



**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

Dispositivos Lógicos Programáveis II

Implementação de PLL para Relógio Digital (Milisegundos)

Arthur Cadore Matuella Barcella e Gabriel Luiz Espindola Pedro

23 de Abril de 2024

Engenharia de Telecomunicações - IFSC-SJ

Sumário

1. Introdução	3
2. Implementação	3
2.1. Parte 1 - Adicionar Centésimo de Segundo ao Relógio	3
2.2. Parte 2 - Adicionar PLL	3
2.3. Parte 3 - Modificar contadores para BCD	5
2.4. Parte 4 - Modificar o r_reg para LFSR	5
3. Implementação com somador binário e conversor BCD	6
4. Conclusão	6
5. Códigos VHDL utilizados	7
5.1. bin2bcd	7

1. Introdução

Neste relatório, será apresentado o desenvolvimento de um relógio digital com precisão de milissegundos, utilizando um PLL (Phase-Locked Loop) para a geração de um sinal de clock de 5 kHz. O projeto foi desenvolvido utilizando a ferramenta Quartus Prime Lite Edition 20.1.0.720 e a placa de desenvolvimento DE2-115.

2. Implementação

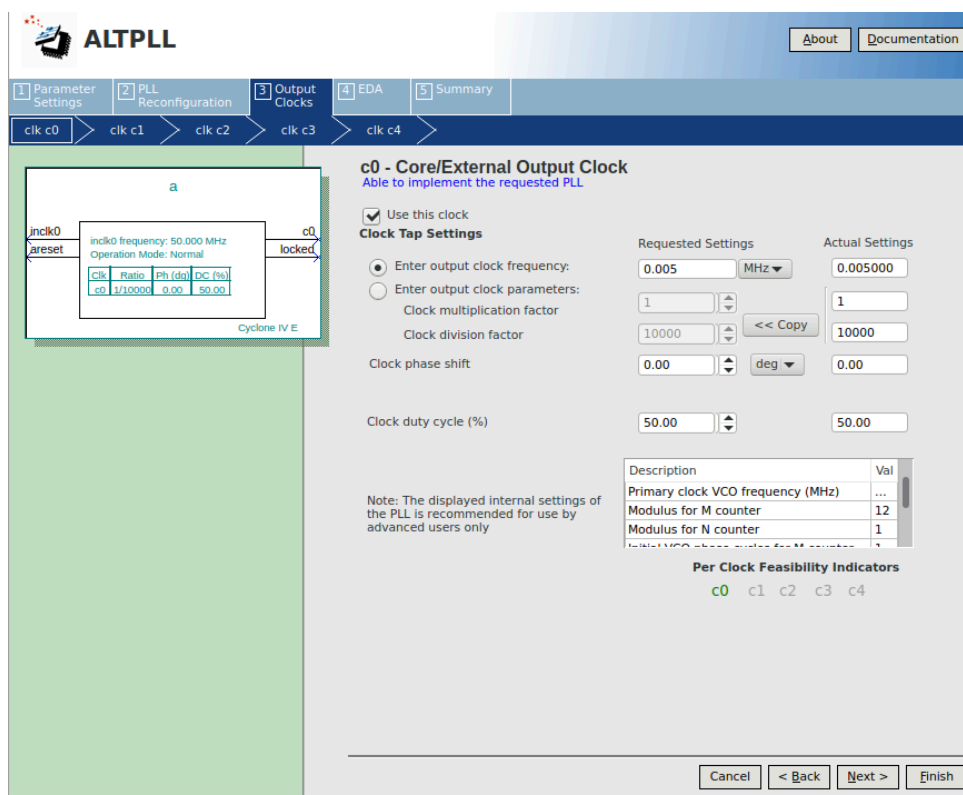
2.1. Parte 1 - Adicionar Centésimo de Segundo ao Relógio

2.2. Parte 2 - Adicionar PLL

A segunda etapa da implementação é a geração de um sinal de clock de 5 kHz (ao invés do sinal de clock padrão utilizado pela FPGA. Para isso, foi utilizado um PLL com um clock de entrada de 50 MHz (valor de clock padrão para o chip implantado nesta placa).

O componente de PLL foi gerado através da ferramenta PLL Intel FPGA IP. Após a configuração do PLL, o sinal de clock de 5 kHz foi obtido na saída deste componente, sendo na sua configuração um **divisor de frequência de 10.000**.

