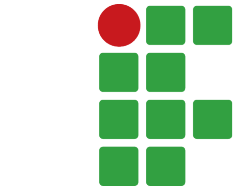


INSTITUTO FEDERAL
Santa Catarina

Câmpus
São José

Relatório Atividade extra-classe 4
Conversor de binário para BCD

Dispositivos lógicos programaveis

Rhenzo Hideki Silva Kajikawa

26 de Setembro de 2023

Sumário

1. Resolução da Atividade extra-classe 4 (AE4) 3
1.1. Código utilizado 3
1.2. Simulação funcional 4
1.3. Número de elementos lógicos 4
1.4. Tempo de propagação 6
1.5. RTL Viewer 7
1.6. Technology Map 8
2. Conclusão 9

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

1.1. 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).

1.2. 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 :

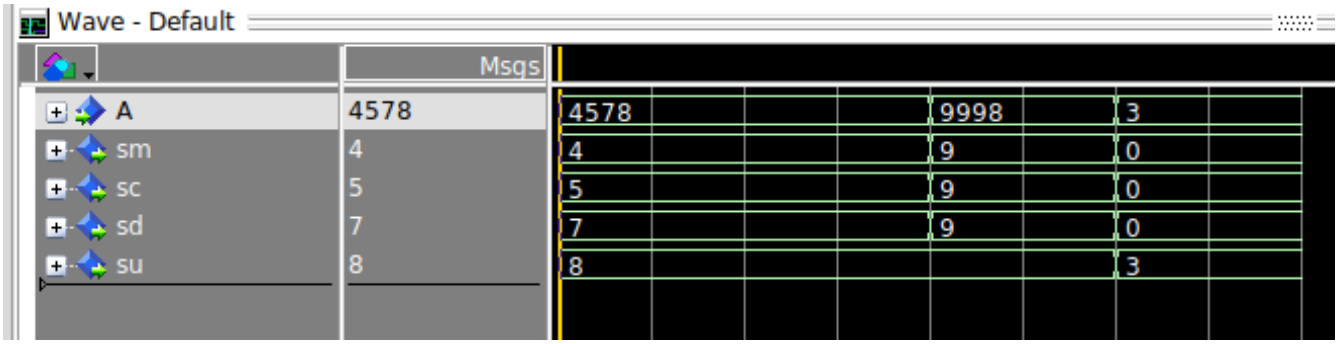


Figura AE4 1: simulação funcional
Fonte: Elaborada pelo autor

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.

1.3. Número de elementos lógicos

Com a simulação funcional feita , é possível ter certeza ver que foi alçado o objetivo em código , mas é necessária a analise de quãao 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:

