento constante, a frequência de clk_{it} deve ser cuidadosamente escolhida de forma que $T_s > n_{PLL}\Delta t$ onde T_s é o período de amostragem do sistema, e n_{PLL}, o número de ciclos de máquina necessários para execução completa do algoritmo do PLL. Para reduzir o número de ciclos de máquina sem aumentar significativamente a quantidade de hardware utilizado, as funções trigonométricas utilizadas nos PLLs foram obtidas através da *Look-up Table* **LUT TRIG**, também indicada na Figura 3.

Algorithm 1 Algoritmo genérico

$x_1 \leftarrow x_1 + y_1$

$z_1 \leftarrow x_1 + 1$

$x_2 \leftarrow x_2 + y_2$

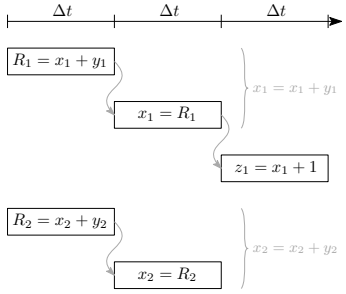


Figura 2. Exemplo de cadeia de processamento no FPGA.

Finalmente, as equações a diferenças dos PLLs foram codificadas em C++ considerando variáveis de ponto fixo de 64 bits (1 bit de sinal, 15 para a parte inteira e 48 para a parte fracionária), garantindo precisão de $2^{-48} \approx 3.5 \times 10^{-15}$. Utilizando o módulo de síntese em alto nível do Xilinx VIVADO 2020.2, foram criados dois IPs, um para cada PLL, englobando a estrutura de interrupção e o algoritmo do PLL. Os fluxogramas destes PLLs são mostrados nas Figuras 4(a) e 4(b). Como mencionado anteriormente, a implementação foi realizada em C++ e não será apresentada aqui por questão de espaço. No entanto, os códigos fontes estão disponíveis para análise e/ou utilização em Freitas et al. (2022). De uma forma geral, cada PLL cotem um arquivo cabeçalho (`.h`) onde as bibliotecas `ap_fixed.h` e `ap_int.h`, necessárias para representação de variáveis dos tipos ponto flutuante e binária, são incluídas e as principais definições apresentadas. Os PLLs são definidos nas funções `APF_SRF_PLL_x64` e `SOGI_SRF_PLL_x64`, implementadas no arquivo `.cpp`. Seguido a lógica indicada nos fluxogramas da Figura 4, estas funções executam as equações a diferença dos PLLs sempre que ocorre uma transição no clock clk_{it}. Os arquivos `.cpp` também contém as funções `sin_2000` e `cos_2000`, que implementam as LUTs de seno e cosseno. Mais precisamente, `sin_2000` armazena os valores do seno para 250 diferentes argumentos no intervalo $[0,\ \pi/4]$. Através de trigonometria básica, estes valores são extrapolados de que forma que a LUT permita obter o seno para 2000 diferentes valores de argumento no intervalo de $[0,\ 2\pi]$. A função `cos_2000`, por sua vez, utiliza a função `sin_2000` para calcular indiretamente (também usando trigonometria básica) o cosseno, sem a necessidade de armazenar novos valores em memória. Para ambos os PLLs, APF e SOGI, o período para execução dos algoritmos, ou latência, foi de 33 ciclos. Por este motivo a frequência de interrupção foi ajustada para um cinquenta avos da frequência de *clock* do sistema, garantindo um período de amostragem de $500ns$. A Figura 4(c) mostra o diagrama de blocos unindo os IPs utilizados para implementação do FPGA. Observe que um IP é utilizado para processar a entrada `sys_clk` da placa e fornecer dois sinais ao sistema, um de $100Mz$ (clk), utilizado como clock do FPGA, e outro de $10MHz$ (clk_2). O IP divisor de frequência atua sobre este último sinal e retorna o *clock* de $1MHz$ (clk_{it}) utilizado como gatilho para a interrupção do PLL. É importante mencionar que a interrupção foi programada para ocorrer nas duas transições, garantindo uma frequência de processamento equivalente a $2MHz$. Para simplificar os testes, foi utilizado um IP gerador senoidal em vez de utilizar-se uma das entradas analógicas da placa de desenvolvimento. Este gerador foi configurado para produzir um sinal de $60Hz$ e $5V$ de amplitude, como indicado nos blocos da Figura 4(c). Para exportar os sinais, foi desenvolvido um IP conversor digital analógico (DAC) baseado em modulação por largura de pulso. Nesta figura ele se encontra conectado ao sinal de saída y, mas também foram incluídas saídas analógicas para os sinais v_{in}, ω e θ.

