

**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

Conversor de binário para BCD

Dispositivos Lógicos Programáveis I

Sumário

1	Orientações	3
1.1	Objetivo	3
1.2	Passo 1 - Escrever o código do conversor:	3
1.3	Passo 2 - Análise de tempo de propagação:	3
1.4	Passo 3 - Otimizações de tempo e propagação:	5
2	Desenvolvimento	5
2.1	Passo 1 - Código VHDL do conversor:	5
2.2	Passo 2 - Análise do tempo de propagação:	7
3	Referências bibliográficas	12

1 Orientações

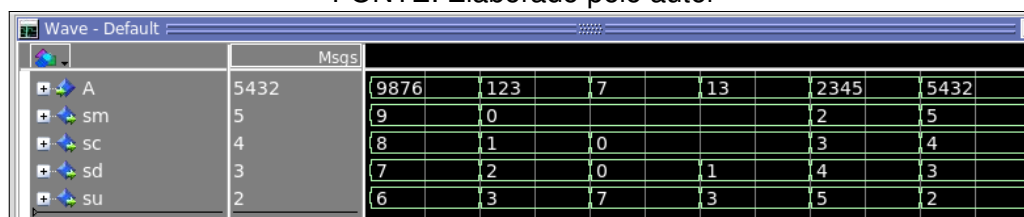
1.1 Objetivo

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.

1.2 Passo 1 - Escrever o código do conversor:

- 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.

FONTE: Elaborado pelo autor



Input A	sm	sc	sd	su
5432	5	4	3	2

Figura 1: Exemplo de simulação funcional de 0 a 9999

1.3 Passo 2 - Análise de tempo de propagação:

- 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.

FONTE: Elaborado pelo autor

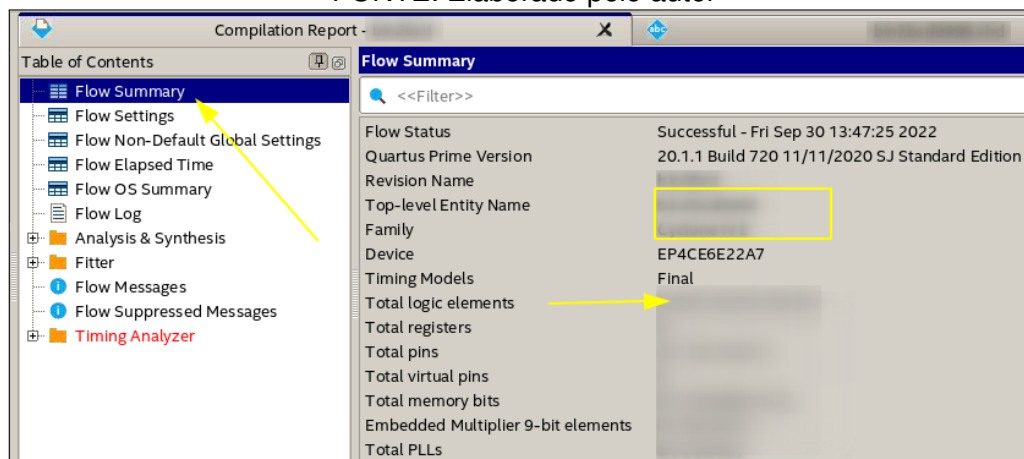


Figura 2: Obtendo o número de elementos no "Flow Summary"

FONTE: Elaborado pelo autor

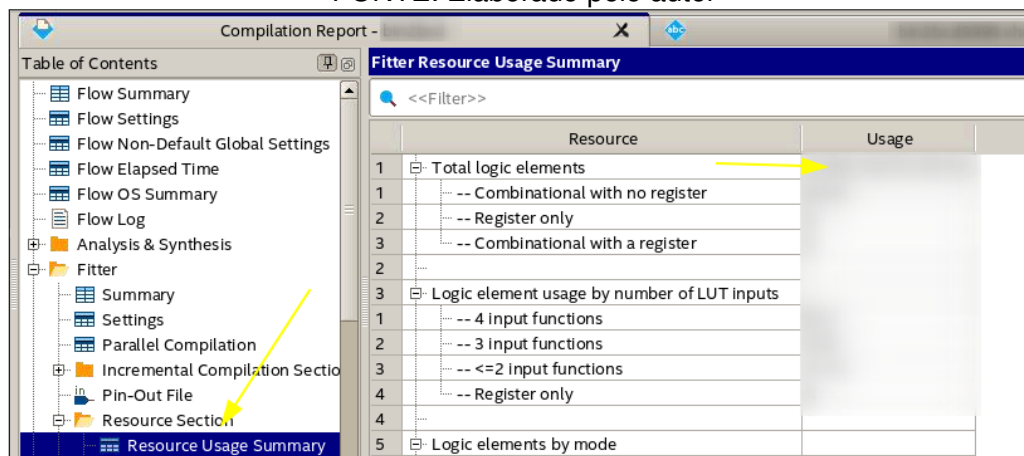


Figura 3: Obtendo o número de elementos no "Resource Usage Summary"

- 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.

