



**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
2. Resolução da Atividade extra-classe 4 (AE4)	6
2.1. Código utilizado	6
2.2. Simulação funcional	7
2.3. Número de elementos lógicos	7
2.4. Tempo de propagação	9
2.5. RTL Viewer	10
2.6. Technology Map	11
3. Conclusão	12

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.

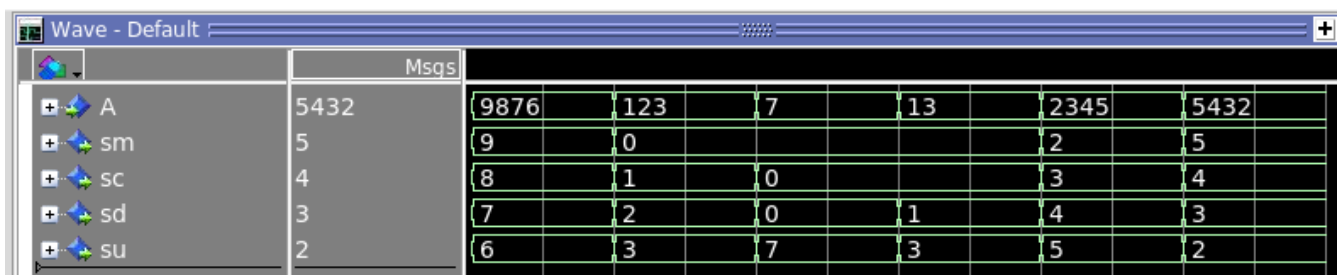


Figura AE4 1: Exemplo de simulação funcional de 0 a 9999

Fonte: Marcos Moecke

- 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.

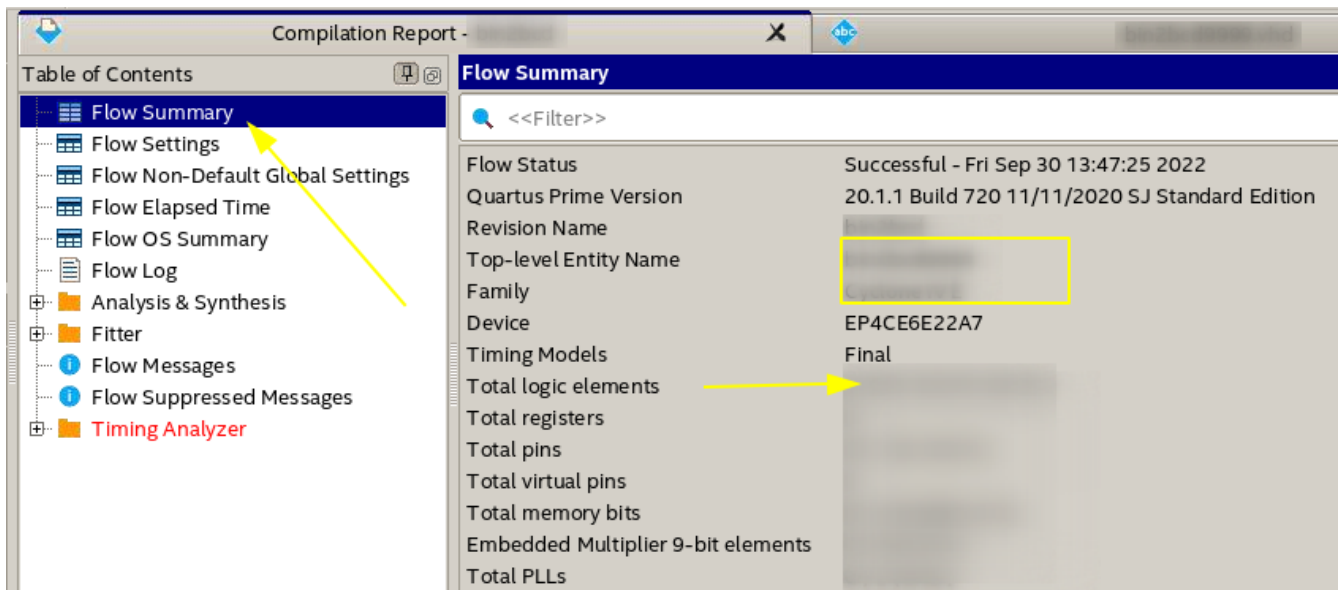


Figura AE4 2: Obtendo o número de elementos no “Flow Summary”