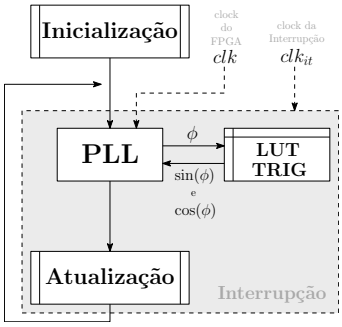


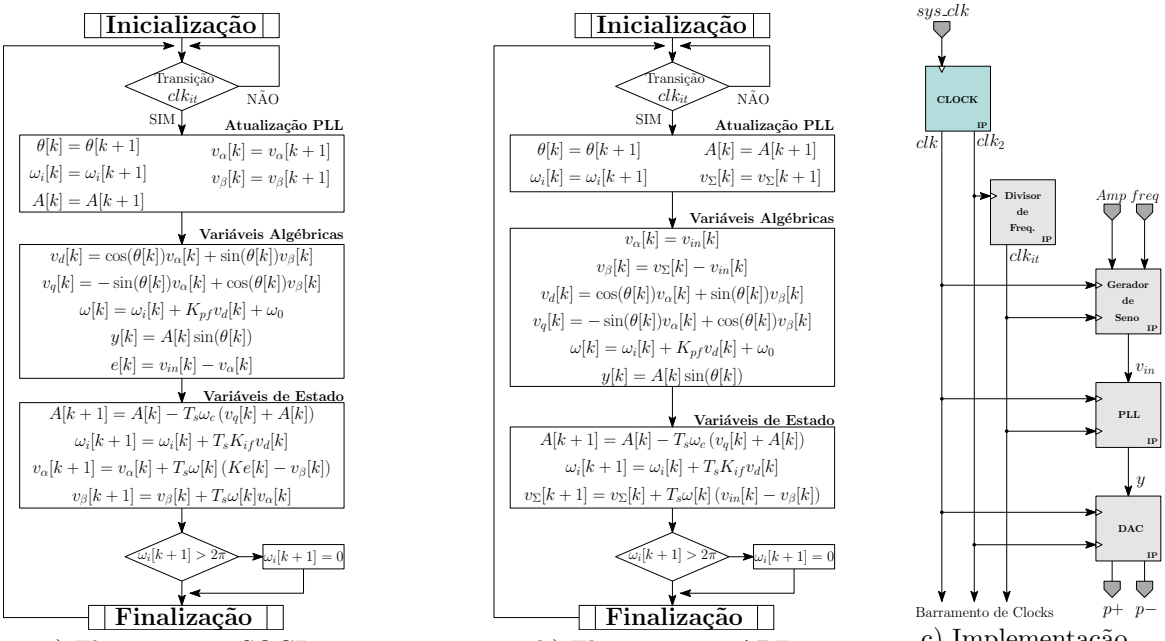
Figura 3. Arquitetura de implementação em FPGA adotada.

4. RESULTADOS DE SIMULAÇÃO

Nesta seção, são apresentados alguns resultados de simulação para verificar o funcionamento dos PLLs com os parâmetros de controle utilizados. Neste sentido, foram realizados dois conjuntos de simulação, um utilizando pacote Python `py1phiPLL` Freitas (2022) e outro utilizando o próprio ambiente de simulação do software Xilinx VIVADO 2020.2. A Tabela 1 apresenta os valores utilizados para as constantes dos PLLs.

Tabela 1. Configurações does parâmetros dos PLLs.

	K	K_{if}	K_{pf}	ω_c	ω_0
APF-PLL	-	1000	30	120	300
SOGI-PLL	2	1000	30	120	300



a) Fluxograma - SOGI b) Fluxograma - APF c) Implementação FPGA

Figura 4. Implementação dos PLLs.