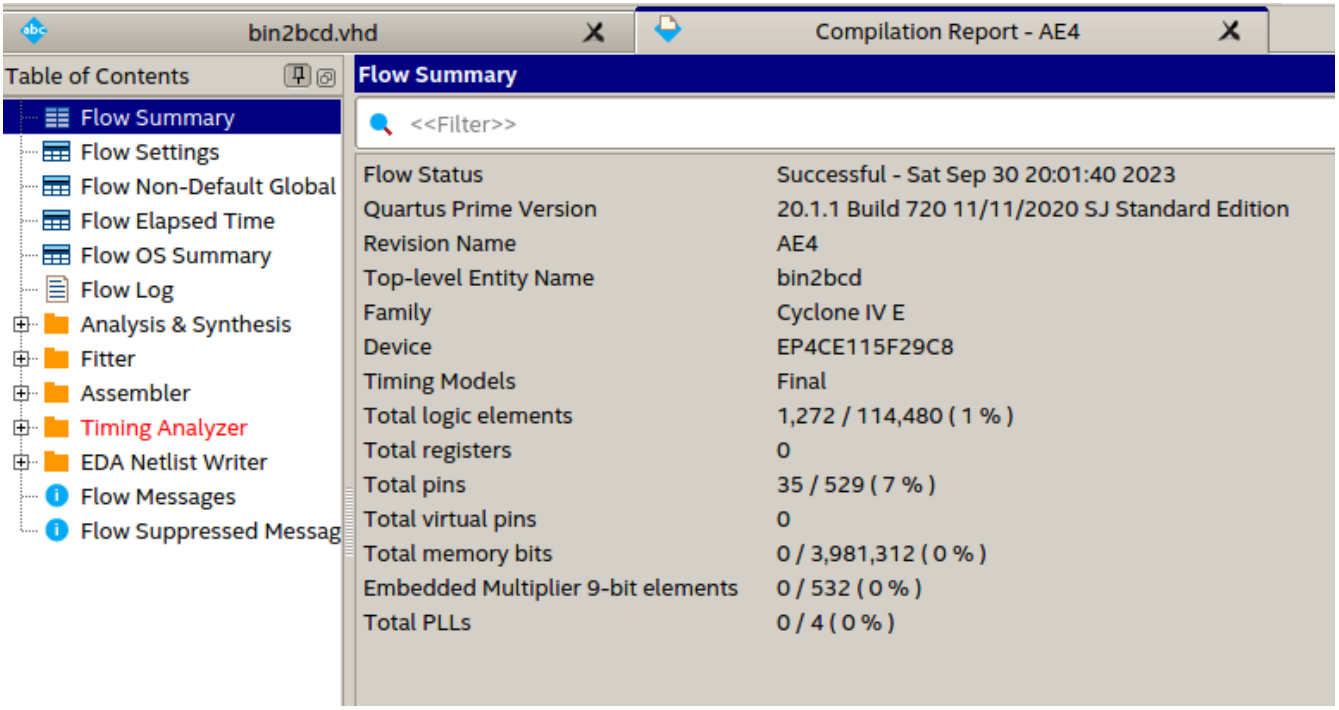


Figura AE4 2: simulação funcional
Fonte: Elaborada pelo autor

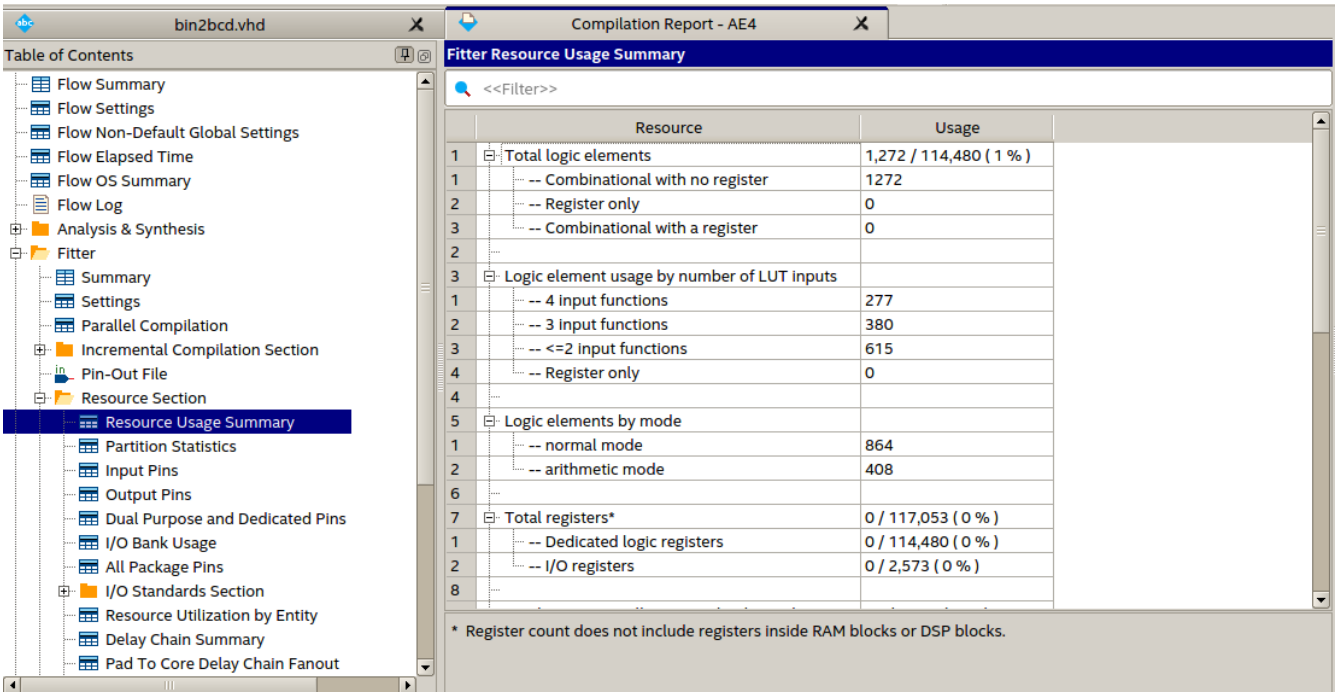