Figure 1: Elaborada pelo Autor



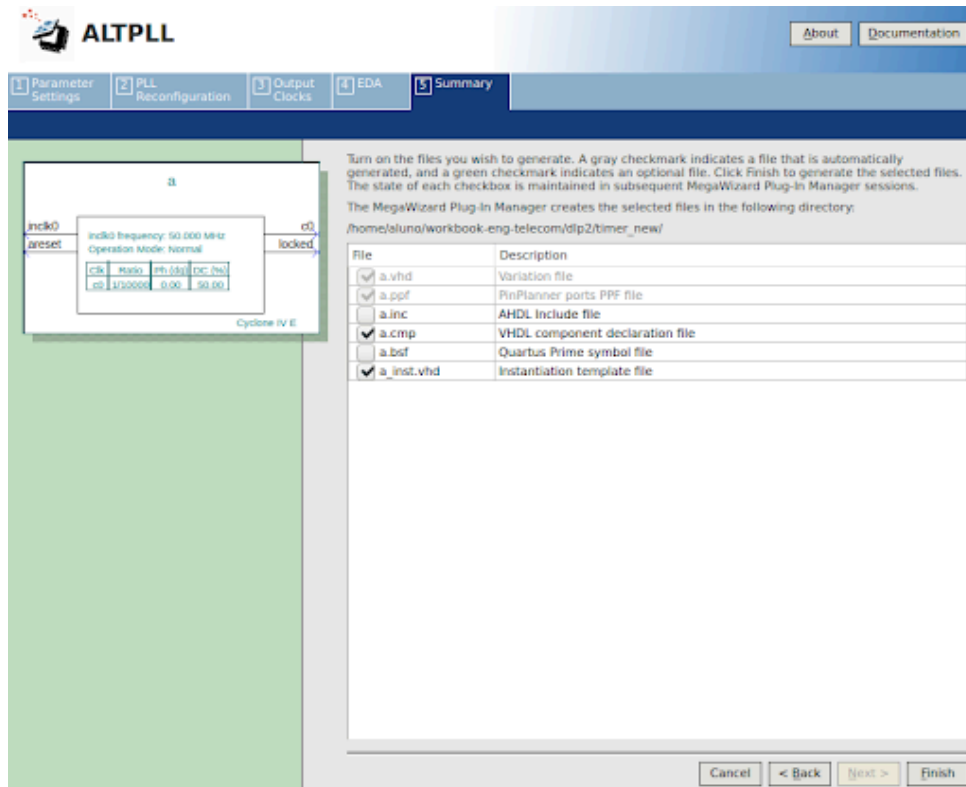
Configuração do PLL através da ferramenta ALT-PLL

Na própria ferramenta, ao inserir os valores de entrada e saída desejados para o circuito de PLL, o Quartus gera o código VHDL necessário para a configuração do circuito que irá con-

trolar a seção analógica do PLL, assim sendo possível realizar a multiplicação ou divisão de frequência corretamente.

Ao finalizar a configuração, foi solicitado gerar os seguintes arquivos:

Figure 2: Elaborada pelo Autor



Configuração do PLL através da ferramenta ALT-PLL

Abaixo está uma sessão do código VHD gerado pelo Quartus, para a configuração VHDL do PLL, demais arquivos são necessários para realizar a instânciação do PLL como um componente do circuito principal.

```

1  GENERIC MAP (
2    bandwidth_type => "AUTO",
3    clk0_divide_by => 10000,
4    clk0_duty_cycle => 50,
5    clk0_multiply_by => 1,
6    clk0_phase_shift => "0",
7    compensate_clock => "CLK0",
8    inclk0_input_frequency => 50000,
9    intended_device_family => "Cyclone IV E",
10   lpm_hint => "CBX_MODULE_PREFIX=p1l",
11   lpm_type => "altpll",
12   operation_mode => "NORMAL",
13   pll_type => "AUTO",

```

É possível notar na descrição acima frequência de entrada, a frequência de saída, o fator de divisão e o duty-cycle do circuito de PLL.

Esses parâmetros são necessários para determinar o formato da onda na saída do circuito, sendo que a frequência precisa ser dividida pelas 10.000 vezes para obter a frequência de 5 kHz.

O duty cycle é de 50% para que a onda seja simétrica, abaixo está uma imagem para ilustrar a diferença entre um duty-cycle de 50% entre 25% e 75%:

