INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS JOÃO PESSOA
CURSO SUPERIOR EM ENGENHARIA ELÉTRICA
SISTEMAS DIGITAIS

EDMILA DE MACÊDO GOMES
GUSTAVO RODRIGUES OLIVEIRA GOLZIO

SEMÁFORO UTILIZANDO FPGA

JOÃO PESSOA – PB
2019

EDMILA DE MACÊDO GOMES
GUSTAVO RODRIGUES OLIVEIRA GOLZIO

SEMÁFORO UTILIZANDO FPGA

Trabalho apresentado à disciplina de Sistemas Digitais, no Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial para obtenção de nota.

Professor: Dr. Lincoln Machado

João Pessoa - PB

2019

LISTA DE FIGURAS

SUMÁRIO

1 - INTRODUÇÃO

Com o passar do tempo o fluxo de automóveis nas ruas tem crescido drasticamente, por causa disso, vê-se o grande aumento de acidentes no trânsito. Como no tráfego existe muitos motivos de incidentes, a sua principal causa seria a ultrapassagem indevida nos cruzamentos. Como uma possível minimização para esse problema foi pensado em um semáforo com contagem regressiva, pois assim evitaria a ultrapassagem nos cruzamentos e ofereceria mais segurança para os motoristas que atravessam o cruzamento, assim como auxilia os pedestres na travessia, pois desse modo eles podem ter noção da hora que o sinal irá abrir ou fechar.

Assim sendo, é imprescindível que o discente como futuro engenheiro possua uma vasta experiência com solução de problemas durante e depois da sua graduação, além disso, é necessário que ele saiba realizar projetos que tragam o bem-estar aos que estão a sua volta e tenha conhecimento sobre o funcionamento das ferramentas que auxiliam no cotidiano, como a família FPGA para projetos futuros.

Para a realização do projeto irão ser necessários equipamentos como: FPGA Cyclone II; Altera Quartus II; resistores; fios para conexão e Leds para simular o semáforo.

2 - OBJETIVOS

Este projeto possui o objetivo de aplicar a teoria de Sistemas Digitais utilizando a placa Altera Cyclone II 2C20, além de seus componentes. Logo, o propósito é a implementação de um semáforo digital fazendo o uso da memória flash, o GPIO e a placa FPGA.