Fonte: Marcos Moecke

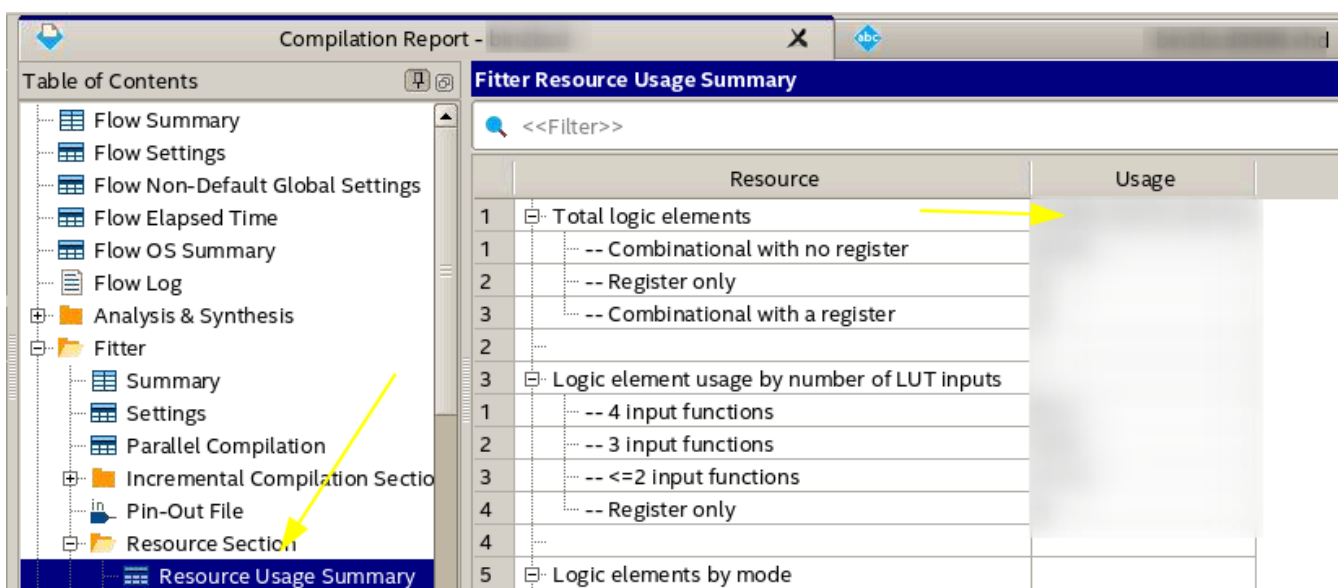


Figura AE4 3: Obtendo o número de elementos no “Resource Usage Summary”

Fonte: Marcos Moecke

- 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.

