



**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

Relatório Atividade extra-classe 4 Conversor de binário para BCD

Dispositivos lógicos programáveis

Rhenzo Hideki Silva Kajikawa

26 de Setembro de 2023

Sumário

1. Comando da Atividade extra-classe 4 (AE4):	3
1.1. Resolução da Atividade extra-classe 4 (AE4)	6
1.2. Código utilizado	6
1.3. Simulação funcional	7

1. Comando da Atividade extra-classe 4 (AE4):

Neste laboratório remoto, os alunos deverão implementar uma solução do para um circuito conversor de binário para BCD (bin2bcd) com entrada binária variando entre 0 a 9999.

- Baseado no exemplo do conversor de binário para BCD - Binary-coded decimal de dois dígitos decimais (00 a 99), mostrado em aula, projete um conversor para 4 dígitos (0000 a 9999).
- Escreva o código em VHDL, que dada uma entrada A (entre 0 e 9999), fornece nas saídas os dígitos da milhar (sm), centena (sc), dezena (sd) e unidade (su).
- Utilize as diferentes estratégias ensinadas para reduzir a quantidade de elementos lógicos, aproveitando resultados intermediários, e definindo com exatidão o número de bits a ser usado. O uso de configurações diferentes no compilador Quartus Prime 20.1.1, uso de restrições de tempo através de comandos no arquivo .SDC, e escolha do dispositivo da família de FPGA CYCLONE IV E é permitida.
- Realize a Simulação Funcional usando o ModelSim para mostrar que o circuito funciona.

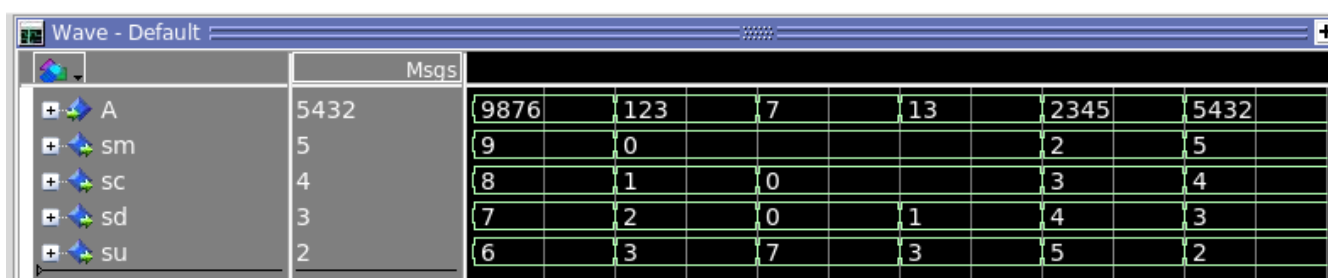


Figura AE4 1: Exemplo de simulação funcional de 0 a 9999

- Analise o tempo de propagação e área ocupada (número de elementos lógicos) e tente otimizar um ou os dois parâmetros. Se realizar diversas versões, pode anotar os valores de todas elas e fornecer todas as versões, mas foque no melhor desempenho.
- O número de elementos lógicos pode ser obtido no Flow Summary ou no Resource Usage Summary, conforme mostram as figuras a seguir. Anote a quantidade de elementos lógicos do circuito.