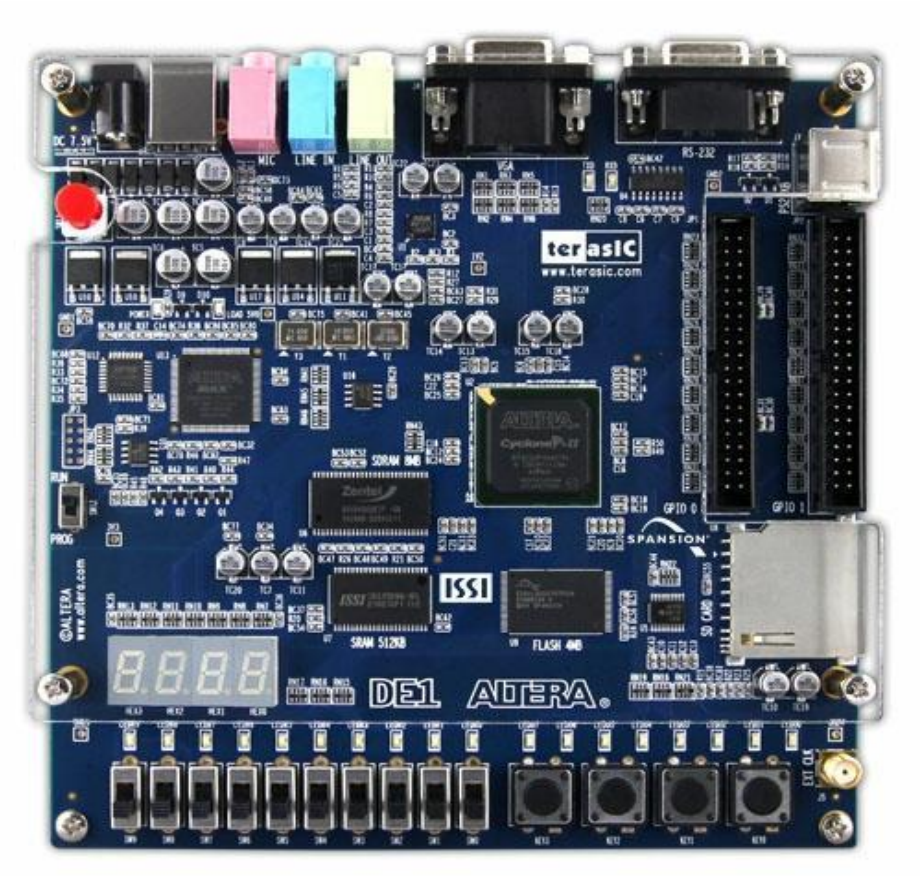
3 - FUNDAMENTAÇÃO TEÓRICA

3.1 FPGA Altera Cyclone II 2C20

O kit de desenvolvimento do FPGA Altera DE1 da família Cyclone II, é um dispositivo lógico programável que auxilia a implementação de circuitos digitais. Além disso, o FPGA é portador de várias entradas e saídas lógicas, por isso, ele consegue diminuir o uso de CIs nos circuitos, atualmente a arquitetura desse kit contém blocos de memória, switches, DSP e processador ARM. Portanto, são diversas as funções lógicas, que vão desde circuitos mais simples como somadores, subtratores, acumuladores - blocos de Entradas e Saídas-, até os mais complexos como, memória, DSP, que podem se conectar de formas diferentes para representar um hardware descrito em uma linguagem de descrição de hardware (HDL) ou em um diagrama de blocos.

A utilização do FPGA é muito destacada em diversos setores da indústria, tal equipamento é bastante implementado em setores nos quais o desempenho, operações em paralelo e em tempo real são cruciais. Sendo um hardware, é possível realizar a operação por ciclo de clock e consequentemente, várias operações podem ser executadas em paralelo e apresentar o resultado no mesmo pulso de clock. Algo que para um software é bastante complexo.

Figura 1 - FPGA Altera Cyclone II 2C20



Fonte: Intel, 2019.

3.2 VHDL

O VHDL - sigla para "VHSIC Hardware Description Language", no qual VHSIC significa "Very High Speed Integrated Circuits" - é a linguagem de descrição de hardware para a programação do FPGA. O ambiente de desenvolvimento integrado utilizado é o Altera Quartus II.

Nesta linguagem, existem 4 estruturas principais:

- Os comentários, os quais não são compilados no código e servem para explicar o funcionamento do código ou seu cabeçalho.
- A *entitity*, onde são descritas as entradas e saídas do projeto.
- A *architecture*, a qual é o corpo do sistema, onde é localizado suas declarações e diretivas de operações e comparações.

- O *process*, no qual há uma sequência de operações a serem realizadas, de maneira estruturada dentro da *architecture*.