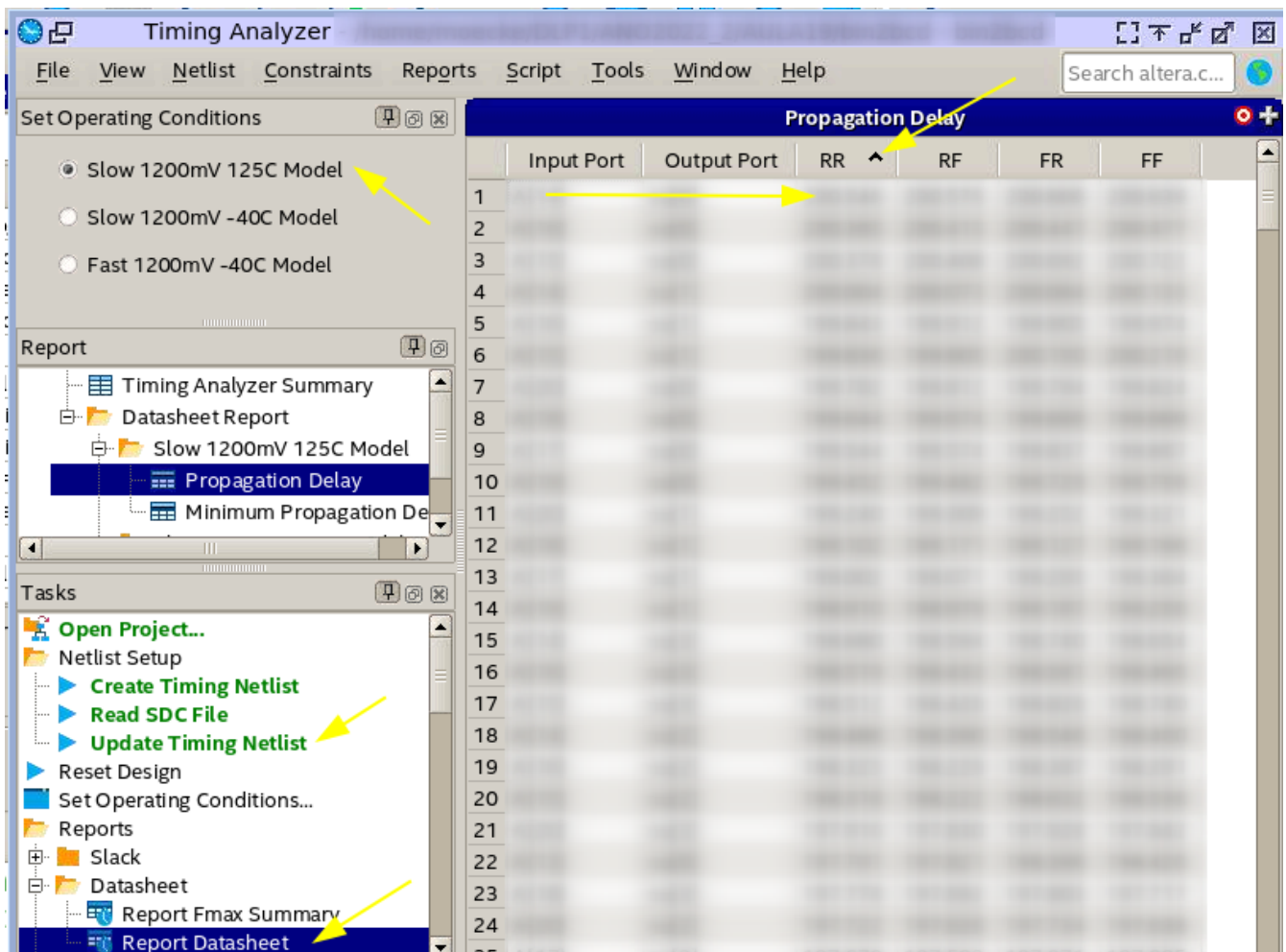


Figura AE4 4: Exemplo de tempo máximo de propagação

Fonte: Marcos Moecke

- 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.

2. 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. A família utilizada foi Cyclone IV E e a placa escolhida foi a EP4CE115F29C8 , estando de acordo com as orientações anteriores.

2.1. 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

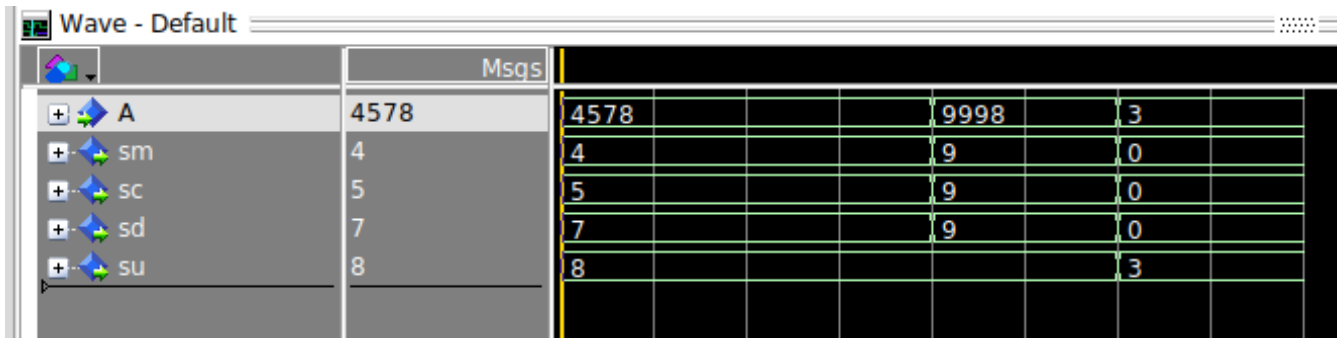
  signal A_uns : unsigned(14 downto 0);
  signal slice_mil: unsigned(14 downto 0);
  signal slice_cem: unsigned(14 downto 0);
  signal slice_dez: unsigned(14 downto 0);
  signal slice_uni: unsigned(14 downto 0);

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).

