Dispositivos Lógicos Progamáveis 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

Neste relatório, será apresentado o desenvolvimento de um relógio digital com precisão de milisegundos, 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

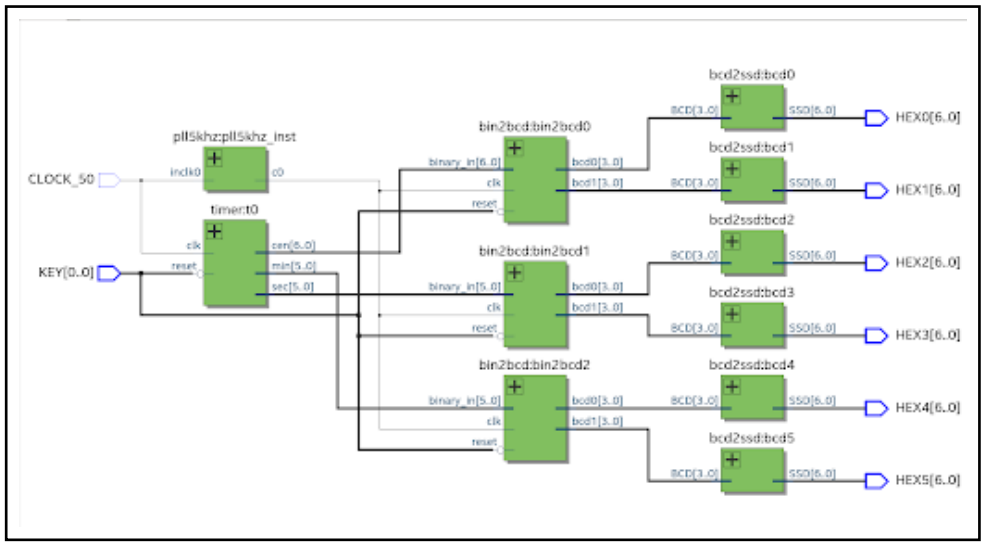
2.1. Parte 1 - Adicionar Centésimo de Segundo ao Relógio

A primeira etapa da implementação foi a adição de um contador de 100 para a contagem de centésimos de segundo. Para isso, foi utilizado um contador de 7 bits, que conta de 0 a 99, e um comparador para resetar o contador quando atingir o valor de 100.

Foi instânciado um novo componente de contagem ao circuito e dois conversores BDC2SSD para a impressão dos digitos em um display de 7 segmentos.

O seguinte RTL foi gerado após a adição do contador de centésimos de segundo ao circuito:

Figure 1: Elaborada pelo Autor



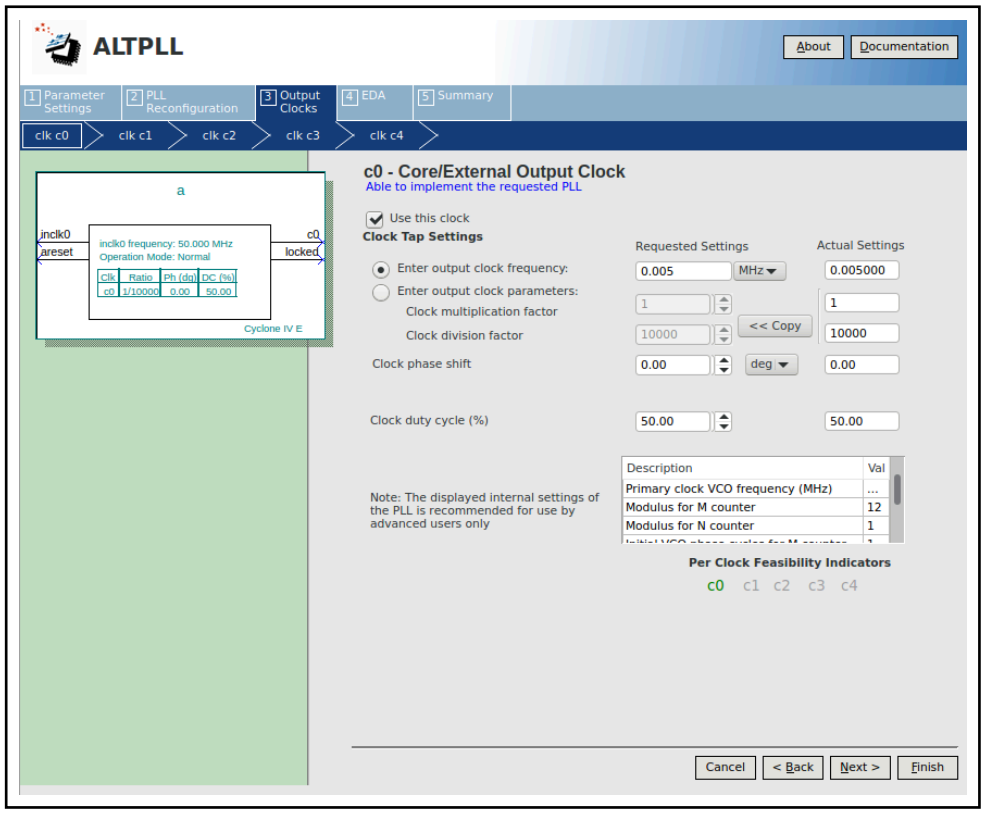
RTL do circuito operando com PLL

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 2: Elaborada pelo Autor

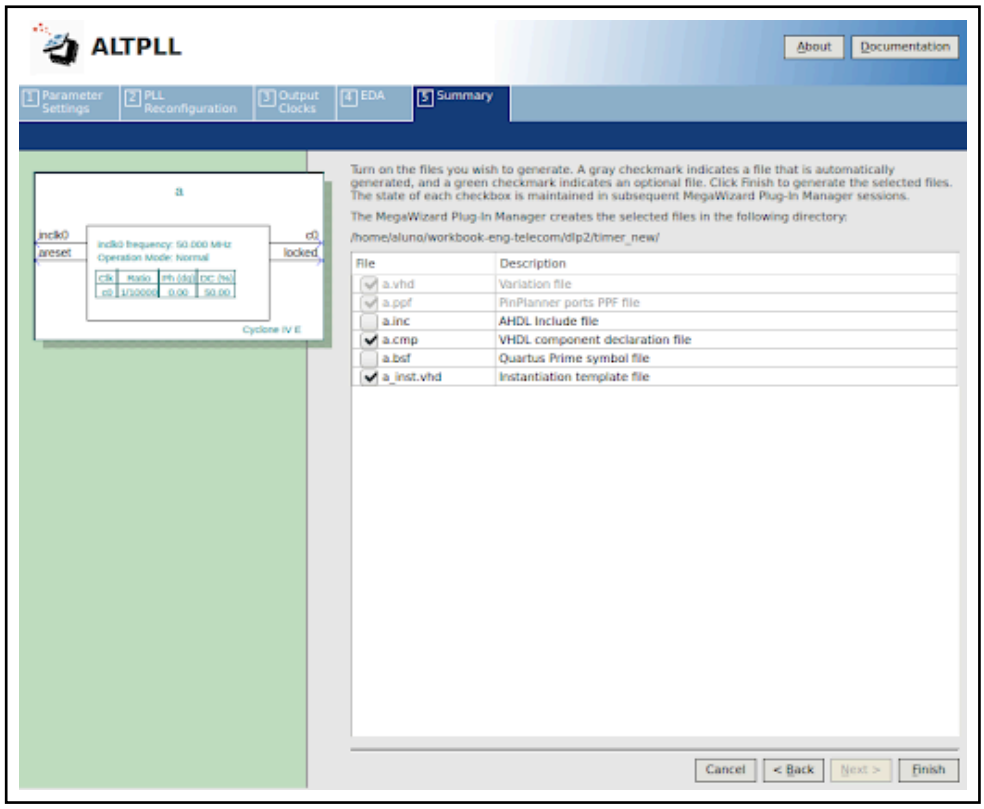


Configuração do PLL através da ferramenta ALT-PLL

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

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

Figure 3: 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.

```
GENERIC MAP (
  bandwidth_type => "AUTO",
  clk0_divide_by => 10000,
  clk0_duty_cycle => 50,
  clk0_multiply_by => 1,
  clk0_phase_shift => "0",
  compensate_clock => "CLK0",
  inclk0_input_frequency => 50000,
  intended_device_family => "Cyclone IV E",
  lpm_hint => "CBX_MODULE_PREFIX=pll",
  lpm_type => "altpll",
  operation_mode => "NORMAL",
  pll_type => "AUTO",
```

É possivel notar na descrição acima frequência de entrada, a frequência de saída, o fator de divisão e o duty-cicle 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 cicle é de 50% para que a onda seja simétrica, abaixo está uma imagem para ilustrar a diferença entre um duty-cicle de 50% entre 25% e 75%:

Figure 4: Elaborada pelo Autor

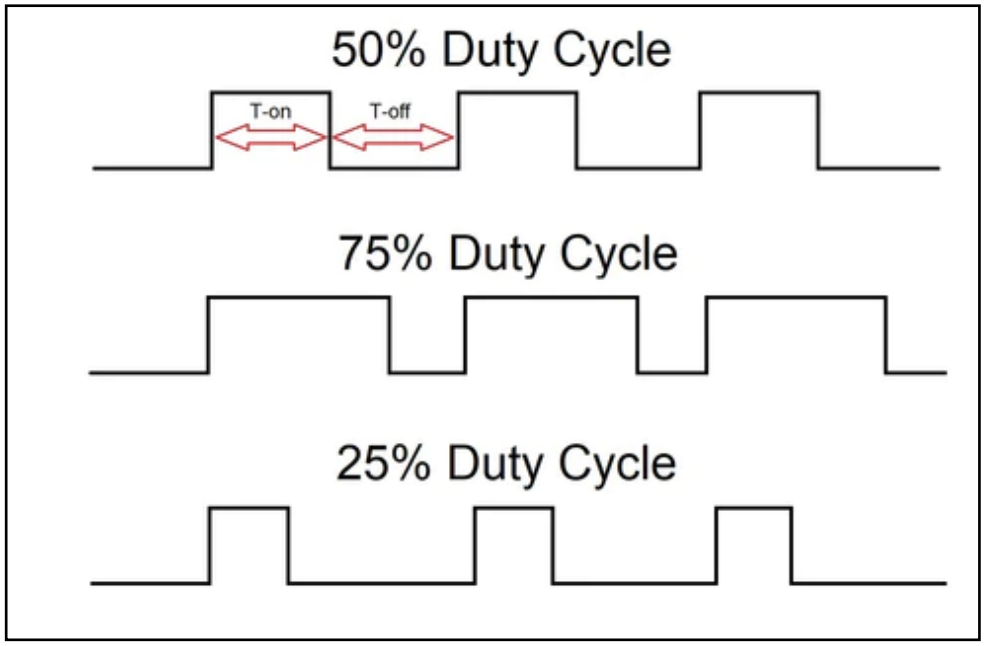
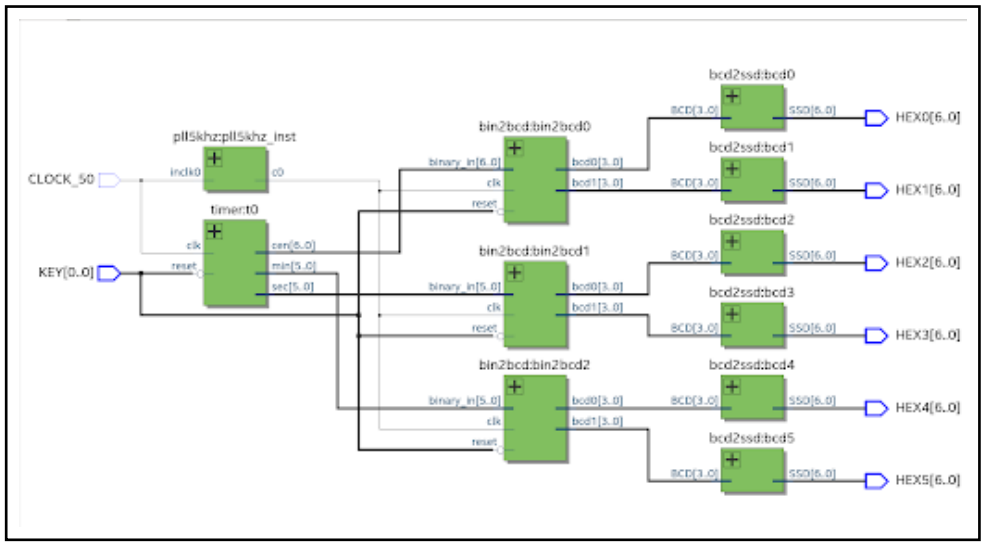


Ilustração de variação de duty-cicle

Com a adição do PLL no circuito, temos uma topologia RTL com um intermediário entre o clock de entrada e o clock de saída, como ilustrado abaixo.