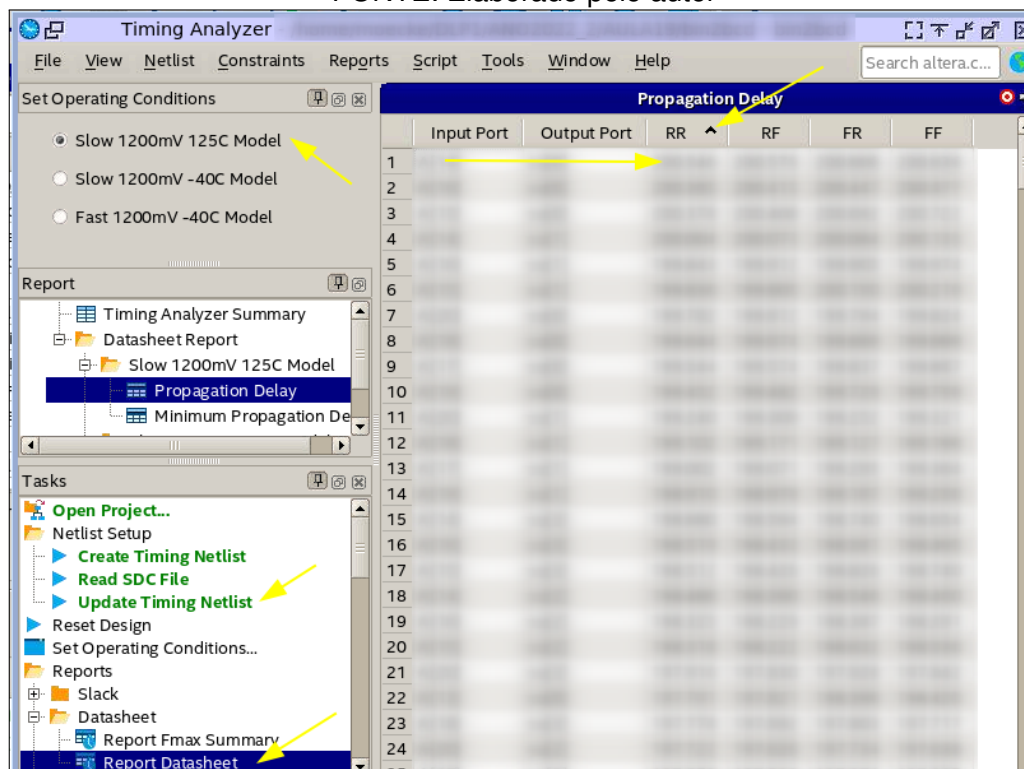


Figura 4: AE4(d) - Exemplo de tempo máximo de propagação

1.4 Passo 3 - Otimizações de tempo e 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 tbbin2bcd.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 Desenvolvimento

2.1 Passo 1 - Código VHDL do conversor:

Inicialmente fiz a montagem e plot do arquivo VHDL em um projeto de testes até que obtive o seguinte código:

```

1  -- Aluno: Arthur Cadore M. Barcella
2  -- DLP1: Eng telecom
3  -----
4  library ieee;
5  use ieee.std_logic_1164.all;
6  use ieee.numeric_std.all;
7
8  entity bin_to_bcd_lab is
9      port (
10         A : in std_logic_vector(14 downto 0); -- Entrada de 15 bits em binário

```

```

11      sm : out std_logic_vector(4 downto 0); -- Milhar - Entrada mais significativo
12      sc : out std_logic_vector(4 downto 0); -- Centena
13      sd : out std_logic_vector(4 downto 0); -- Dezena
14      su : out std_logic_vector(4 downto 0) -- Unidade - Entrada menos significativo
15  );
16 end entity bin_to_bcd_lab;
17
18 architecture ae4 of bin_to_bcd_lab is
19     signal A_uns : unsigned(14 downto 0); -- Sinal para armazenar o valor de entrada.
20     signal slice_mil : unsigned(14 downto 0); -- Milhar em BCD
21     signal slice_cem : unsigned(14 downto 0); -- Centena em BCD
22     signal slice_dez : unsigned(14 downto 0); -- Dezena em BCD
23     signal slice_uni : unsigned(14 downto 0); -- Unidade em BCD
24 begin
25
26     -- Converter entrada binária para número inteiro sem sinal
27     A_uns <= unsigned(A);
28
29     -- Converter binário para BCD
30     process(A_uns)
31     begin
32         slice_mil <= A_uns / 1000; -- Extrair milhar do número BCD
33         slice_cem <= (A_uns mod 1000) / 100; -- Extrair centena do número BCD
34         slice_dez <= (A_uns mod 100) / 10; -- Extrair dezena do número BCD
35         slice_uni <= A_uns mod 10; -- Extrair unidade do número BCD
36     end process;
37
38     -- Saída dos dígitos BCD
39     -- Converter para vetor lógico
40     sm <= std_logic_vector(slice_mil(4 downto 0));
41     sc <= std_logic_vector(slice_cem(4 downto 0));
42     sd <= std_logic_vector(slice_dez(4 downto 0));
43     su <= std_logic_vector(slice_uni(4 downto 0));
44 end architecture;