2.2. 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 :



Signal	Value 1	Value 2	Value 3	Value 4	Value 5
A	4578	4	5	7	8
sm	9998	3	0	0	0
sc	9998	3	0	0	0
sd	9998	3	0	0	0
su	9998	3	0	0	0

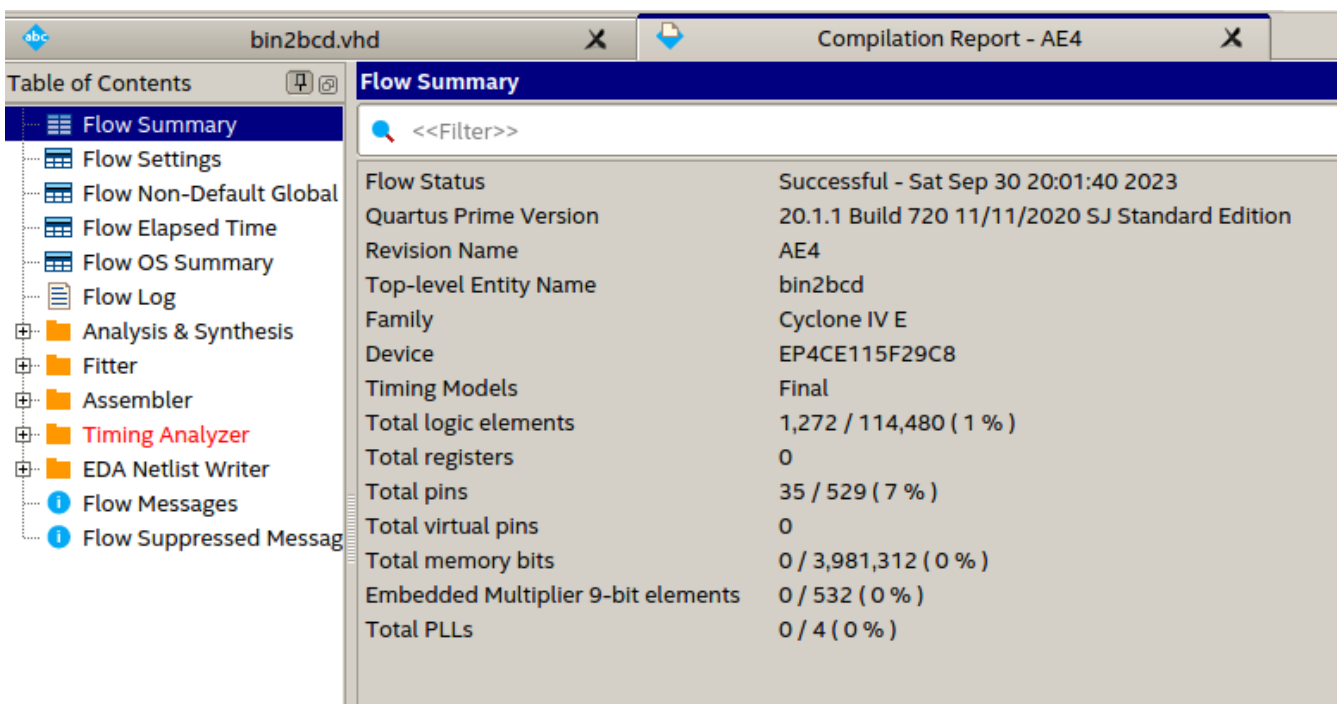
Figura AE4 5: simulação funcional

Fonte: Elaborada pelo autor

Foi analisado 3 valores nessa simulação com A sendo 4578, 9998, 0003, é possível ver que os valores de sm, sc, sd, su foram alterados nos momentos que A recebeu os valores de entrada. Os resultados satisfazem o objetivo do código e da atividade extra-classe 4.

