[INF01175] Sistemas Digitais para Computadores A

Trabalho 2 - PCPO e HLS

Luccas da Silva Lima e Matheus Almeida Silva

1

A Conjectura de Collatz

Como ela funciona?

A Conjectura de Collatz

Como ela funciona?

A conjectura diz que, para qualquer número natural inteiro inicial, chegaremos sempre ao valor 1 depois de uma quantidade finita de passos.

$$C(x) = \begin{cases} 3x+1 & \text{if } x \equiv 1 \pmod 2 \\ x/2 & \text{if } x \equiv 0 \pmod 2 \end{cases}$$

Os passos seguem da seguinte forma:

- Caso x seja par, deve-se dividir o número por 2.
- Caso x seja ímpar, deve-se multiplicar o número por 3 e somar 1.

2

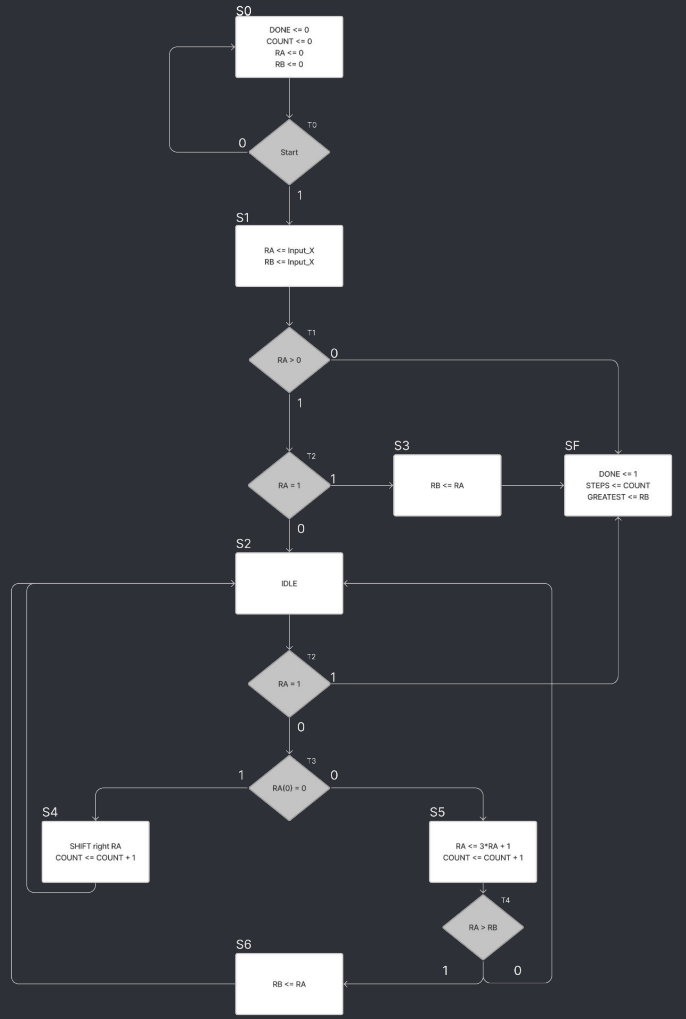
Fluxograma ASM

Uma representação gráfica!

Fluxograma ASM

No fluxograma, foram criados oito estados diferentes para que fosse possível lidar com qualquer número natural inteiro x recebido. Dentre todos os estados utilizados, podemos citar brevemente alguns:

- O estado S0 é responsável por resetar os registradores e aguardar o sinal de start.
- O estado SF é o sinal de done. Sinalizando o fim do algoritmo.
- O estado S2 é o estado IDLE, o qual terá papel fundamental durante o loop de operações envolvendo a variável x.
- Os estados S4 e S5 realizam as operações para caso o número x seja par e ímpar, respectivamente.