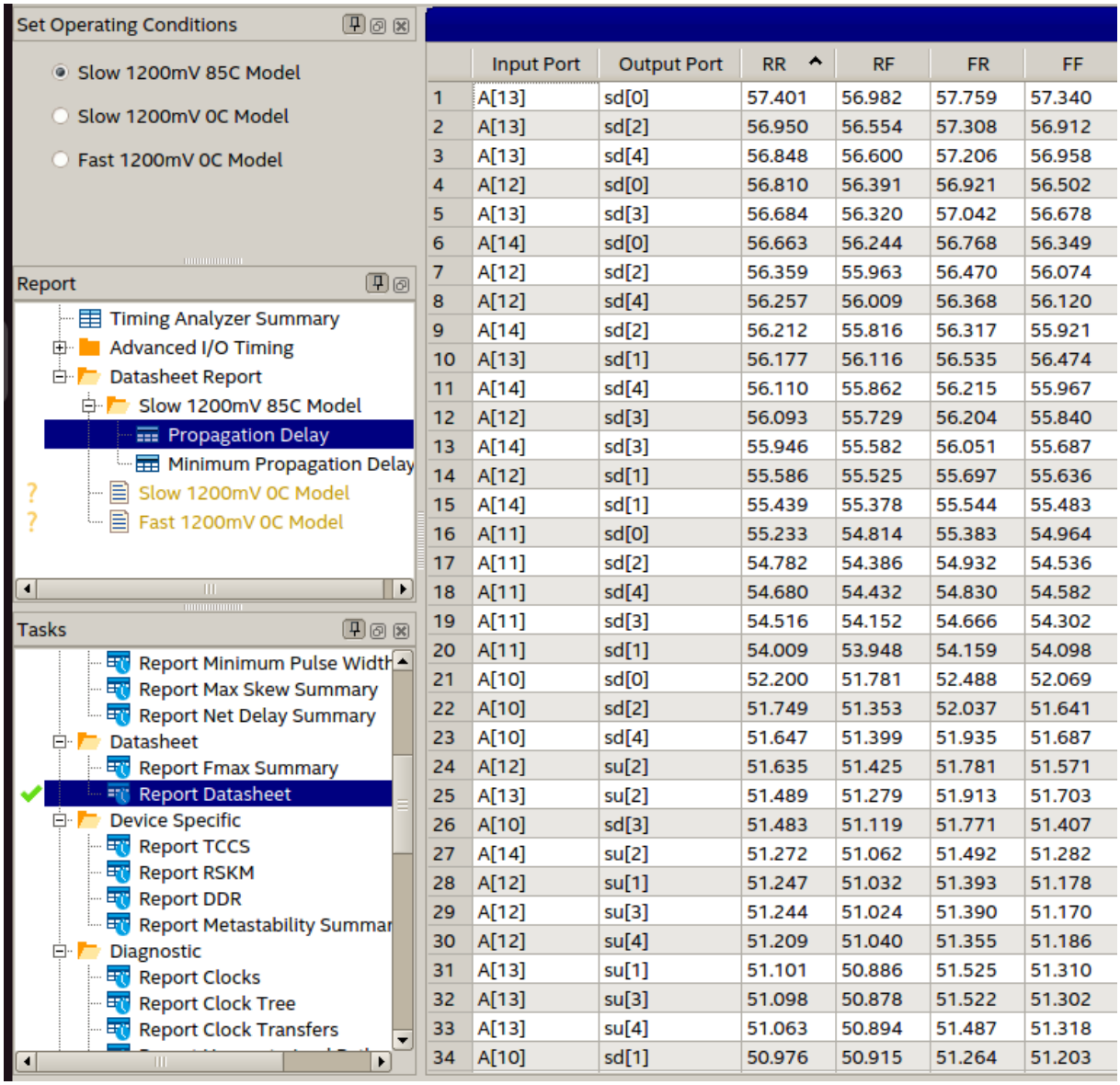


Figura AE4 3: simulação funcional
Fonte: Elaborada pelo autor

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.

