PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
CENTRO DAS CIÊNCIAS EXATAS E TECNOLOGIA
CURSO: CIÊNCIA DA COMPUTAÇÃO
DISCIPLINA: DESCRIÇÃO DE SISTEMAS DIGITAIS
PROFESSORA: EDITH RANZINI

CONTADOR DE UNS

ARTHUR CALDEIRA BRANT
SILMARA ALVARES BARBOSA

SÃO PAULO
2019

INTRODUÇÃO

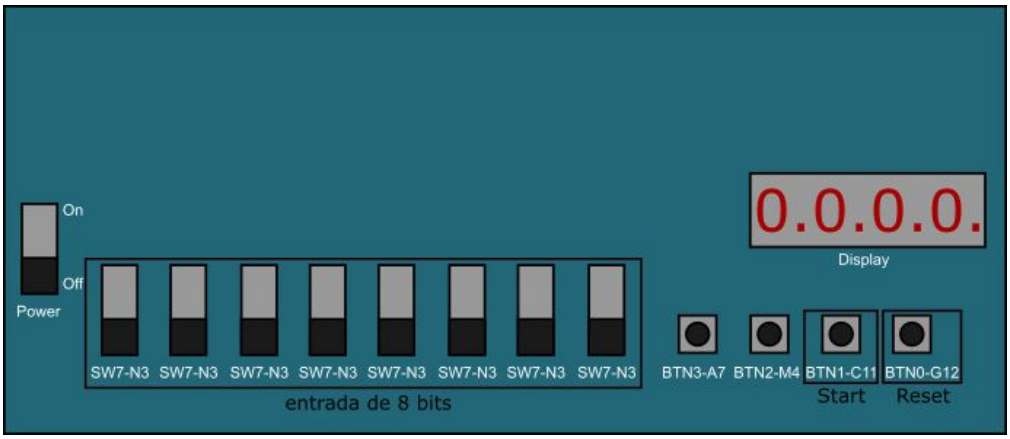
Nas seções seguintes será apresentado as etapas do desenvolvimento do contador de uns. As etapas consistem em discutir, descrever, definir, implementar e apresentar uma conclusão dos resultados positivos e negativos obtidos no processo de desenvolvimento.

DESCRIÇÃO DO PROJETO

O projeto consiste na contagem de uns em uma entrada de 8 bits manipulada pelo usuário, e ao final apresenta o resultado no *display*. A entrada de 8 bits é definida pelas 8 chaves da placa, onde o usuário pode alterar para os possíveis valores 0 (*low*) e 1 (*high*).

Depois de manipular as chaves da forma desejada, basta acionar o botão de *start* para a entrada ser carregar e inicializar a contagem. Quando a contagem é finalizada o resultado é enviado para o *display*.

O botão de *reset* deve ser acionado toda vez que o usuário deseja carregar uma nova entrada.

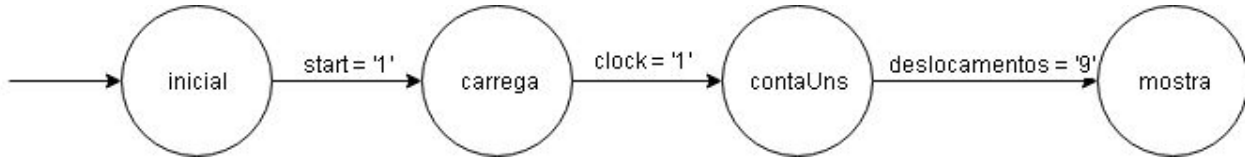


[Figura 1] FPGA Basys 2 - esboço das chaves e botões da placa.