4 - MATERIAL UTILIZADO

4.1 Flash Memory 4MB

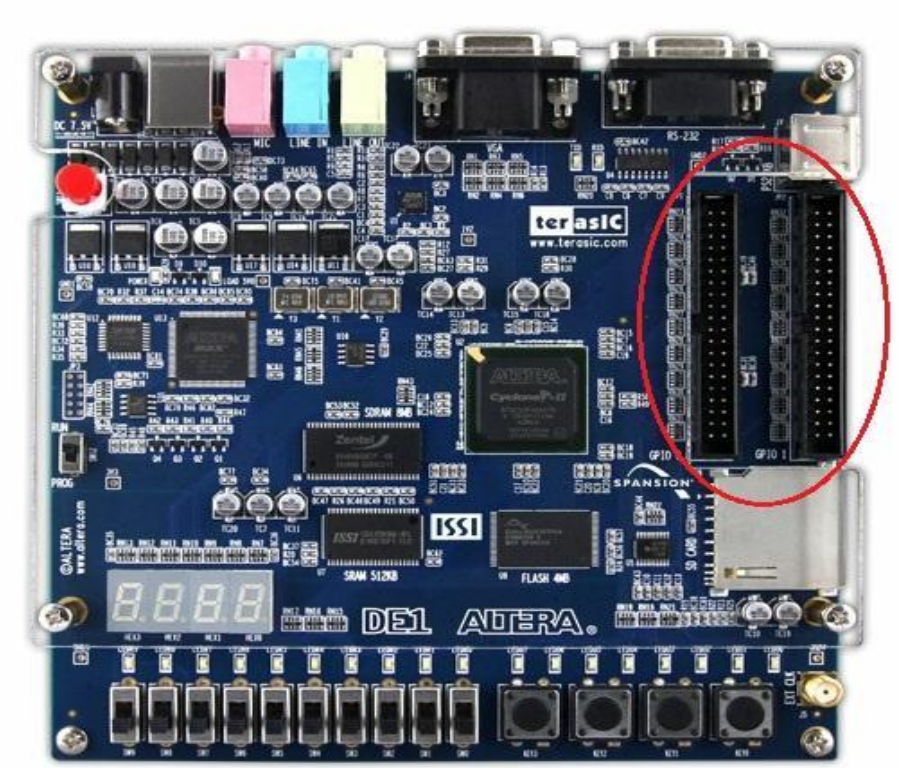
Figura 2 - Flash Memory 4MB

Fonte: Wikipedia, 2019.

A memória flash é um chip de armazenamento não-volátil que pode ser apagado e reprogramado eletricamente. Trata-se de um chip de memória de computador que mantém informações armazenadas sem a necessidade de uma fonte de energia. Esse tipo de memória tem a capacidade de apagar a os dados de um bloco inteiro, o que o torna mais rápido que outros tipos de memória. Esse componente possui 4MB (Mega bites) de armazenamento. Usado para gravar o tempo do clock com o intuito de ligar e/ou desligar os leds.

4.2 GPIO (General Purpose Input/Output)

Figura 3 - GPIO



Fonte: Intel, 2019.

São portas programáveis de entrada e saída de dados que são utilizadas para prover uma interface entre os periféricos e os microcontroladores/microprocessadores. Foram utilizadas para ligar e/ou desligar os Leds.

4.3 Resistores de 10 ohms

Figura 4 - Resistor de 10 ohms

Fonte: Baú da Eletrônica, 2019.

No circuito montado, foi necessário utilizar resistores 2 de 10 ohms para assegurar de que os leds não esquentem e queimem.

4.4 Leds 5 mm

Figura 5 - Led 5 mm

Fonte: Baú da Eletrônica, 2019.

No circuito montado, foi necessário utilizar 2 leds, um vemelho e outro verde, para a indicação do semáforo.

5 - DESENVOLVIMENTO DO PROJETO

O FPGA foi programado com a linguagem de descrição de hardware VHDL. O código está nos apêndices A e B deste relatório. Os leds são conectados às portas GPIO 0 da placa que utilizamos.

Foi criado, na *entity*, duas saídas que estão conectadas as digitais do GPIO 0 que irão indicar o sinal verde e vermelho. Também foi declarado um reset para reiniciar a contagem. Além disso, foram declaradas as entradas e saídas para ter acesso a memória flash.

Na *architecture*, foi obtido um componente chamado divisor clock que tem por finalidade dividir a frequência do clock de modo que ele pulse a cada segundo. Para ter acesso ao arquivo contido na memoria flash é necessário informar o endereço, assim como

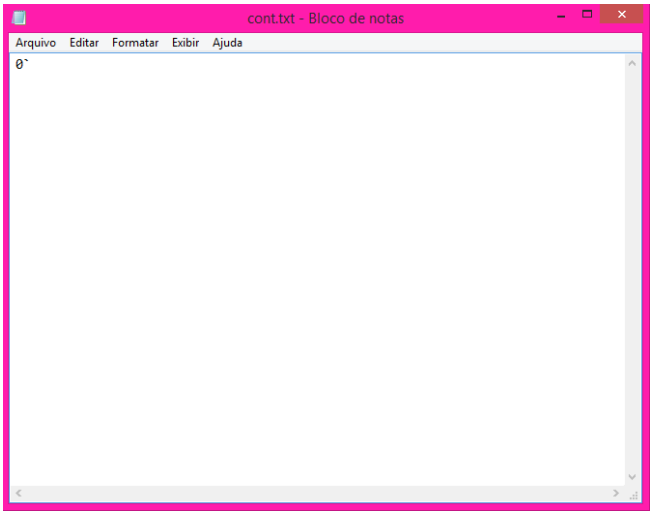
armazená-lo em um local. Caso o valor na memória seja igual ao valor setado, então um led indicador irá acender na placa e também será possível setar um valor utilizado para a contagem do tempo em que o semáforo irá mudar de posição. Além disso, é inicializado um *process* que irá incrementar um valor temporário a cada borda do clock.