Da Figura 5 à Figura 8 apresentam-se os comportamentos dinâmicos das frequências rastreadas pelos dois PLLs. Em ambos os casos ω foi inicializado em 300rad/s e após um transitório de cerca de 0,06s, atingiu-se a condição de regime permanente, $120\pi \approx 377 rad/s$. Também é possível notar que os resultados das simulações envolvendo

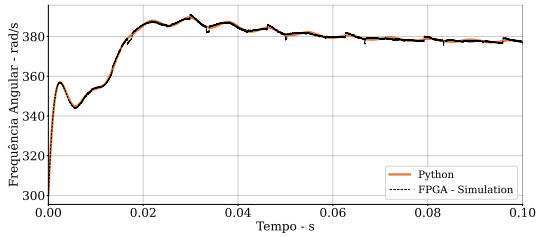


Figura 5. Resultado de simulação do APF-PLL: frequência angular ω

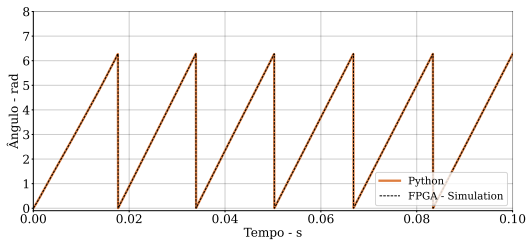


Figura 6. Resultado de simulação do APF-PLL: ângulo elétrico θ

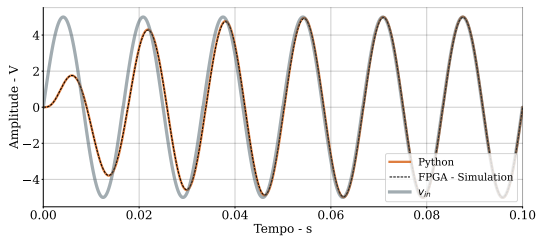


Figura 7. Resultado de simulação do APF-PLL: comparação entre sinal de entrada v_{in} e o de saída y

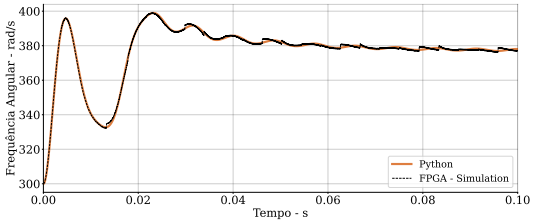


Figura 8. Resultado de simulação do SOGI-PLL: frequência angular ω

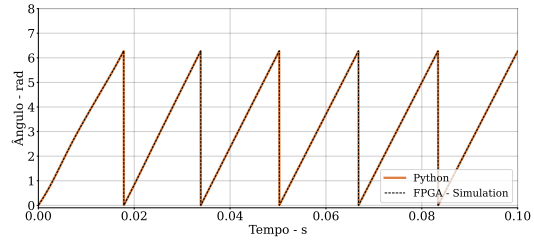


Figura 9. Resultado de simulação do SOGI-PLL: ângulo elétrico θ

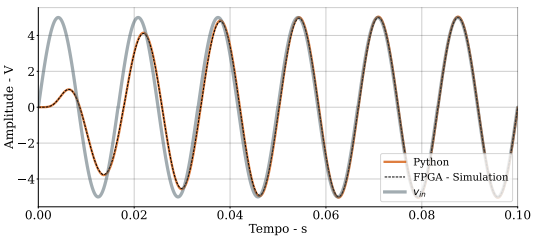


Figura 10. Resultado de simulação do SOGI-PLL: comparação entre sinal de entrada v_{in} e o de saída y

os IPs desenvolvidos foram equivalentes às simulações realizadas em Python, embora se note a ocorrência de pequenos *spikes* nos resultados do VIVADO. Estes *spikes* são causados pelo processo de *reset* da variável θ e, em geral, não causaram grandes problemas ao sistema.

O comportamento do ângulo elétrico rastreado θ também foi similar nas diferentes simulações, como pode ser observado nas Figuras 6 e 9. Neste caso, não existe nenhuma diferença observável entre as implementações com síntese digital de alto nível (VIVADO) e com Python.