ELABORAÇÃO DO PROJETO

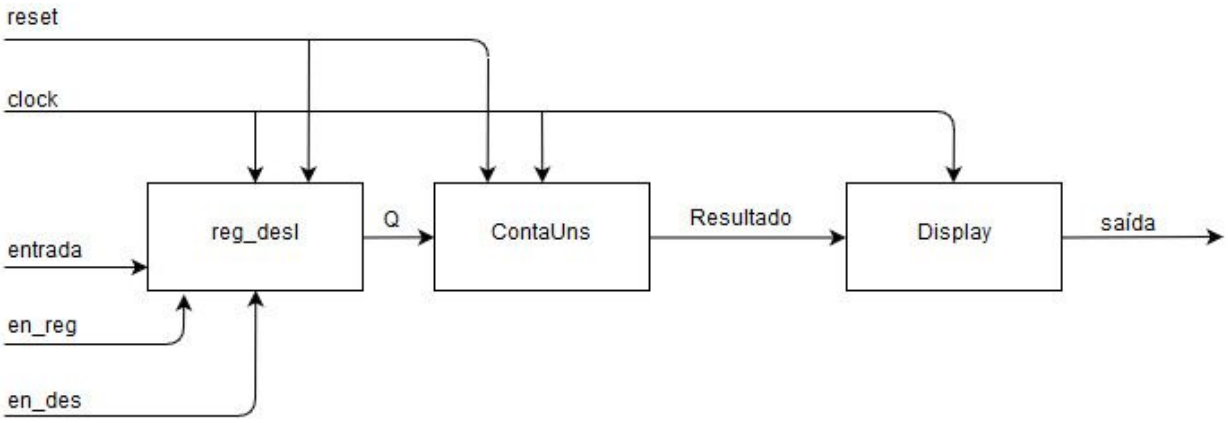
O diagrama de máquina de estados definida para o contador de uns possui 4 estados, sendo eles: inicial, carrega, contaUns e mostra. O DME Controla a interação do usuário com os botões *reset* e *start* emitindo sinais para o fluxo de dados.

O estado contaUns possui um contador de deslocamentos para que o registrador desloque os bits apenas 8 vezes.



[Figura 2] DME - Diagrama de Máquina de Estados do contador de uns.

O fluxo de dados possui os seguintes componentes: registrador de deslocamento, contador de uns e o display de 7 segmentos.



[Figura 3] FD - Diagrama do Fluxo de Dados do contador de uns.

UNIDADE DE CONTROLE

A unidade de controle elaborada no projeto possui 4 estados. Segue abaixo cada um deles com suas respectivas funcionalidades.

- **inicial:** Neste estado, o usuário pode interagir com as 8 chaves da placa para montar a entrada desejada, podendo colocar cada uma delas em 1 (*high*) ou 0 (*low*). Ao pressionar *start* ocorre uma transição de estado para o carrega.
- **carrega:** O sistema chega neste estado quando o *start* = 1. Neste estado, a entrada de 8 bits é armazenada no registrador de deslocamento. Realizado o carregamento, na próxima subida de borda do clock ocorre uma transição de estado para o contaUns.
- **contaUns:** Após o carregamento, nas próximas bordas de subida do clock será feita uma verificação do bit deslocado pelo registrador, checando se é 1 ou 0. Se o bit deslocado for igual a 1 é contabilizado no contador. Foi decidido que o registrador só desloca 8 vezes, depois de 8 iterações o registrador terá um dado de 8 bits zerados.

  Quando chega na nona iteração ocorre uma transição de estado para mostra.
- **mostra:** Neste estado, é apresentado no *display* o resultado final da contagem de uns. Pressionando o botão *reset* o usuário volta ao estado inicial.

A unidade de controle sinaliza para o fluxo de dados quando pode ocorrer o armazenamento e deslocamento.