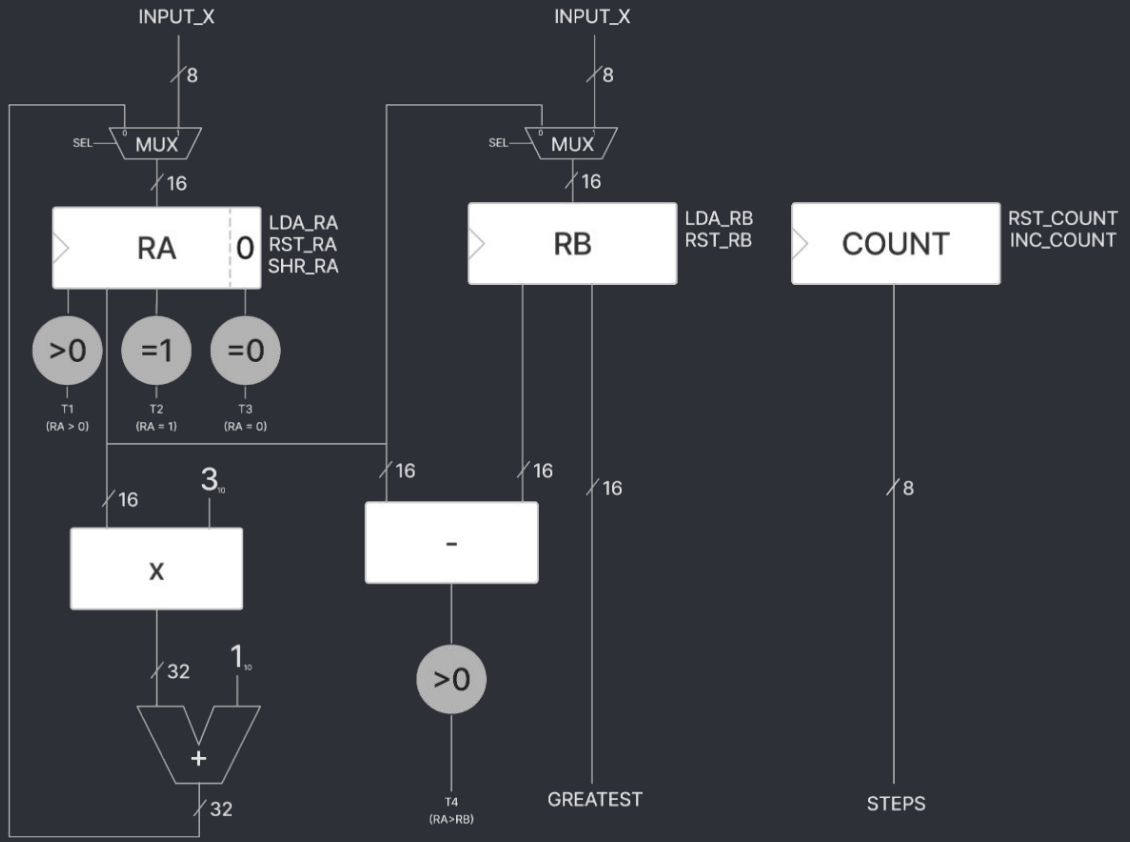


3

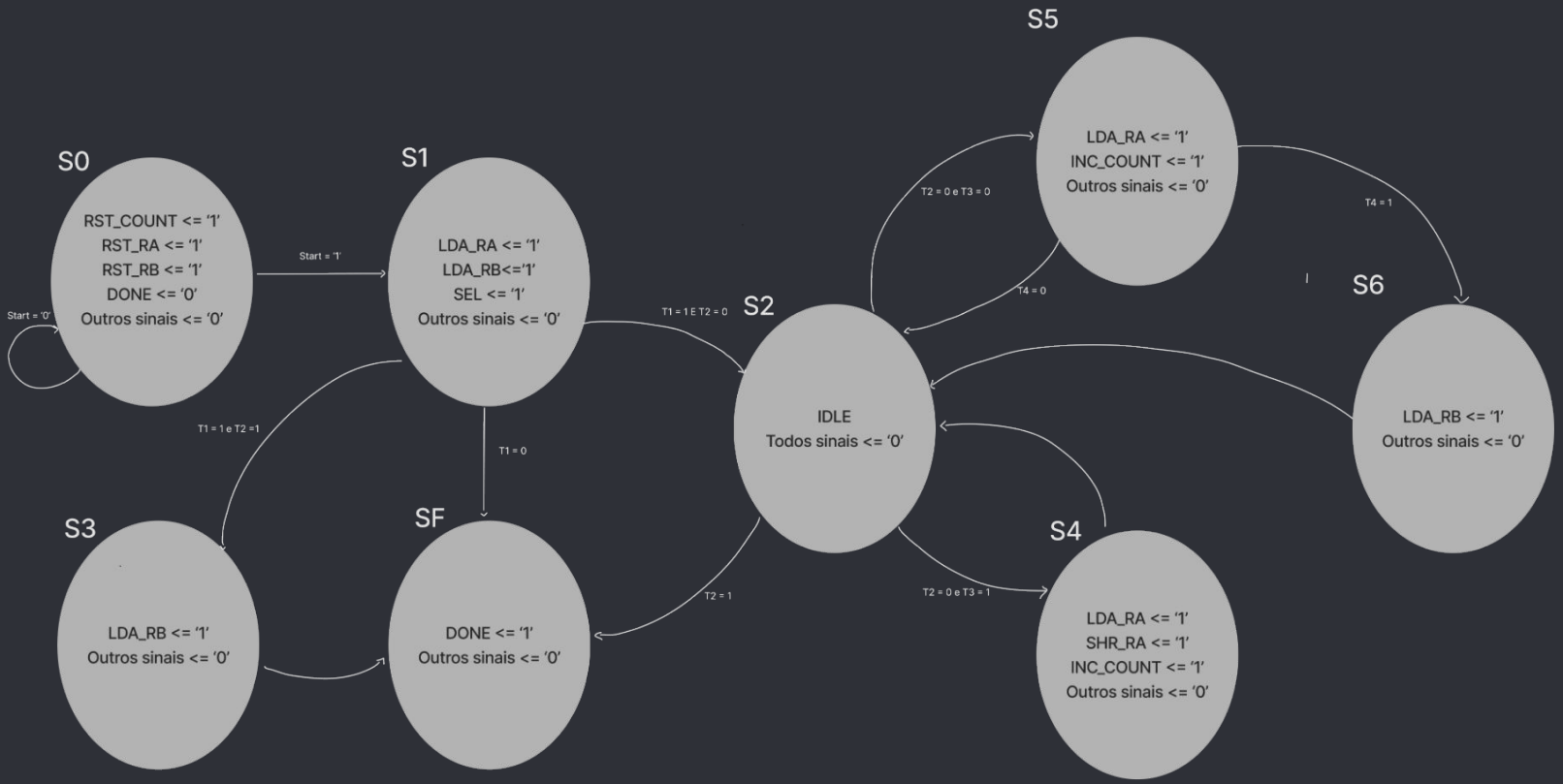
Parte Operativa e Parte de Controle

Diagrama PC-PO!

Parte Operativa



Parte de Controle



4

VHDL do PC-PO

Hora do código!

VHDL do PC-PO

Apenas uma parte dele!

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Collatz_VHDL is
    Port ( start : in STD_LOGIC;
           rst : in STD_LOGIC;
           clk : in  STD_LOGIC;
           inputX : in STD_LOGIC_VECTOR (7 downto 0);
           done : out STD_LOGIC;
           greatest : out STD_LOGIC_VECTOR (15 downto 0);
           steps : out STD_LOGIC_VECTOR (7 downto 0));
end Collatz_VHDL;