Por fim, as Figuras 7 e 10 apresentam os sinais senoidais de entrada e saída. Como visto nos outros resultados, os PLLs rastrearam corretamente os sinais de entrada em pouco mais de três ciclos de onda. Novamente os resultados obtidos com os IPs gerados no software VIVADO foram exatamente iguais à simulação convencional em Python.

5. RESULTADOS EXPERIMENTAIS

O diagrama de blocos da Figura 4 foi sintetizado, implementado e gravado na placa de desenvolvimento *Xilinx Artix-7 FPGA AC701 Evaluation Kit*. A Figura 11 apresenta o *layout* considerado para os experimentos. Neste caso, os sinais de saída PWM são processados por circuitos de filtragem do tipo RC. O PWM em si foi configurado para 200kHz, mas a frequência equivalente da portadora foi de 400kHz, visto que os sinais foram disponibilizados de forma diferencial usando duas saídas PWM cada. Além disso, o IP modulador PWM foi desenvolvido considerando 50 posições, garantindo uma precisão de 2% para sinais CA y e v_{in}. Os sinais CC θ e ω, no entanto, apresentam uma precisão menor, visto que não foram escalonadas para ocupar toda a faixa de modulação. A frequência de corte do filtro, por sua vez, foi configurada em 10,6kHz de forma que as dinâmicas dos PLLs não fossem afetadas de forma significativa pelo processo de filtragem.

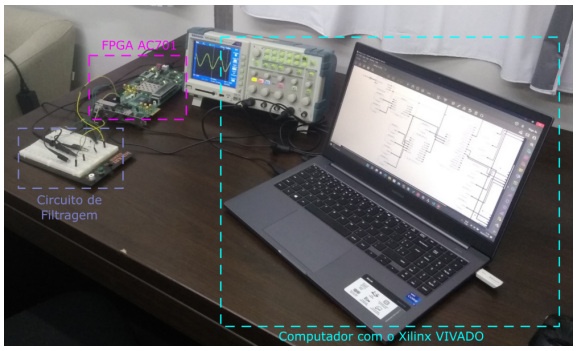


Figura 11. *Layout* utilizado para realização dos experimentos.

As figuras 12 e 13 apresentam a fase e a frequência angular rastreados pelos PLLs estudados. É possível observar que o sistema convergiu como esperado, embora o período transitório de ω pareça ser menor que o obtido nas simulações. A este respeito, é importante ter em mente que o processo de conversão digital-analógica desta variável foi pouco preciso, uma vez que o sinal varia entre 300 e 377rad/s e a portadora da modulação foi ajustada com amplitude de 400rad/s.

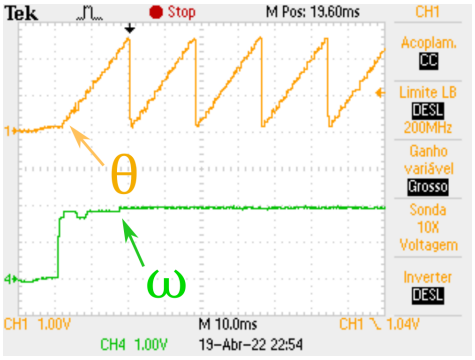


Figura 12. Resultado experimental do APF-PLL: fase θ e frequência angular ω

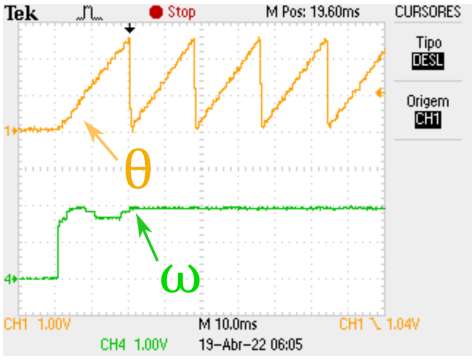


Figura 13. Resultado experimental do SOGI-PLL: fase θ e frequência angular ω

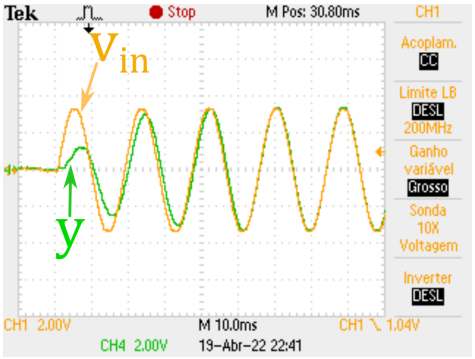


Figura 14. Resultado Experimental do APF-PLL: comparação entre sinal de entrada v_{in} e o de saída y