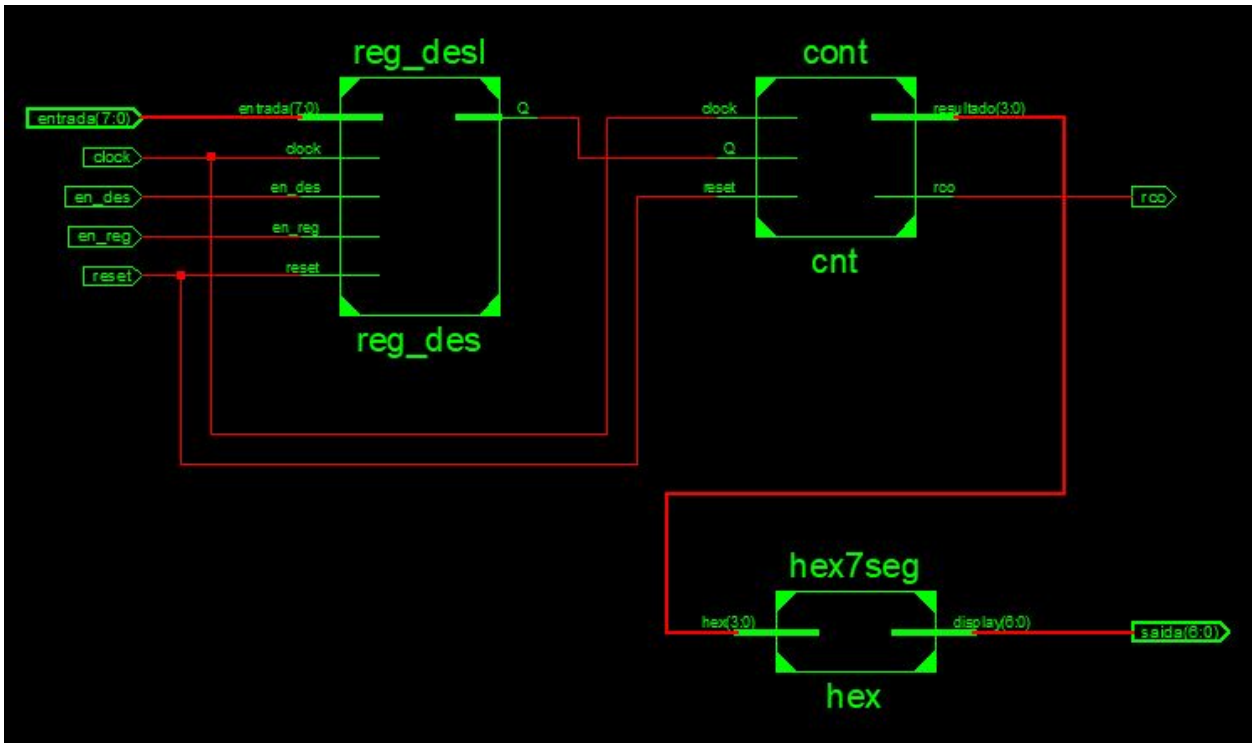
FLUXO DE DADOS

O Fluxo de Dados deste trabalho possui 3 componentes: um registrador de deslocamento (reg_desl), um contador (cont) e o display de sete segmentos (hex7seg). E ele recebe 5 entradas: uma cadeia de 8 bits enviado pelo usuário; o *clock*; o enable do registrador de deslocamento; o *enable* de deslocamento, que é o sinal que diz para o registrador que ele pode deslocar os bits armazenados dentro dele; e um sinal de *reset*.

O registrador de deslocamento tem dois objetivos: armazenar os bits recebido pela entrada do usuário; e deslocar todos os bits armazenados.

O contador é responsável por contar a quantidade de bits 1 presente na cadeia de bits enviado pelo usuário.

O display de sete segmentos imprime a quantidade de bits 1 que foi contabilizada da entrada do usuário.