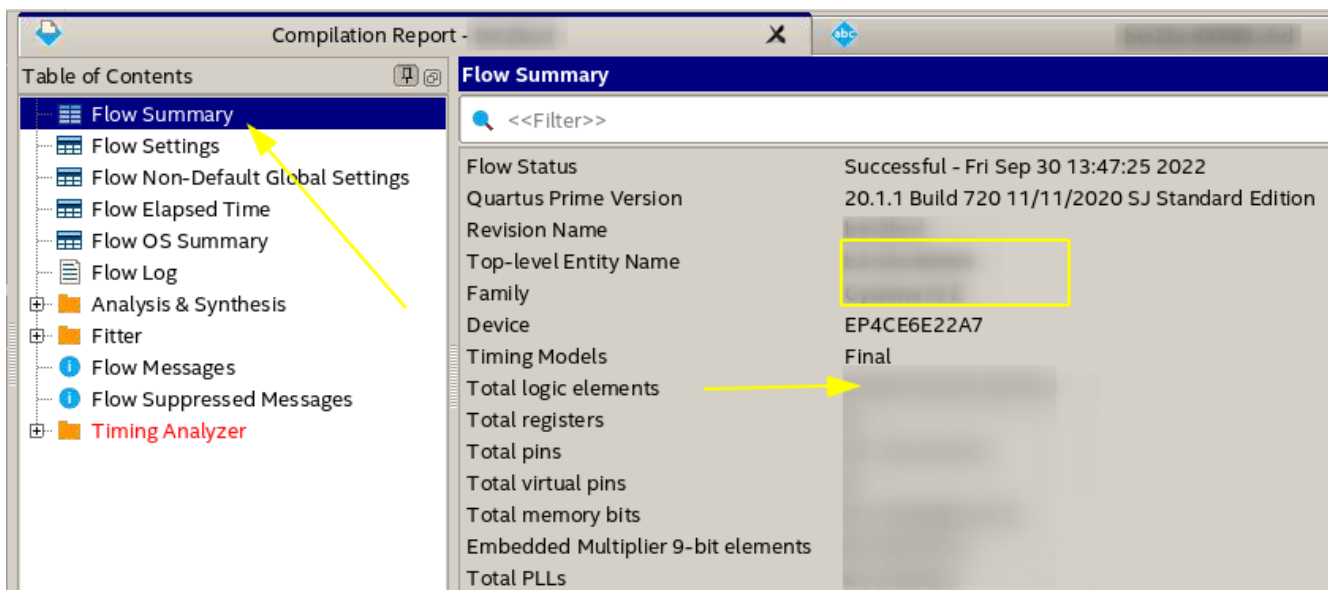


Figura AE4 2: Obtendo o número de elementos no “Flow Summary”

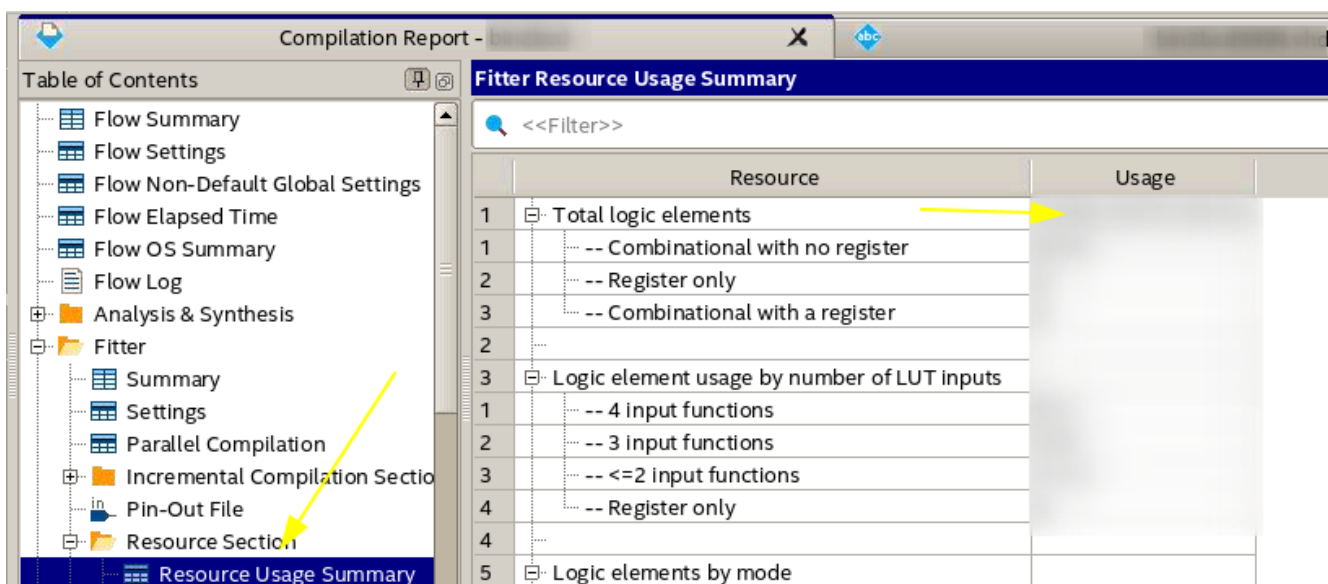


Figura AE4 3: Obtendo o número de elementos no “Resource Usage Summary”

- O tempo máximo de propagação do circuito é obtido no Report Datasheet dentro do aplicativo Timing Analyser .
- Antes de abrir o Timing Analyser é necessário realizar as etapas Analysis & Synthesis, Fitter e Timing Analysis.
- Em seguida no aplicativo Timing Analyser, é necessário executar o Create Timing Netlist, Read SDC File e Update Timing Netlist.
- Selecione o Set Operation Conditions para o modelo Slow 1200mV 125°C, pois corresponde ao pior tempo dos 3 modelos de simulação.
- Em seguida obtenha Report Datasheet. Anote o tempo máximo de propagação do circuito.