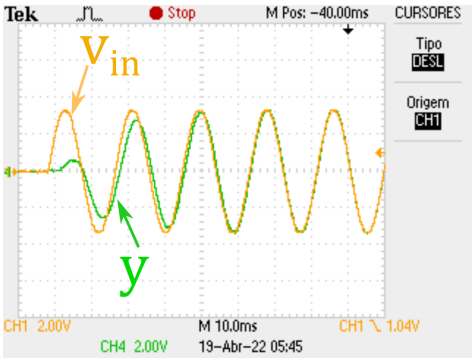


Figura 15. Resultado Experimental do SOGI-PLL: comparação entre sinal de entrada v_{in} e o de saída y

As figuras 14 e 15 apresentam as comparações entre os sinais de entrada e os sinais senoidais produzidos pelos PLLs. Para estes sinais, o processo de conversão digital-analógica foi bem preciso e os resultados experimentais seguiram os padrões observados nas simulações. Assim, os períodos transitórios de ambos os PLLs se aproximaram de 0.06s.

6. CONCLUSÕES

Este trabalho apresentou uma metodologia de implementação digital em FPGA para circuitos de sincronismo monofásicos dos tipos SOGI-PLL e APF-PLL. Com esta metodologia foi possível executar os algoritmos com taxa de amostragem de 2MHz (período de 500ns) utilizando variáveis de ponto fixo com 64 bits e precisão de 3.5×10^{-15}. Os resultados experimentais, obtidos a partir da leitura das saídas analógicas programadas no FPGA, foram consistentes em relação aos resultados de simulação, também apresentados no trabalho. Neste caso, em ambos os PLLs o tempo de convergência foi inferior ao período de três ciclos de onda.

AGRADECIMENTOS

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Processo 422792/2016-0) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

REFERÊNCIAS

Carugati, I., Orallo, C., Donato, P.G., and Maestri, S. (2011). Fpga design of a variable sampling period pll with a digital notch filter for distorted grids. In *2011 Argentine School of Micro-Nanoelectronics, Technology and Applications*, 1–7.

Ciobotaru, M., Teodorescu, R., and Blaabjerg, F. (2006). A new single-phase pll structure based on second order generalized integrator. In *2006 37th IEEE Power Electronics Specialists Conference*, 1–6. doi:10.1109/pesc.2006.1711988.

Cossutta, P., Raffo, S., Cao, A., Ditaranto, F., Aguirre, M.P., and Valla, M.I. (2015). High speed single phase sogi-pll with high resolution implementation on an fpga. In *2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)*, 1004–1009. doi:10.1109/ISIE.2015.7281609.

Freitas, C. (2022). Python Package for simulating single-phase PLLs. URL `https://github.com/cleitoncmf/py1phiPLL`.

Freitas, C.M., Tcheou, M.P., Monteiro, L.F.C., Catão, F.d.S., and Lessa, D.M. (2022). CBA-2022 - Código fonte em c++ para implementação de PLLs monofásicos em FPGA. URL `https://github.com/cleitoncmf/CBA_2022_PLL_CPP_FILES`.

Golestan, S., Guerrero, J.M., and Vasquez, J.C. (2018). A nonadaptive window-based pll for single-phase applications. *IEEE Transactions on Power Electronics*, 33(1), 24–31. doi:10.1109/TPEL.2017.2713379.

Golestan, S., Guerrero, J.M., Vasquez, J.C., Abusorrah, A.M., and Al-Turki, Y. (2020). All-pass-filter-based pll systems: Linear modeling, analysis, and comparative evaluation. *IEEE Transactions on Power Electronics*, 35(4), 3558–3572. doi:10.1109/TPEL.2019.2937936.

Han, Y., Luo, M., Zhao, X., Guerrero, J.M., and Xu, L. (2016). Comparative performance evaluation of orthogonal-signal-generators-based single-phase pll algorithms—a survey. *IEEE Transactions on Power Electronics*, 31(5), 3932–3944. doi:10.1109/TPEL.2015.2466631.