Figure 3: Elaborada pelo Autor

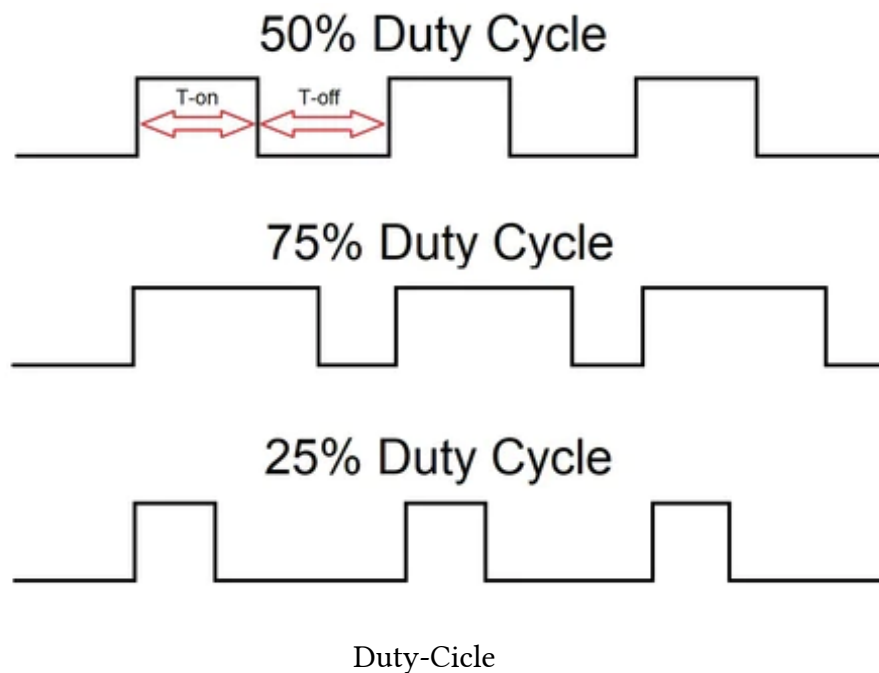
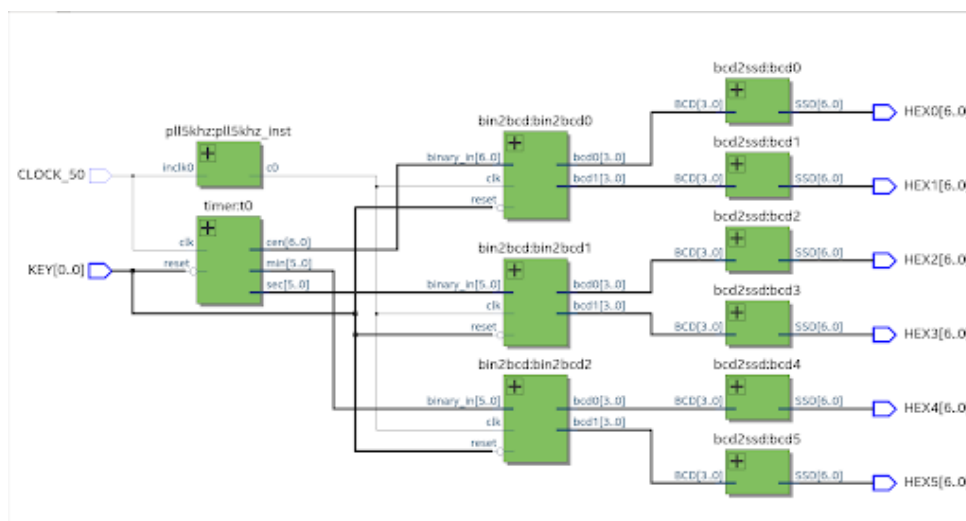