architecture Behavioral of Collatz_VHDL is
    --MUX
    signal SEL: STD_LOGIC := '0';
    signal outputMuxA: STD_LOGIC_VECTOR (15 downto 0);
    signal outputMuxB: STD_LOGIC_VECTOR (15 downto 0);
    --RA
    signal LDA_RA: STD_LOGIC := '0';
    signal RST_RA: STD_LOGIC := '0';
    signal SHR_RA: STD_LOGIC := '0';
    signal outputRA: STD_LOGIC_VECTOR (15 downto 0);
    --RB
    signal LDA_RB: STD_LOGIC := '0';
    signal RST_RB: STD_LOGIC := '0';
    signal outputRB: STD_LOGIC_VECTOR (15 downto 0);
    --COUNT
    signal INC_COUNT: STD_LOGIC := '0';
    signal RST_COUNT: STD_LOGIC := '0';
    signal outputCOUNT: STD_LOGIC_VECTOR (7 downto 0);
    --TESTS
    signal outputT1: STD_LOGIC := '0';
    signal outputT2: STD_LOGIC := '0';
    signal outputT3: STD_LOGIC := '0';
    signal outputT4: STD_LOGIC := '0';
```

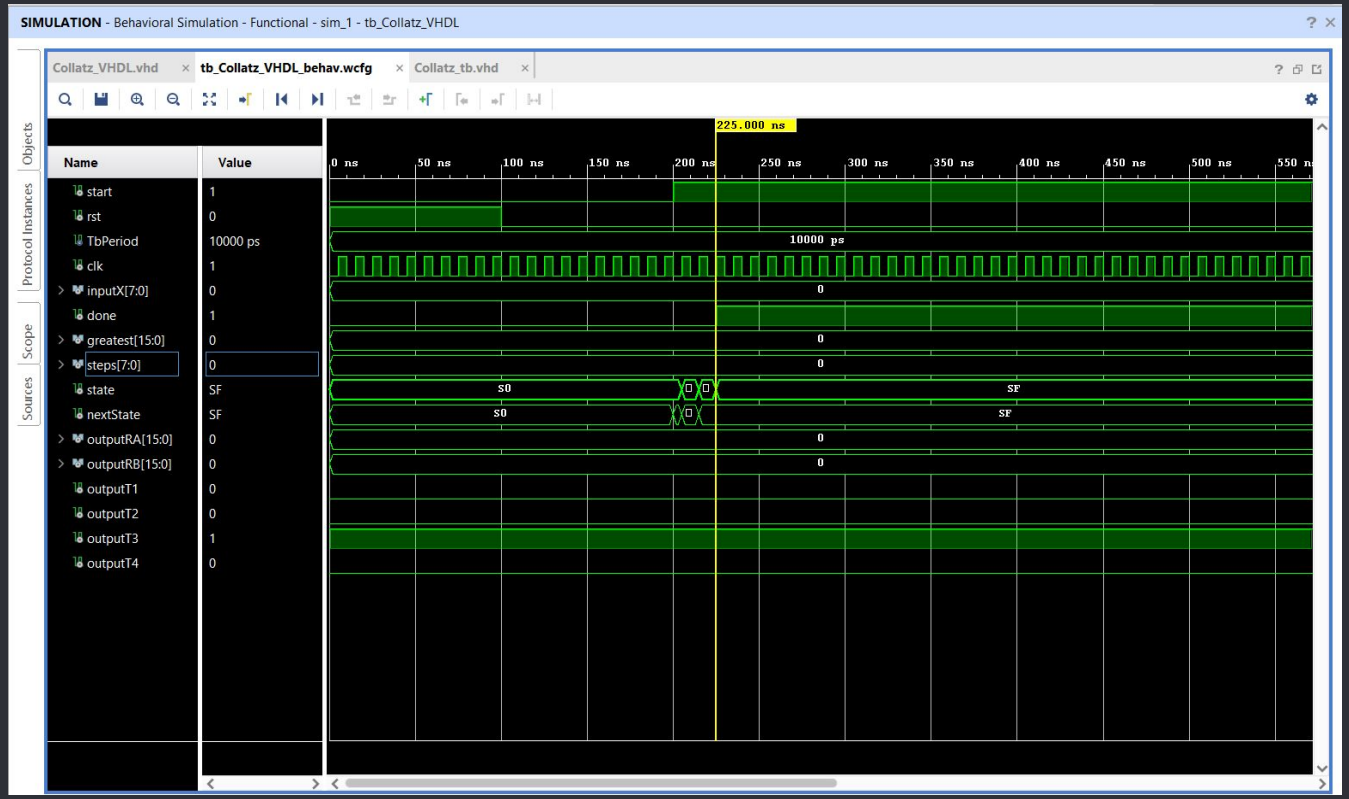
```
-- Estados
    type state_type is (S0, S1, S2, S3, S4, S5, S6, SF, S1T, S2T, S5T);
    signal state, nextState : state_type;