Jo, J., Han, B.M., and Cha, H. (2015). Fpga based dsc-pll for grid harmonics and voltage unbalance effect elimination. In *2015 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2212–2216. doi:10.1109/APEC.2015.7104656.

Jung, Y., Sol, J., Yu, G., and Choj, J. (2004). Modelling and analysis of active islanding detection methods for photovoltaic power conditioning systems. In *Canadian Conference on Electrical and Computer Engineering 2004*, volume 2, 979–982 Vol.2. doi:10.1109/CCECE.2004.1345280.

Karimi-Ghartemani, M. and Iravani, M. (2002). A nonlinear adaptive filter for online signal analysis in power systems: applications. *IEEE Transactions on Power Delivery*, 17(2), 617–622. doi:10.1109/61.997949.

Karimi-Ghartemani, M., Karimi, H., and Iravani, M.R. (2004). A magnitude/phase-locked loop system based on estimation of frequency and in-phase/quadrature-phase amplitudes. *IEEE Transactions on Industrial Electronics*, 51(2), 511–517. doi:10.1109/TIE.2004.825282.

Khvatov, V.M. and Zheleznikov, D.A. (2021). Development of an ip-cores libraries as part of the design flow of integrated circuits on fpga. In *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 2686–2691. doi:10.1109/ElConRus51938.2021.9396219.

Lessa, D.M., Tcheou, M.P., Freitas, C.M., and Monteiro, L.F.C. (2021). Mitigation of Low-Frequency Oscillations by Tuning Single-Phase Phase-Locked Loop Circuits. In J.L. Afonso, V. Monteiro, and J.G. Pinto (eds.), *Sustainable Energy for Smart Cities*, volume 375, 132–151. Springer International Publishing, Cham. doi:10.1007/978-3-030-73585-2_9. URL `https://link.springer.com/10.1007/978-3-030-73585-2_9`. Series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.

Lessa, D.M. (2019). *Uso combinado de filtros digitais com circuitos de sincronismo monofásicos*. Master's thesis, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. URL `https://www.pel.uerj.br/bancodissertacoes/Dissertacao_Dayane_Lessa.pdf`.

Mohamadian, S., Pairo, H., and Ghasemian, A. (2022). A straightforward quadrature signal generator for single-phase sogi-pll with low susceptibility to grid harmonics. *IEEE Transactions on Industrial Electronics*, 69(7), 6997–7007. doi:10.1109/TIE.2021.3095813.

Prakash, S., Singh, J.K., Behera, R.K., and Mondal, A. (2021). A type-3 modified sogi-pll with grid disturbance rejection capability for single-phase grid-tied converters. *IEEE Transactions on Industry Applications*, 57(4), 4242–4252. doi:10.1109/TIA.2021.3079122.

Singh, J.K., Prakash, S., and Behera, R.K. (2022). A nonlinear loop filter based pll with harmonic filtering capability for single-phase grid integrated system with

improved dynamic performance. *IEEE Transactions on Power Delivery*, 1–1. doi:10.1109/TPWRD.2022.3162071.

Sun, B., Dai, N.Y., Chio, U.F., Wong, M.C., Wong, C.K., Sin, S.W., Seng-Pan, U., and Martins, R.P. (2011). Fpga-based decoupled double synchronous reference frame pll for active power filters. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, 2145–2150. doi:10.1109/ICIEA.2011.5975946.

Wang, Y.F. and Li, Y.W. (2011). Grid synchronization pll based on cascaded delayed signal cancellation. *IEEE Transactions on Power Electronics*, 26(7), 1987–1997. doi:10.1109/TPEL.2010.2099669.

Xu, J., Qian, H., Bian, S., Hu, Y., and Xie, S. (2022). Comparative study of single-phase phase-locked loops for grid-connected inverters under non-ideal grid conditions. *CSEE Journal of Power and Energy Systems*, 8(1), 155–164. doi:10.17775/CSEEJPES.2019.02390.

Xu, Z.x., He, X.q., Shu, Z.l., Wang, Y., and Zhou, Y.y. (2013). Implementation of single-phase pll based on fpga. In *2013 International Conference on Mechanical and Automation Engineering*, 168–171. doi:10.1109/MAEE.2013.50.