[Figura 4] RTL do fluxo de dados

SISTEMA DIGITAL

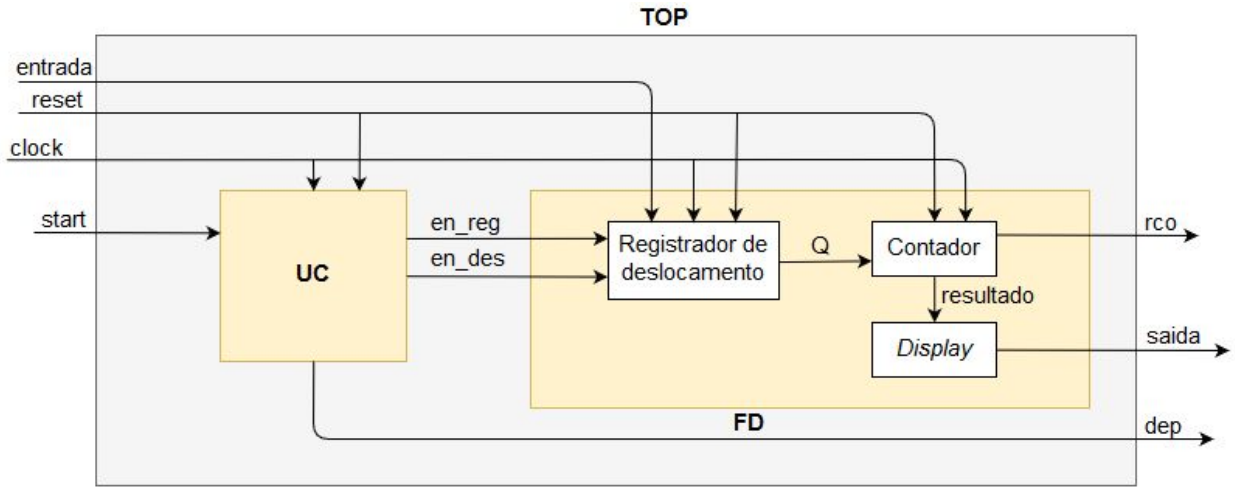
A Unidade de Controle e o Fluxo de dados se interagem através de um componente que está acima dos dois, o *top*. Este componente é responsável por receber as entradas diretas do usuário, realizar os *port maps* dos dois componentes abaixo dele e enviar os dados da UC para o FD.

O Top recebe como entrada uma cadeia de *bits* do usuário, um sinal de *reset*, *start* e o *clock*.

A UC recebe o sinal de *start*, o que desencadeia a mudança do estado inicial para o estado carrega. No estado carrega, a UC envia para o FD um sinal de *enable* para o registrador armazenar a cadeia de *bits* enviada pelo usuário.

Na próxima borda de *clock,* a UC sai do estado carrega e vai para o estado contaUns. No estado contaUns, a UC envia para o FD um sinal de *enable* para deslocar um *bit* no registrador de deslocamento. A UC envia esse sinal de deslocamento por oito vezes. Enquanto isso, o contador recebeu cada *bit* que foi deslocado, verificou se o valor era '1' e somou.

Após todos os bits terem sido deslocados, o contador envia o resultado da soma para o *display* de sete segmentos.



[Figura 5] Top - ligação do UC com FD.

TABELA DE PINAGEM

Abaixo segue a tabela de pinagem utilizada no desenvolvimento:

Sinal	Interface	Pino
clock	clock da placa	B8
reset	botão BTN0	G12
start	botão BTN1	C11
entrada(0)	SW0	P11

entrada(1)	SW1	L3
entrada(2)	SW2	K3
entrada(3)	SW3	B4
entrada(4)	SW4	G3
entrada(5)	SW5	F3
entrada(6)	SW6	E2
entrada(7)	SW7	N3
saida(0)	Hex(0)	L14
saida(1)	Hex(1)	H12
saida(2)	Hex(2)	N14
saida(3)	Hex(3)	N11
saida(4)	Hex(4)	P12
saida(5)	Hex(5)	L13
saida(6)	Hex(6)	M12

RESULTADOS

O trabalho Contador de Uns funcionou como o esperado. A interação do usuário é simples e o display mostra a quantidade de bits 1 inserido como desejado.

Parte do código também foi simplificado. Por exemplo, foi decidido que o controle da quantidade de deslocamentos feita no registrador de deslocamentos - dentro do FD - seria feito na UC utilizando uma variável simples. Como foi explicado pelo professor, não é a solução ideal, mas é uma saída.