1.4. 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:



	Input Port	Output Port	RR	RF	FR	FF
1	A[13]	sd[0]	57.401	56.982	57.759	57.340
2	A[13]	sd[2]	56.950	56.554	57.308	56.912
3	A[13]	sd[4]	56.848	56.600	57.206	56.958
4	A[12]	sd[0]	56.810	56.391	56.921	56.502
5	A[13]	sd[3]	56.684	56.320	57.042	56.678
6	A[14]	sd[0]	56.663	56.244	56.768	56.349
7	A[12]	sd[2]	56.359	55.963	56.470	56.074
8	A[12]	sd[4]	56.257	56.009	56.368	56.120
9	A[14]	sd[2]	56.212	55.816	56.317	55.921
10	A[13]	sd[1]	56.177	56.116	56.535	56.474
11	A[14]	sd[4]	56.110	55.862	56.215	55.967
12	A[12]	sd[3]	56.093	55.729	56.204	55.840
13	A[14]	sd[3]	55.946	55.582	56.051	55.687
14	A[12]	sd[1]	55.586	55.525	55.697	55.636
15	A[14]	sd[1]	55.439	55.378	55.544	55.483
16	A[11]	sd[0]	55.233	54.814	55.383	54.964
17	A[11]	sd[2]	54.782	54.386	54.932	54.536
18	A[11]	sd[4]	54.680	54.432	54.830	54.582
19	A[11]	sd[3]	54.516	54.152	54.666	54.302
20	A[11]	sd[1]	54.009	53.948	54.159	54.098
21	A[10]	sd[0]	52.200	51.781	52.488	52.069
22	A[10]	sd[2]	51.749	51.353	52.037	51.641
23	A[10]	sd[4]	51.647	51.399	51.935	51.687
24	A[12]	su[2]	51.635	51.425	51.781	51.571
25	A[13]	su[2]	51.489	51.279	51.913	51.703
26	A[10]	sd[3]	51.483	51.119	51.771	51.407
27	A[14]	su[2]	51.272	51.062	51.492	51.282
28	A[12]	su[1]	51.247	51.032	51.393	51.178
29	A[12]	su[3]	51.244	51.024	51.390	51.170
30	A[12]	su[4]	51.209	51.040	51.355	51.186
31	A[13]	su[1]	51.101	50.886	51.525	51.310
32	A[13]	su[3]	51.098	50.878	51.522	51.302
33	A[13]	su[4]	51.063	50.894	51.487	51.318
34	A[10]	sd[1]	50.976	50.915	51.264	51.203

Figura AE4 4: 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 analise 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 familia Cyclone IV E , e procurar por outras otimizações no código e distribuição dos elementos lógicos na placa e diminuir suas distancias.

1.5. RTL Viewer

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

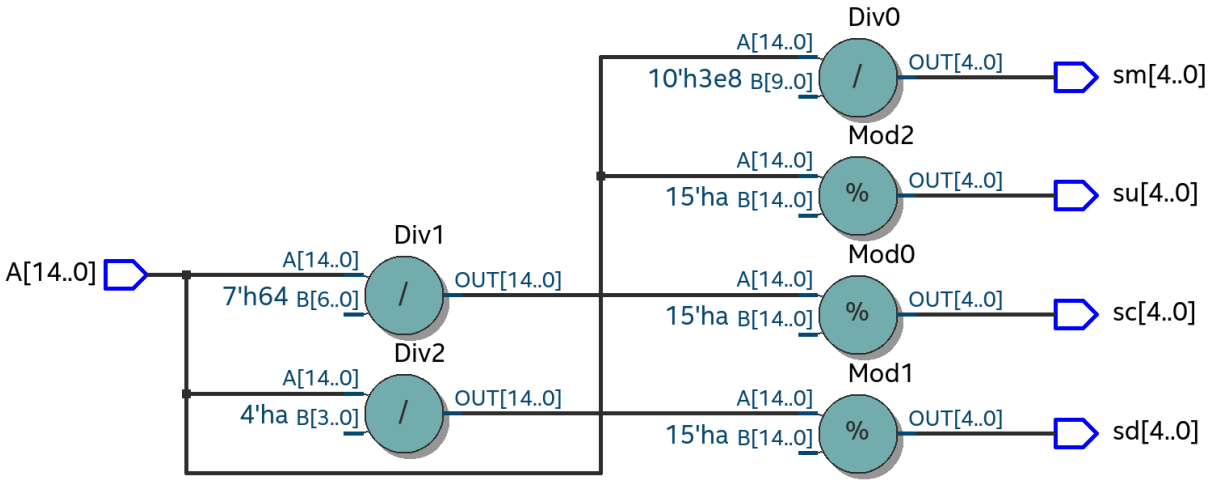


Figura AE4 5: RTL viewer
Fonte: Elaborada pelo autor

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) .