2.3. Número de elementos lógicos

Com a simulação funcional feita , é possível ter certeza ver que foi alcançado o objetivo em código , mas é necessária a análise de quão custoso o código está sendo e se é aceitável o número de recursos. As figuras a seguir mostram o número de recursos utilizados para que o código seja implementado:



Category	Value
Flow Status	Successful - Sat Sep 30 20:01:40 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Standard Edition
Revision Name	AE4
Top-level Entity Name	bin2bcd
Family	Cyclone IV E
Device	EP4CE115F29C8
Timing Models	Final
Total logic elements	1,272 / 114,480 (1 %)
Total registers	0
Total pins	35 / 529 (7 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figura AE4 6: simulação funcional

Fonte: Elaborada pelo autor

bin2bcd.vhd Compilation Report - AE4

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
 - Fitter
 - Summary
 - Settings
 - Parallel Compilation
 - Incremental Compilation Section
 - Pin-Out File
 - Resource Section
 - Resource Usage Summary**
 - Partition Statistics
 - Input Pins
 - Output Pins
 - Dual Purpose and Dedicated Pins
 - I/O Bank Usage
 - All Package Pins
 - I/O Standards Section
 - Resource Utilization by Entity
 - Delay Chain Summary
 - Pad To Core Delay Chain Fanout

Fitter Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Total logic elements	1,272 / 114,480 (1 %)
1	-- Combinational with no register	1272
2	-- Register only	0
3	-- Combinational with a register	0
2		
3	Logic element usage by number of LUT inputs	
1	-- 4 Input functions	277
2	-- 3 Input functions	380
3	-- <=2 Input functions	615
4	-- Register only	0
4		
5	Logic elements by mode	
1	-- normal mode	864
2	-- arithmetic mode	408
6		
7	Total registers*	0 / 117,053 (0 %)
1	-- Dedicated logic registers	0 / 114,480 (0 %)
2	-- I/O registers	0 / 2,573 (0 %)
8		

* Register count does not include registers inside RAM blocks or DSP blocks.

Figura AE4 7: simulação funcional

Fonte: Elaborada pelo autor

Esses são os registros da quantidade de recurso utilizada para funcionamento se baseando no código anterior. o valor representa 1% do total de elementos lógicos da placa , porém utiliza 1272 elementos lógicos. Após algumas tentativas tentando otimizar o uso dos elementos lógicos , foi concluído que com os recursos aprendidos até a AE4 não foi observado maneira melhor ou mais intuitiva de executar um código que atendesse as requisições sem alterar outras partes além do código.

2.4. Tempo de propagação

Para essa atividade também foi requisitado que fosse estudado o quão rápido era a execução do código e o tempo de propagação na placa. Esse resultado é mostrado na figura a seguir:

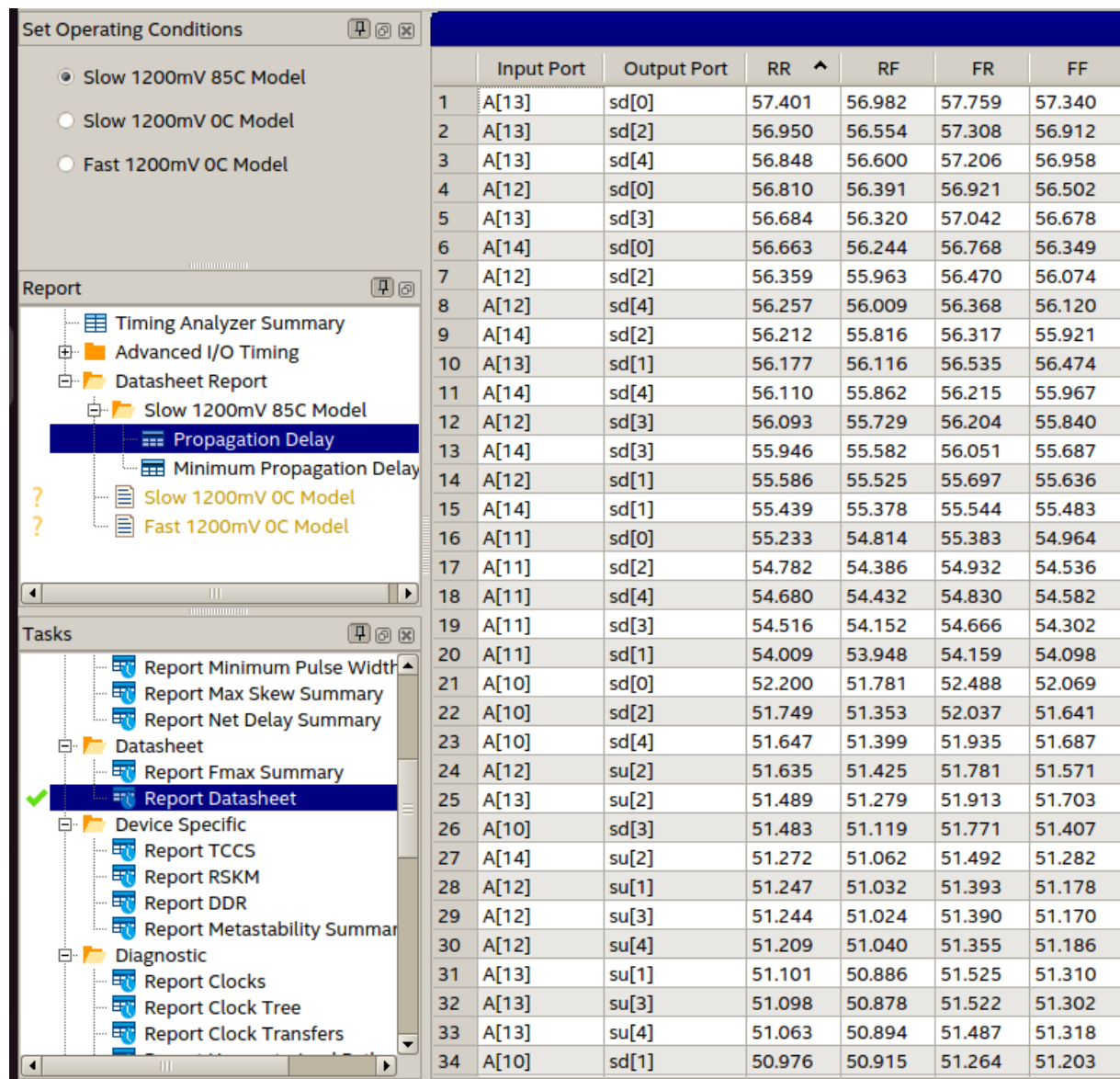


Figura AE4 8: simulação funcional

Fonte: Elaborada pelo autor

Pode-se ver na figura 8 que o tempo de propagação é de 57.4 ns , essa é a pior situação simulada possível com o modelo utilizando sua forma de operação mais lenta , dessa forma pode-se fazer a análise e decidir se é aceitável ou não a propagação . O que é possível fazer para alterar esses valores é alterar o valor de seed das simulações , alterar a placa utilizada mantendo-se dentro da família Cyclone IV E , e procurar por outras otimizações no código e distribuição dos elementos lógicos na placa e diminuir suas distâncias.

2.5. RTL Viewer

A seguir uma imagem do RTL viewer após a compilação do código.

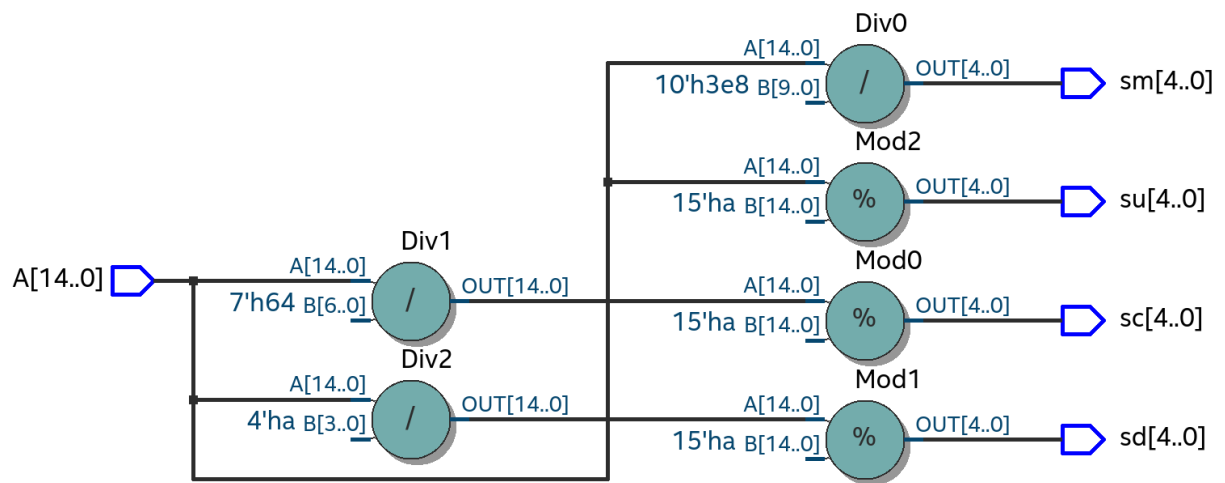


Figura AE4 9: RTL viewer

Fonte: Elaborada pelo autor

É possível ver no RTL Viewer os circuitos utilizados para satisfazer o código. Utilizando 3 circuitos de divisão e mais 3 circuitos de Mod (Resto da divisão inteira) .