Também decidiu-se pela criação de um componente especializado em receber o bit deslocado do registrador de deslocamento e que contasse sempre que tal bit fosse ‘1’.

CONCLUSÃO

Em geral, boa parte do trabalho de Contador de Uns reaproveitou componentes que já tinham sido criados em trabalhos anteriores.

Apesar dos reaproveitamentos, houve uma dificuldade inicial para entender e implementar o registrador de deslocamento. Como deslocar os bits dentro do vetor? Como obter o bit deslocado? Após pesquisas e ajuda, foi possível solucionar o problema.

O projeto também ajudou a ganhar mais confiança na hora de criar a UC. Inicialmente, ainda havia certa dificuldade na implementação dos estados em VHDL.

APÊNDICE – Implementação do contador de uns

```
-----------------------------------------top.vhd-----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity top is
   port (      clock, btn_reset, btn_start  :in std_logic;
               entrada                      :in std_logic_vector(7 downto 0);
               saida                        :out std_logic_vector(6 downto 0);
               rco                          :out std_logic;
               dep                          :out std_logic_vector(1 downto 0)  );
end top;

architecture hardware_top of top is
   component unidadeControle is
         port ( clock, reset, start         :in std_logic;
               en_reg, en_des               :out std_logic;
               dep                          :out  std_logic_vector(1 downto 0) );
   end component;

   component fluxoDados is
         port ( clock, reset, en_reg, en_des      :in  std_logic;
               entrada                            :in std_logic_vector(7 downto 0);
               saida                              :out std_logic_vector(6 downto 0);
               rco                                :out std_logic  );
   end component;
```

```
   signal en_reg, en_des                        :std_logic;
begin
   uc0: unidadeControle port map(clock, btn_reset, btn_start, en_reg, en_des, dep);
   fd: fluxoDados port map(clock, btn_reset, en_reg, en_des, entrada, saida, rco );
end hardware_top;

-----------------------------------------uc0.vhd-----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity unidadeControle is
   port (       clock, reset, start          :in std_logic;
                en_reg, en_des               :out std_logic;
                dep                          :out  std_logic_vector(1 downto 0) );
end unidadeControle;

architecture hardware_uc of unidadeControle is
   type tipo_estado is (inicial, carrega, contaUns, mostra);
   signal estado   : tipo_estado;
   signal deslocamentos: unsigned (3 downto 0);
begin
   process (estado, clock, reset, start, deslocamentos)
   begin
        if reset = '1' then
                estado <= inicial;
                deslocamentos <= (others => '0');
        else if (clock'event and clock = '1') then
                case estado is
                        when inicial =>
                                if start = '1' then
                                        estado <= carrega;
                                else
                                        estado <= inicial;
                                end if;

                        when carrega =>
                                        estado <= contaUns;

                        when contaUns =>
                                if deslocamentos = 9 then
                                        estado <= mostra;
                                else
                                        deslocamentos <= deslocamentos + 1;
```

```
                                    estado <= contaUns;
                                end if;

                            when mostra =>
                                    estado <= mostra;
                        end case;
                end if;
                end if;
        end process;

        with estado select en_reg <=
                '1' when carrega,
                '0' when others;

        with estado select en_des <=
                '1' when contaUns,
                '0' when others;

        with estado select dep <=
                "00" when inicial,
                "01" when carrega,
                "10" when contaUns,
                "11" when mostra,
                "11" when others;
end hardware_uc;

-----------------------------------------fd0.vhd-----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fluxoDados is
        port(           clock, reset, en_reg, en_des    :in  std_logic;
                        entrada                         :in std_logic_vector(7 downto 0);