Por fim, a saída resultante será setada em nível lógico alto ou baixo, dependendo do estado de duas variáveis.

Figura 6 – Local para armazenar o arquivo na memória

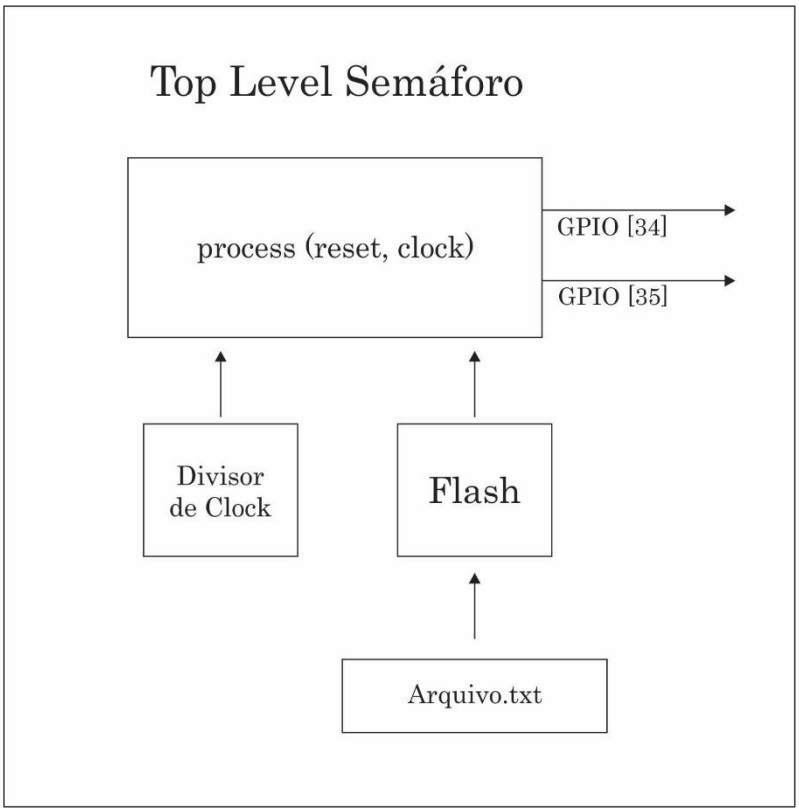
Fonte: Elaborado pelos autores, 2019

Figura 7 – Arquivo em .txt enviado



Fonte: Elaborado pelos autores, 2019

Figura 8 - Diagrama de blocos referente ao código VHDL

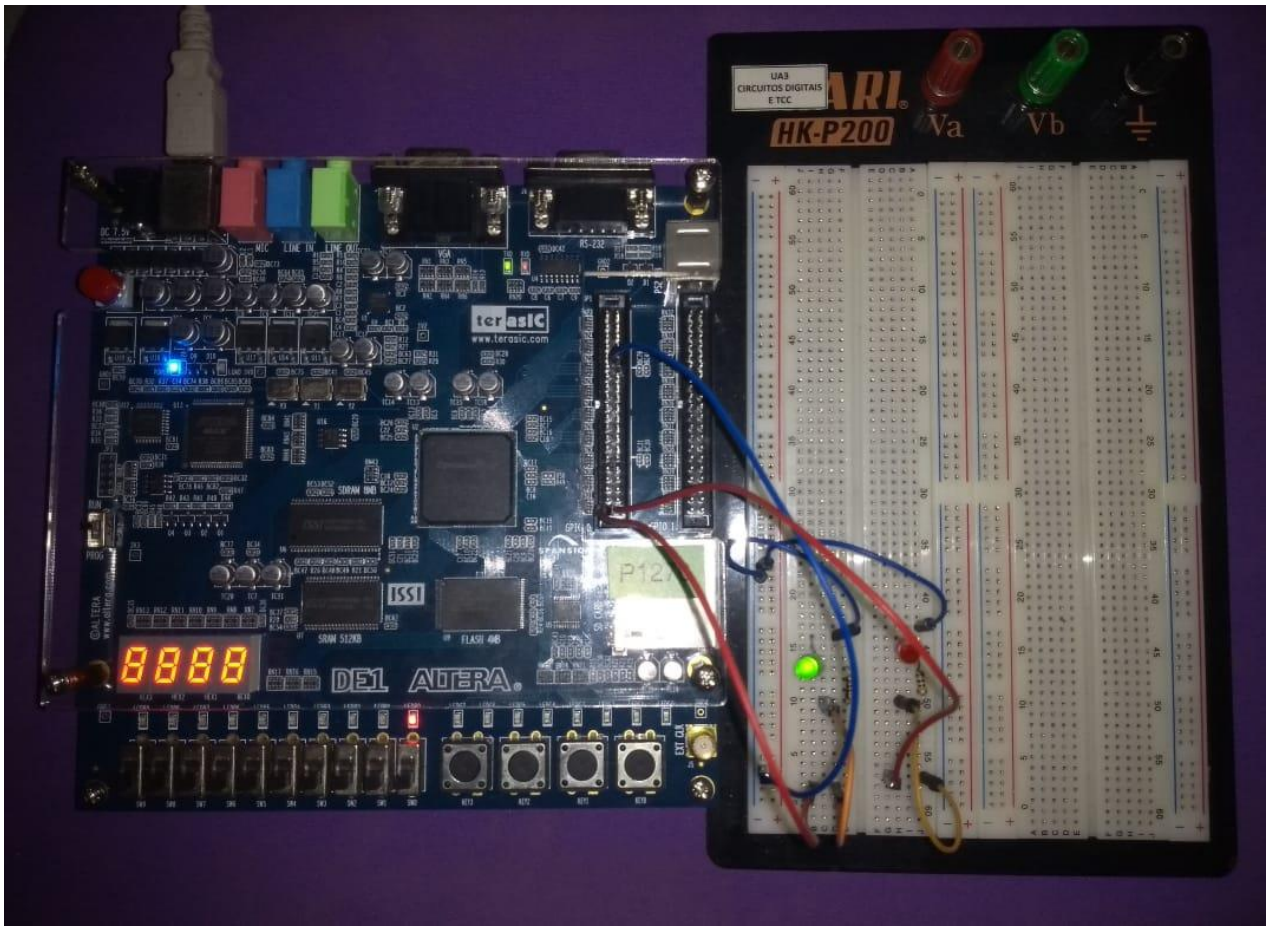


Fonte: Elaborado pelos autores, 2019.

6 - RESULTADOS

Foram obtidos resultados satisfatórios para semáforo, tendo em vista que ele atendeu às expectativas de comparar um valor com o que estava na memória e acionar o contador para contar o tempo de acender os leds. Assim como também foi possível observar que ao alterar o endereço da memória, o valor que passou a ser outro, não habilita a contagem do tempo, atendendo ao objetivo de utilizar o arquivo contido na memória para desbloquear a contagem.

Figura 9 – Circuito equivalente



7 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] EESC - USP. **Memória Flash**. Disponível em <https://edisciplinas.usp.br/mod/resource/view.php?id=227498> Acesso em 24 de julho, 2019.

[2] Altera Corporation.. **DE1 Development and Education Board - User Manual**. <https://www.intel.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-4904342209-de1-usermanual.pdf>. Acesso em 25 de julho, 2019.

[3] **CYCLONE II FPGAS. INTEL.** Disponível em: <https://www.intel.com/content/www/us/en/programmable/products/fpga/cyclone- series/cyclone-ii/support.html>. Acesso em 25 de julho, 2019.