2.6. Technology Map

A seguir os prints dos Technology Maps tirados da atividade.

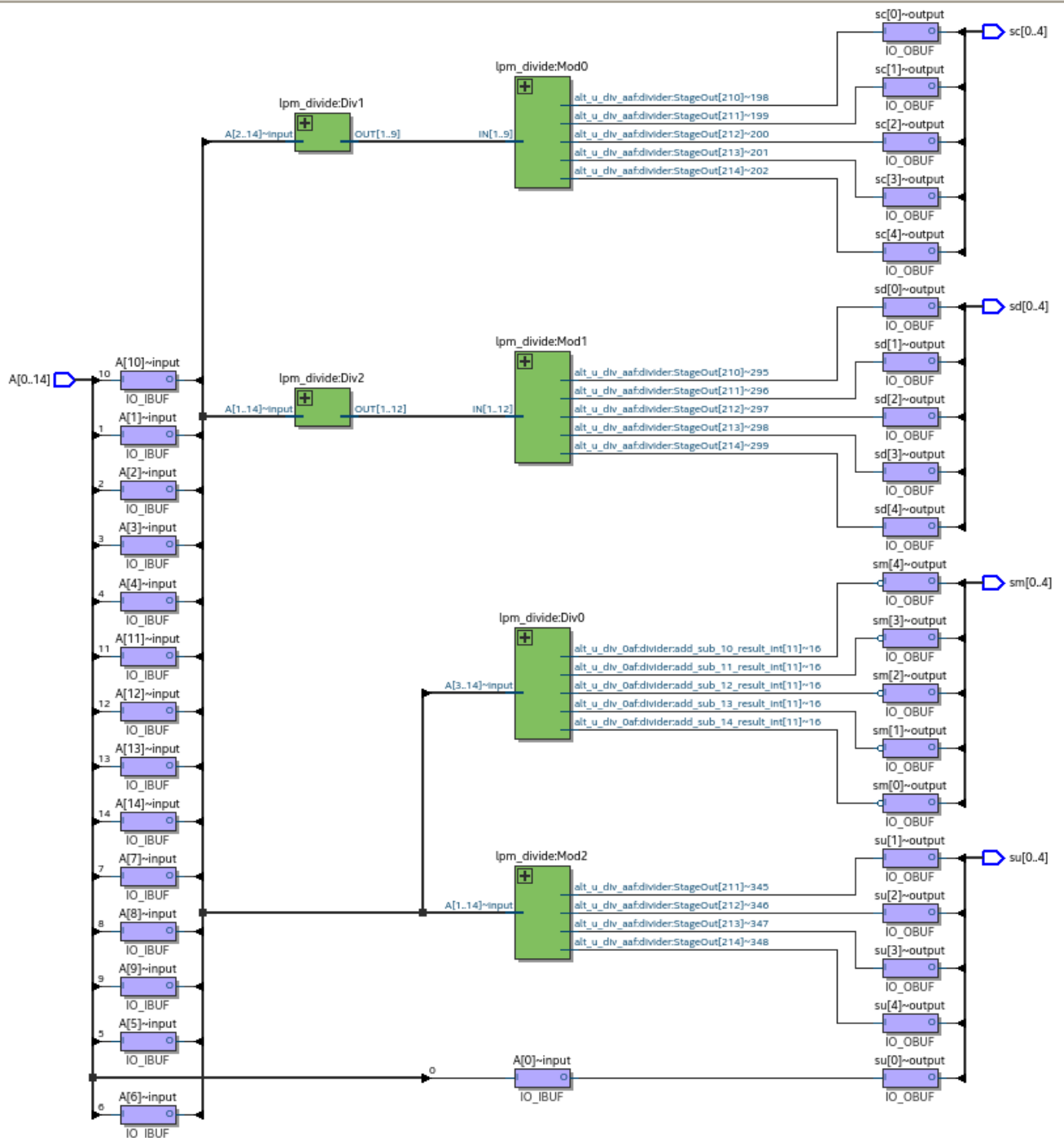


Figura AE4 10: Technology Map Viewer (Post-Fitting)

Fonte: Elaborada pelo autor