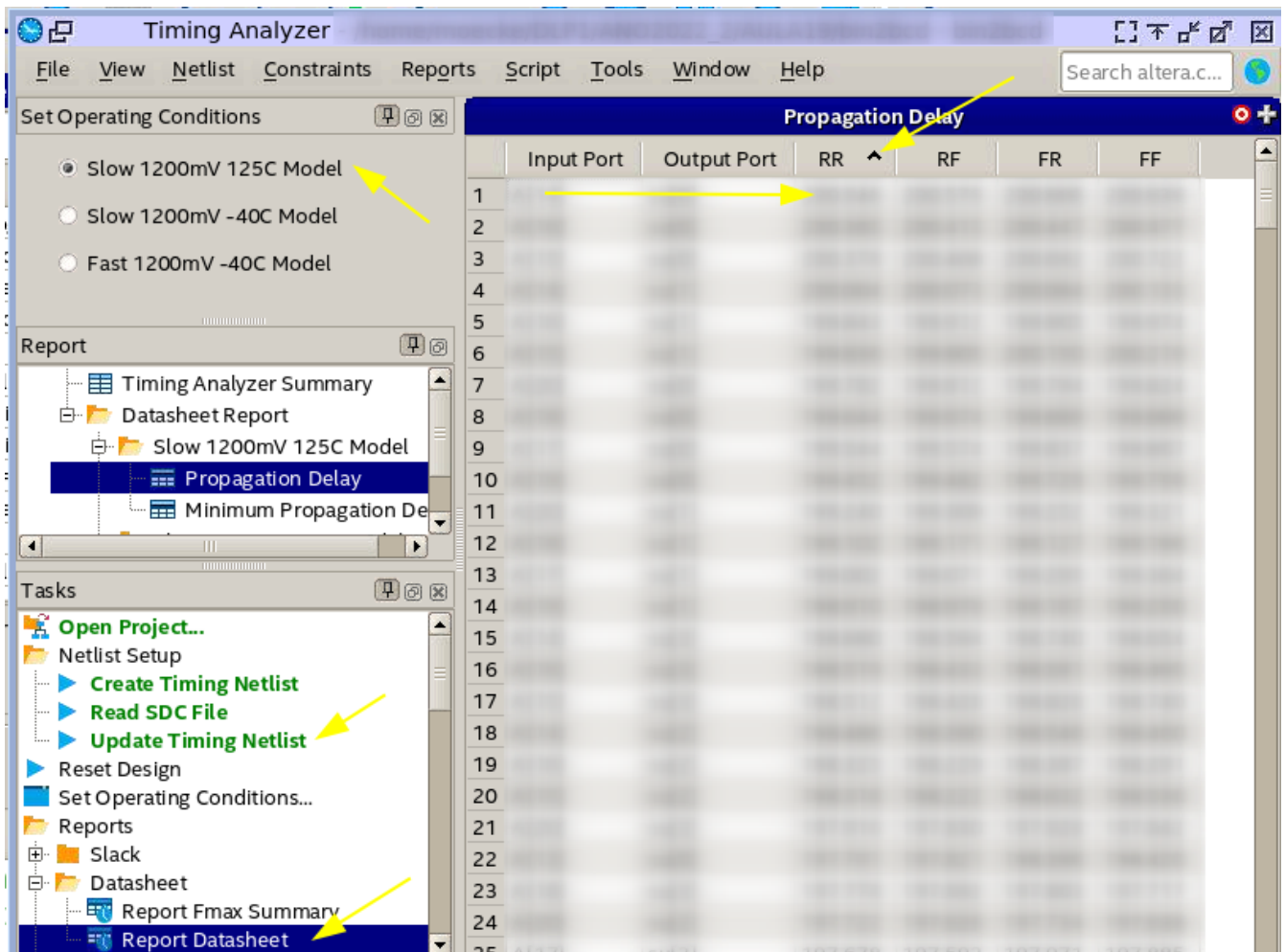


Figura AE4 4: Exemplo de tempo máximo de propagação

- Se quiser o(a) estudante pode apresentar dois projetos, sendo um para o menor tempo máximo de propagação e outro para menor área ocupada (número de elementos lógicos).
- O arquivo QAR entregue deve ser plenamente compilável e permitir após a Análise e Síntese e execução do comando de simulação do tb_bin2bcd.do deve apresentar o resultado final.
- Neste laboratório é necessário fornecer a imagem RTL e Technology Map usadas para obter e melhorar os circuitos, e a imagem da simulação que mostra que a versão entregue funciona.
- Não é permitido o uso do algoritmo Double Dabble para fazer a conversão entre binário e BCD.