Figure 4: Elaborada pelo Autor



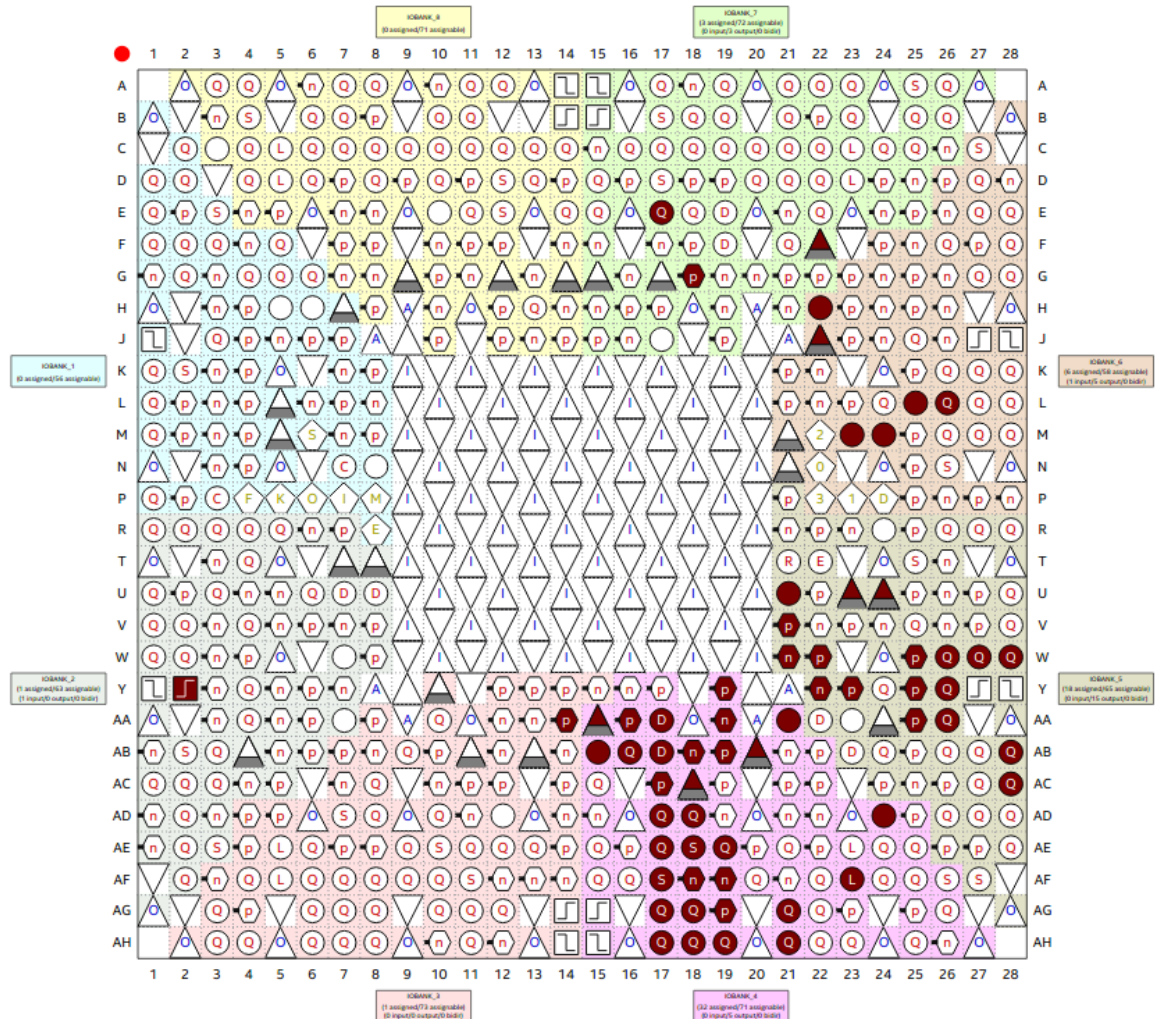
RTL do circuito operando com PLL

2.3. Parte 3 - Modificar contadores para BCD

2.4. Parte 4 - Modificar o r_reg para LFSR

Figure 5: Elaborada pelo Autor

Top View - Wire Bond Cyclone IV E - EP4CE115F29C7



Sinal de entrada no domínio do tempo

3. Implementação com somador binário e conversor BCD

4. Conclusão

A partir da implementação do PLL vista anteriormente, juntamente com a implementação de divisão de clock sem o uso do PLL, podemos concluir que a utilização de um PLL é muito útil para a geração de sinais de clock com frequências específicas de maneira confiável.

Isso pois o PLL é capaz de gerar sinais de clock com frequências específicas, além de possuir uma maior precisão e estabilidade em relação a outros métodos de geração de clock.

Abaixo estão as principais diferenças de tempo de propagação e quantidade de registradores utilizados entre a implementação com e sem o uso do PLL:

Table 1: Elaborada pelo Autor

Implementacao	Área (LE)	Registradores
Parte 1	239	124
Parte 2	83	13.699

Sinal de entrada no domínio do tempo

5. Códigos VHDL utilizados

5.1. bin2bcd

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity bin2bcd is
6      port (
7          A      : in  std_logic_vector (7 downto 0);
8          sd, su, sc : out std_logic_vector (3 downto 0)
9      );
10 end entity;
11
12 architecture ifsc_v1 of bin2bcd is
13     signal A_uns      : unsigned (7 downto 0);
14     signal sd_uns, su_uns, sc_uns : unsigned (7 downto 0);
15
16 begin
17     A_uns <= unsigned(A);
18     sc_uns <= A_uns/100;
19     sd_uns <= A_uns/10;
20     su_uns <= A_uns rem 10;
21     sc <= std_logic_vector(resize(sc_uns, 4));
22     sd <= std_logic_vector(resize(sd_uns, 4));
23     su <= std_logic_vector(resize(su_uns, 4));
24 end architecture;

```

O código binAdder é reponsavel por somar dois números binários de 7 (128 represetações possiveis, e portanto atendendo a especificação) bits e retornar o resultado em binário com 8 bits.