1.6. Technology Map

A seguir os prints dos Technology Maps tirados da atividade.

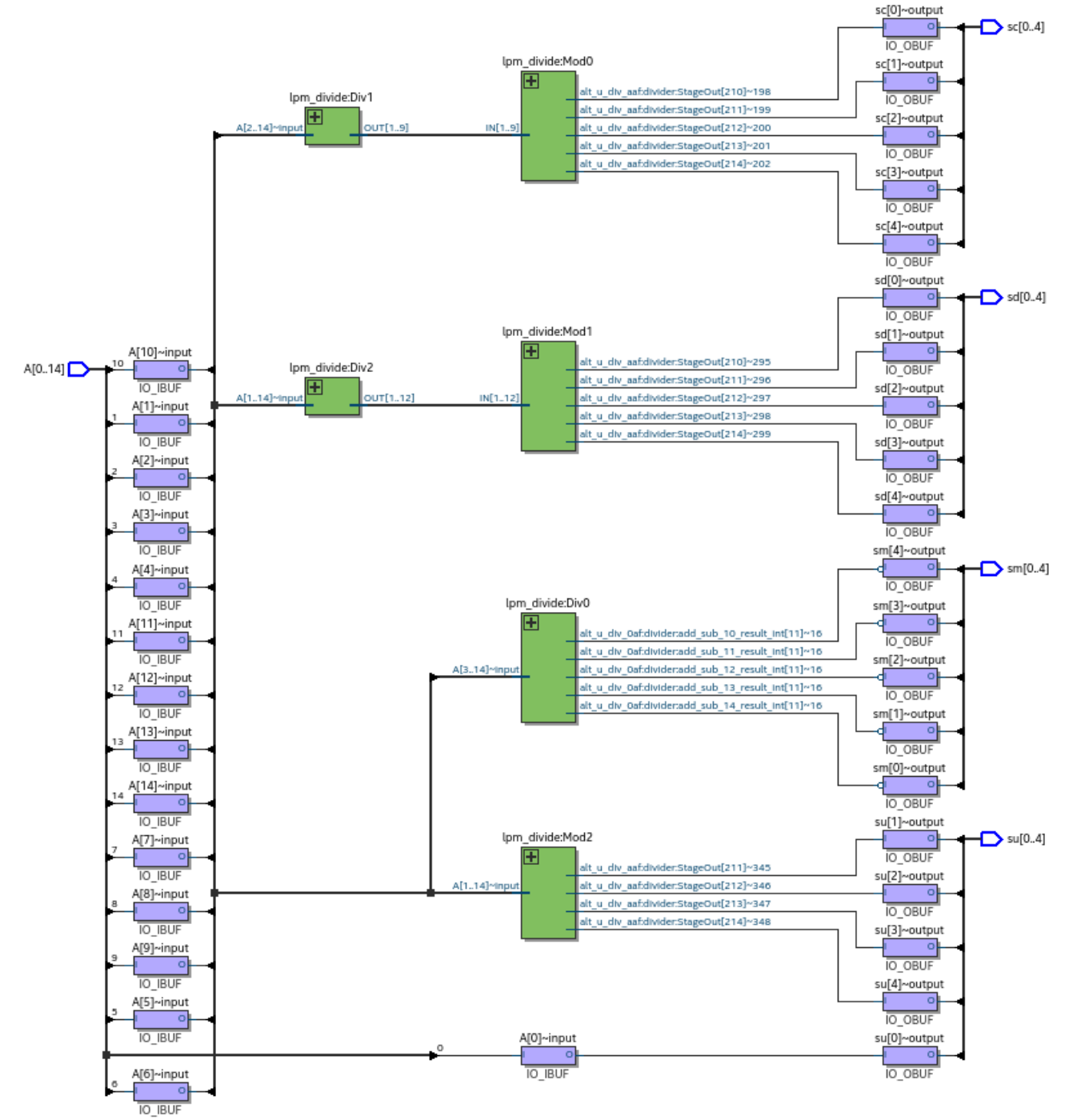


Figura AE4 6: Technology Map Viewer (Post-Fitting)
Fonte: Elaborada pelo autor

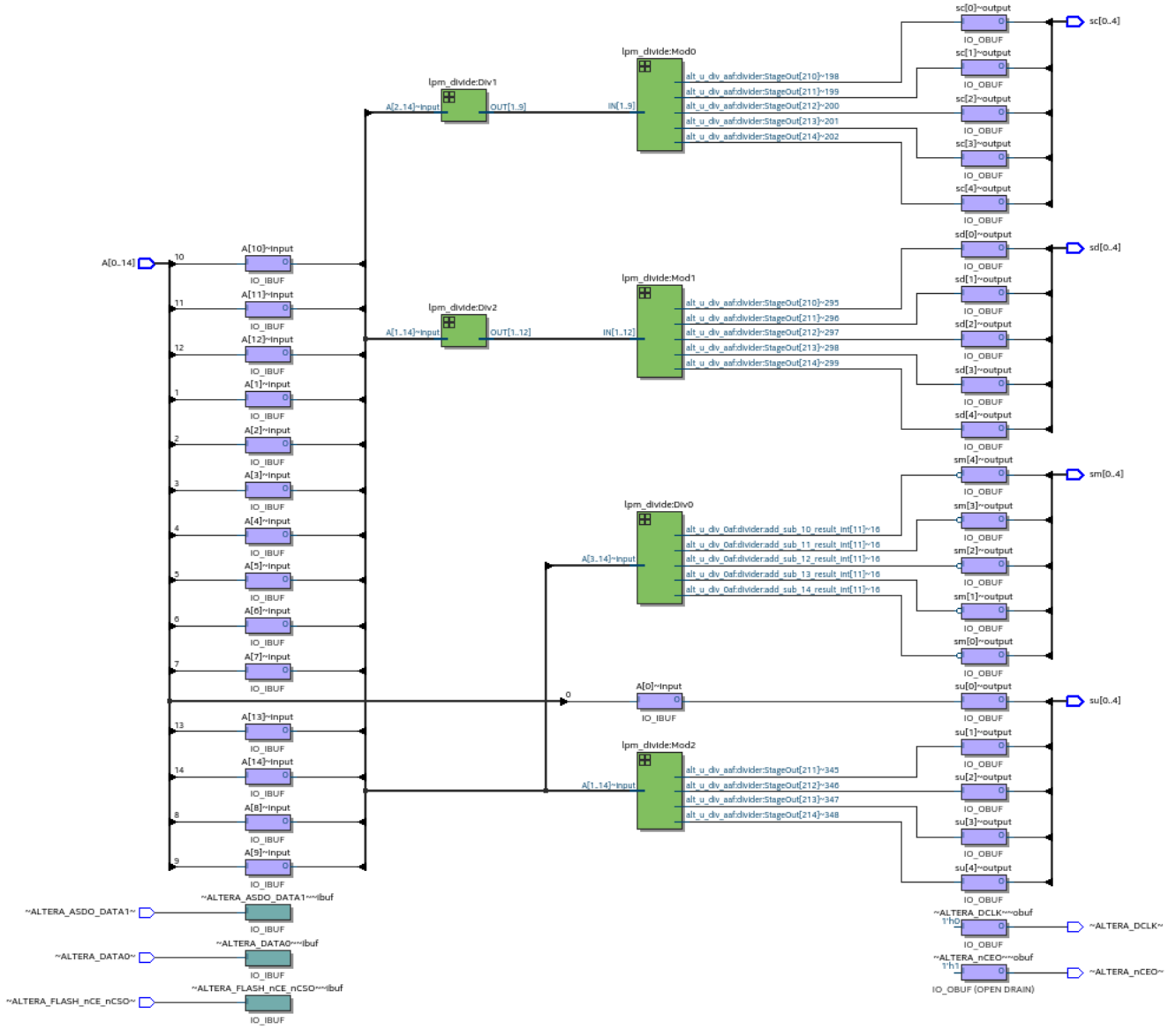


Figura AE4 7: Technology Map Viewer (Post Mapping)
Fonte: Elaborada pelo autor

2. Conclusão

Observando os resultados obtidos , é crível que eles são aceitáveis com os recursos aprendidos até o momento. Porém é bem possível diversas possíveis otimizações como mencionando anteriormente , alterando tanto o tempo de propagação quanto o número de elementos. Ou também é possível que diminuir o tempo de propagação leve um uso maior de elementos lógicos.