APÊNDICES

APÊNDICE A – CÓDIGO TOP LEVEL EM VHDL DO SEMÁFORO

```
-- Semaforo Digtal Utilizando FPGA (Top Level)
--Autores: Edmila de Macedo e Gustavo Golzio.

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity semaforo  is
        port(
        clock : in std_logic;
        reset : in std_logic;
        verde: out std_logic;   -- GPIO 34
        vermelho:out std_logic;         -- GPIO 35

                --------- MEMORIA FLASH
        led: out std_logic;
        key : in std_logic:='0';
        flash_adress:out std_logic_vector(21 downto 0);
        f_reset:out std_logic;
        f_out_ena: out std_logic;
        f_write_ena:out std_logic;
```

```
        f_data: in std_logic_vector(7 downto 0)
        );

end semaforo ;

architecture arq of semaforo  is

        signal temp:integer range 0 to 60;
        signal clk_1hz:std_logic;
        signal ver:std_logic_vector(7 downto 0);
        signal teste:std_logic_vector(7 downto 0):="00110000";--"01100000"; -- 30 e 60 que
esta na memoria
        signal cont:std_logic;--_vector(7 downto 0);
                -- signal para o sinal convertido receber 30 do de que tem na memoriaa
        signal conver:integer range 0 to 30;
        signal conver1:integer range 0 to 31;
        signal conver2:integer range 0 to 32;
        signal conver3:integer range 0 to 33;
        signal conver4:integer range 0 to 34;

        component divisor_clk is
        port (clk_in: in std_logic;
        q: out std_logic);
        end component;

        begin
        ----- FLASH --------
        flash_adress<= "00000000000000000000";          -- endereço do que esta na
memoria
        ver(7 downto 0)<= f_data when key='0' else (others => '0'); -- ver = dado que esta na
memoria
        led<= '1' when ver = teste else '0';      -- acender o led caso o que esteja na memoria
seja o mesmo que esta em teste
        f_reset<='1';
        f_out_ena<=key;
        f_write_ena<='1';
        -- 30 em bin e 30 em hexa                 --o que tem em ver ta em hexa ai passa pra
binario pra poder comparar

        conver <= 30 when ver = "00110000" else 0;  --"01100000"
        conver1 <= 31 when ver = "00110000" else 0; --"01100000"
        conver2 <= 32 when ver = "00110000" else 0; --"01100000"
        conver3 <= 33 when ver = "00110000" else 0; --"01100000"
        conver4 <= 34 when ver = "00110000" else 0; --"01100000"
```

```
        -- quando o que tem na memoria for igual àquele valor, entao vai ter o range

        -- Declaracao do objeto
        divisor: divisor_clk port map(clk_in =>clock, q =>clk_1hz);

        process (reset, clock) --sensibilidade
        begin
      if reset = '1' then
        temp <= 0;
   elsif rising_edge(clk_1hz) then --elsif clock'event and clock ='1' then tambem serve
        temp <= temp + 1;
      end if;
  end process;

         verde <= '0' when temp = conver or temp = conver1 or temp = conver2 or temp =
conver3 or temp = conver4 else '1';
         vermelho <= '1' when temp = conver or temp = conver1 or temp = conver2 or temp
= conver3 or temp = conver3 else '0';

end arq;
```

APÊNDICE B - CÓDIGO DIVISOR DE CLOCK

```
-- DIVISOR DE CLOCK para dar o tempo correto
library IEEE;
-- Semaforo Digtal Utilizando FPGA (Divisor de Clock)
--Autores: Edmila de Macedo e Gustavo Golzio.
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity divisor_clk is
port (clk_in: in std_logic;
      q: out std_logic);
end divisor_clk;

architecture behavioral of divisor_clk is
signal clk_div: std_logic;
begin

process(clk_in)
variable count: integer:= 0;
```

```
begin
    if(clk_in'event and clk_in='1') then
        count:= count+1;
        if(count = 25000000) then
            clk_div <= not clk_div;
            count:= 1;
        end if;
    end if;
end process;

q <= clk_div;

end behavioral;
```