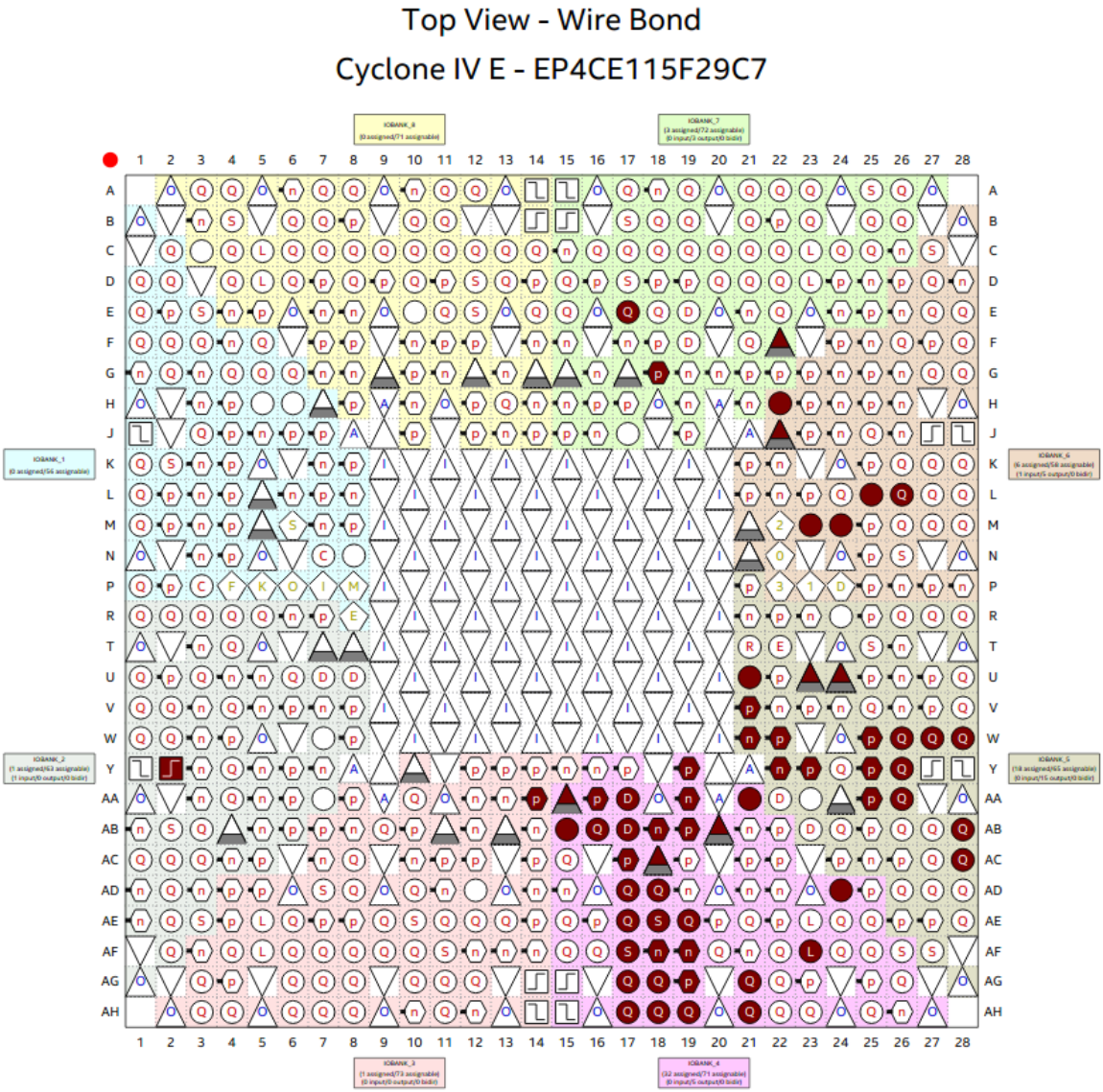
Para ajustar a contagem do segundo, o componente de timer também teve que ser ajustado para contar 5.000 vezes mais rápido, ou seja, de 0 a 99,99 em 5.000 ms.

Figure 5: Elaborada pelo Autor



RTL do circuito operando com PLL

Figure 6: Elaborada pelo Autor



Sinal de entrada no domínio do tempo

2.3. Parte 3 - Modificar contadores para BCD

2.4. Parte 4 - Modificar o r_reg para LFSR

3. 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	83	13.699
Parte 2	239	124

Sinal de entrada no domínio do tempo

4. Códigos VHDL utilizados

4.1. bin2bcd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bin2bcd is
    port (
        A      : in  std_logic_vector (7 downto 0);
        sd, su, sc : out std_logic_vector (3 downto 0)
    );
end entity;

architecture ifsc_v1 of bin2bcd is
    signal A_uns          : unsigned (7 downto 0);
    signal sd_uns, su_uns, sc_uns : unsigned (7 downto 0);

begin
    A_uns  <= unsigned(A);
   sc_uns <= A_uns/100;
    sd_uns <= A_uns/10;
    su_uns <= A_uns rem 10;
    sc     <= std_logic_vector(resize(sc_uns, 4));
    sd     <= std_logic_vector(resize(sd_uns, 4));
    su     <= std_logic_vector(resize(su_uns, 4));
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.