                        saida                           :out std_logic_vector(6 downto 0);
                        rco                             :out std_logic );
end fluxoDados;

architecture hardware_fd of fluxoDados is
        component reg_desl is
                port  (clock, reset, en_reg, en_des     :in  STD_LOGIC;
                        entrada                         :in  STD_LOGIC_VECTOR(7 downto 0);
                        Q                               :out  STD_LOGIC );
        end component;

        component cont is
```

```
        port (clock, reset, Q                   :in  std_logic;
              resultado                         :out std_logic_vector(3 downto 0);
              rco                               :out std_logic );
    end component;

    component hex7seg is
        port (hex                               :in  std_logic_vector(3 downto 0);
              display                           :out std_logic_vector(6 downto 0) );
    end component;

    signal Q                                    :std_logic;
    signal resultado                            :std_logic_vector(3 downto 0);

begin
    reg_des: reg_desl port map(clock, reset, en_reg, en_des, entrada, Q);
    cnt: cont port map(clock, reset, Q, resultado, rco);
    hex: hex7seg port map(resultado, saida);
end hardware_fd;

-----------------------------------reg_desl.vhd-----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity reg_desl is
        Port (  clock, reset, en_reg, en_des    :in  STD_LOGIC;
                entrada                         :in  STD_LOGIC_VECTOR(7 downto 0);
                Q                               :out  STD_LOGIC );
end reg_desl;

architecture hardware_reg of reg_desl is
signal IQ: std_logic_vector(7 downto 0);
begin
    process (clock, reset, en_reg, en_des, entrada, IQ)
    begin
        if reset = '1' then
                IQ <= (others => '0');
        elsif clock'event and clock = '1' then
                if en_reg = '1' then
                        IQ <= entrada;
                elsif en_des = '1' then
                        IQ <= '0' & IQ(7 downto 1);
                end if;
        end if;
        Q <= IQ(0);
```

```
    end process;
end hardware_reg;

-----------------------------------cnt.vhd-----------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity cont is
    port (      clock, reset, Q             :in  std_logic;
                resultado                   :out std_logic_vector(3 downto 0);
                 rco                        :out std_logic  );
end cont;

architecture hardware_cont of cont is
signal IQ: unsigned (3 downto 0);
begin
    process (clock, reset, IQ, Q)
    begin
        if reset = '1' then
                IQ <= (others => '0');
        elsif clock'event and clock = '1' then
                if Q = '1' then
                        IQ <= IQ + 1;
                end if;
        end if;
        resultado <= std_logic_vector(IQ);
    end process;
    rco <= '1' when (IQ = 15) else '0';
end hardware_cont;

-----------------------------------hex.vhd-----------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity hex7seg is
    port (      hex                 :in  std_logic_vector(3 downto 0);
                display             :out std_logic_vector(6 downto 0)
          );
end hex7seg;

architecture hardware_hex of hex7seg is
begin
    with hex select display <=
```

```
            "1000000" when "0000",
            "1111001" when "0001",
            "0100100" when "0010",
            "0110000" when "0011",
            "0011001" when "0100",
            "0010010" when "0101",
            "0000010" when "0110",
            "1111000" when "0111",
            "0000000" when "1000",
            "0011000" when "1001",
            "0001000" when "1010",
            "0000011" when "1011",
            "1000110" when "1100",
            "0100001" when "1101",
            "0000110" when "1110",
            "0001110" when others;
end hardware_hex;

----------------------------------pinagem.ucf-------------------------------------------------------
NET "clock" LOC = "B8";
//botoes
NET "btn_start" LOC = "C11";
NET "btn_reset" LOC = "G12";
NET "entrada(0)" LOC = "P11";
NET "entrada(1)" LOC = "L3";
NET "entrada(2)" LOC = "K3";
NET "entrada(3)" LOC = "B4";
NET "entrada(4)" LOC = "G3";
NET "entrada(5)" LOC = "F3";
NET "entrada(6)" LOC = "E2";
NET "entrada(7)" LOC = "N3";
NET "saida(0)" LOC = "L14";
NET "saida(1)" LOC = "H12";
NET "saida(2)" LOC = "N14";
NET "saida(3)" LOC = "N11";
NET "saida(4)" LOC = "P12";
NET "saida(5)" LOC = "L13";
NET "saida(6)" LOC = "M12";
```