    begin
    -- PARTE OPERATIVA
    -- PROCESS MUX
    MUX : process(SEL, inputX, outputRA)
    variable inputMUXA : STD_LOGIC_VECTOR(31 downto 0);
    begin
      if  (SEL = '1') then
        outputMuxA(15 downto 8) <= "00000000";
        outputMuxA(7 downto 0) <= inputX;
        outputMuxB(15 downto 8) <= "00000000";
        outputMuxB(7 downto 0) <= inputX;
      else
        inputMuxA := STD_LOGIC_VECTOR(unsigned(outputRA) * 3 + 1);
        outputMUXA <= inputMUXA(15 downto 0);
        outputMuxB <= outputRA;
      end if;
    end process;
    -- PROCESS RA
    RA : process(clk, rst)
    begin
      if rst = '1' then
            outputRA <= "0000000000000000";
      elsif rising_edge(clk) then
        if (LDA_RA = '1') then
          if (SHR_RA = '1') then
            outputRA <= STD_LOGIC_VECTOR(shift_right(unsigned(outputRA),1));
          else
            outputRA <= outputMuxA;
          end if;
        elsif (RST_RA = '1') then
          outputRA <= "0000000000000000";
        else
          outputRA <= outputRA;
        end if;
      end if;
    end process;
```

Além do código em VHDL, também foram realizadas algumas simulações, as quais podem ser vistas a seguir:

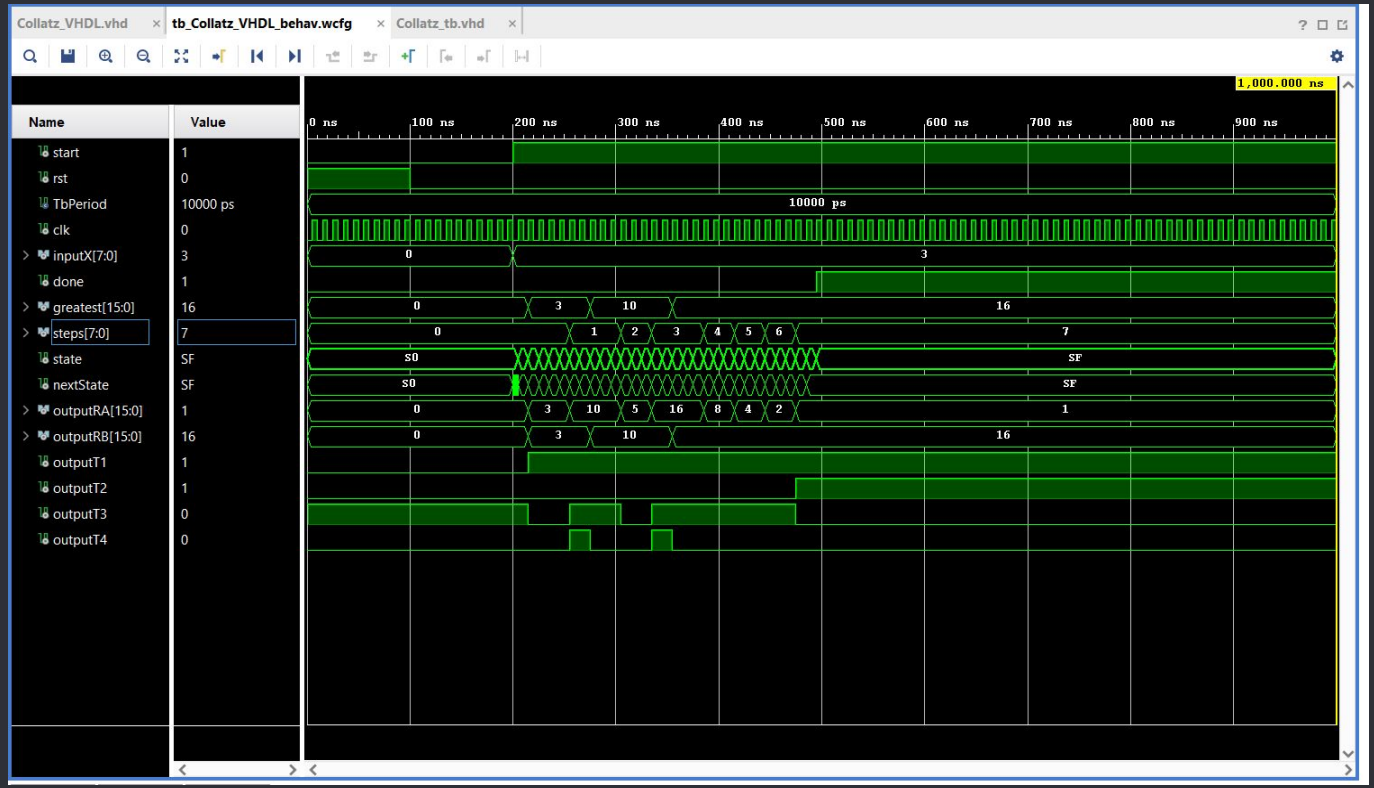
VHDL do PC-PO

Para x = 0



VHDL do PC-PO

Para x = 3



5

Algorítmo em C

Um pouco de C

Algorítmo em C

```c
#ifndef __COLLATZ_H__
#define __COLLATZ_H__
#define input 3


typedef unsigned char Uint8; //8bits
typedef unsigned short Uint16; //16bits
typedef struct Output{
    Uint8 steps;
    Uint8 greatest;
}Output;

// Prototype of top level function for C-synthesis
Output Collatz(Uint8 x);

#endif
```

Arquivo .h

Algorítmo em C

```c
#include "Collatz.h"

Output Collatz(Uint8 x){
    x = (Uint8)input;
    Uint16 aux = (Uint16)x;
    Uint16 greatest = (Uint16)x;
    Uint8 steps;

    if (aux > (Uint16)0){
        loop: for(steps = (Uint8)0; aux != (Uint16)1; steps++){
            if (aux % (Uint16)2 == 0){
                aux = aux >> 1;
            } else{
                aux = (aux * (Uint16)3) + (Uint16)1;
            }
            if (aux > greatest){
                greatest = aux;
            }
        }
    }
    Output out = {greatest, steps};
    return out;
}
```

Arquivo .cpp

6

Tabela comparativa

E por fim...

Tabela comparativa

Versão	PC-PO	HLS
# LUTs	71	85
# ffps	60	18
# DSP	0	0
# BRAM	0	0
# BUFG	1	0
Tclk	10	3
# cc	49	8

Obrigado pela atenção!