1.1. Resolução da Atividade extra-classe 4 (AE4)

Seguindo as orientações da atividade , foi feito um código conversor de binário para BCD (bin2bcd) com entrada binária variando entre 0 a 9999.

1.2. Código utilizado

o código feito foi este:

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

entity bin2bcd is
  port (
    A : in std_logic_vector(14 downto 0);
    sm : out std_logic_vector( 4 downto 0 );
    sc : out std_logic_vector( 4 downto 0 );
    sd : out std_logic_vector( 4 downto 0 );
    su: out std_logic_vector( 4 downto 0 )
  );
end entity;

architecture ae4 of bin2bcd is

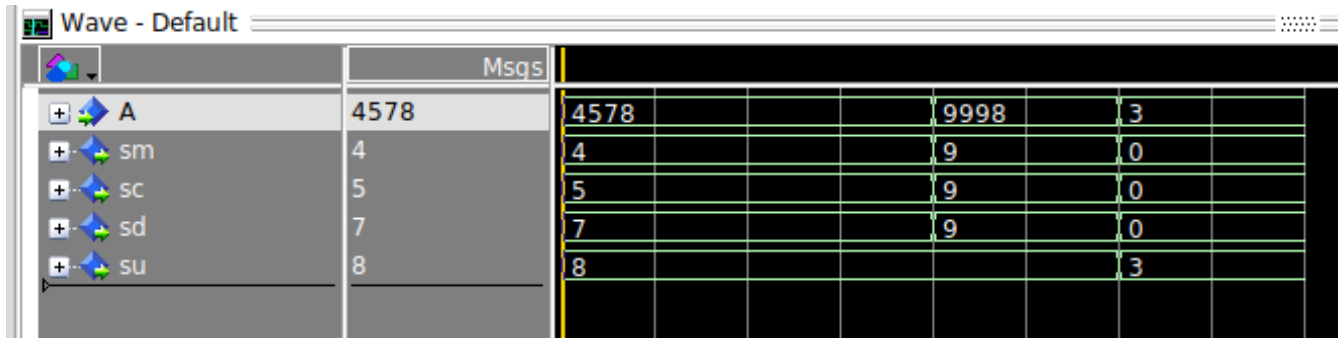
  signal A_uns : unsigned(14 downto 0);
  signal slice_mil: unsigned(14 downto 0);
  signal slice_cem: unsigned(14 downto 0);
  signal slice_dez: unsigned(14 downto 0);
  signal slice_uni: unsigned(14 downto 0);

begin
  A_uns <= unsigned(A);
  sm <= std_logic_vector(resize(slice_mil,5));
  sc <= std_logic_vector(resize(slice_cem,5));
  sd <= std_logic_vector(resize(slice_dez,5));
  su <= std_logic_vector(resize(slice_uni,5));
  -- Convert each binary digit to BCD
  slice_mil <= A_uns/1000;
  slice_cem <= (A_uns/100) rem 10;
  slice_dez <= (A_uns/10) rem 10 ;
  slice_uni <= A_uns rem 10;
end architecture;
```

O código foi baseado nos código feitos em aula junto com o conhecimento adquirido. Utilizando 4 saídas std_logic_vector sm (Sinal milhar), sc (Sinal centena), sd (Sinal dezena), su (Sinal unidade), e utilizando uma entrada A . slice_mil , slice_cem , slice_dez , slice_uni são os intermediários para trocar de sinal não sinalizado (unsigned).

1.3. Simulação funcional

Utilizando esse código , foi possível obter a Simulação funcional usando o ModelSim de acordo com o comando da questão , desta forma foi feito alguns testes para testar o código , este foi o resultado obtido :



Component	Value
A	4578
sm	4
sc	5
sd	7
su	8

Msgs	Value
4578	9998
4	9
5	9
7	9
8	3

Figura AE4 5: simulação funcional