```

A ideia do código é receber o valor de entrada na porta "A" e em seguida retirar o valor da casa do milhar do número, da centena, da dezena e por fim da unidade. Esse código foi desenvolvido através de uma das explicações do professor sobre este tipo de operação, onde coletamos o valor de cada casa decimal separadamente. Em seguida, para saída dos valores obtidos, faço a conversão para STD-LOG antes de encaminhá-los à saída.

Para verificar o funcionamento do código apresentado acima, utilizei a simulação funcional do ModelSim inserindo diferentes valores e observando sua saída, para assim validar a saída correta dos valores:

FONTE: Elaborado pelo autor

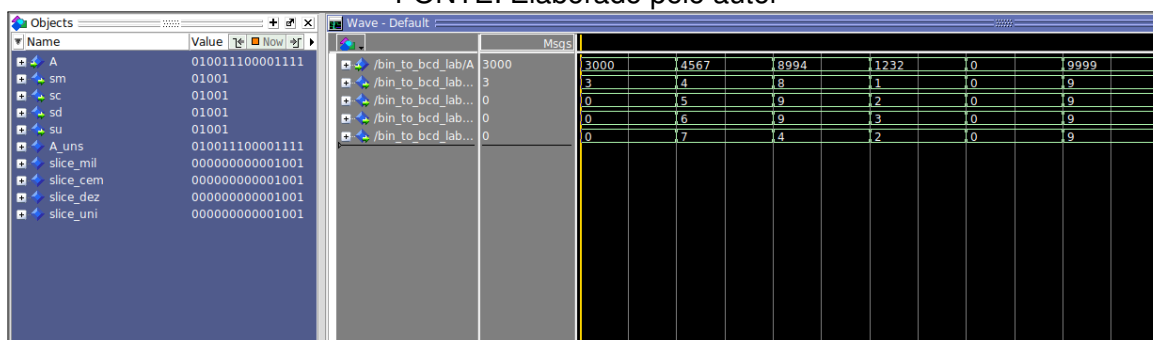


Figura 5: Simulação do código desenvolvido para validar a saída de A

No caso acima, foram inseridos seis diferentes valores na simulação, sendo quatro desses valores aleatórios e também outros dois valores utilizados para testar os limites da faixa de valores (0000) e

(9999). Note que todos os valores tiveram a saída correta, conforme esperado pelo funcionamento do conversor.

2.2 Passo 2 - Análise do tempo de propagação:

Para melhorar o tempo de propagação do circuito, primeiramente fiz uma compilação sem nenhuma alteração nos parâmetros de fitter do circuito, apenas para determinar qual o valor de propagação máximo que o circuito tinha até o momento:

FONTE: Elaborado pelo autor

	Input Port	Output Port	RR	RF	FR	FF
10	A[14]	sd[1]	76.023	75.782	76.157	75.916
11	A[9]	sd[0]	75.721	75.644	75.935	75.858
12	A[8]	sd[0]	74.107	74.030	74.378	74.301
13	A[9]	sd[1]	73.748	73.507	73.962	73.721
14	A[10]	sd[2]	72.174	72.072	72.350	72.248
15	A[8]	sd[1]	72.134	71.893	72.405	72.164
16	A[13]	sd[2]	71.586	71.484	72.011	71.909
17	A[12]	sd[2]	71.343	71.241	71.479	71.377
18	A[11]	sd[2]	71.274	71.172	71.658	71.556
19	A[14]	sd[2]	71.224	71.122	71.358	71.256
20	A[7]	sd[0]	70.436	70.359	70.654	70.577
21	A[9]	sd[2]	68.949	68.847	69.163	69.061
22	A[10]	sd[3]	68.466	68.619	68.642	68.795
23	A[7]	sd[1]	68.463	68.222	68.681	68.440
24	A[13]	sd[3]	67.878	68.031	68.303	68.456
25	A[12]	sd[3]	67.635	67.788	67.771	67.924
26	A[11]	sd[3]	67.566	67.719	67.950	68.103
27	A[14]	sd[3]	67.516	67.669	67.650	67.803
28	A[8]	sd[2]	67.335	67.233	67.606	67.504
29	A[6]	sd[0]	66.356	66.279	66.410	66.333
30	A[10]	sd[4]	65.564	65.910	65.740	66.086
31	A[9]	sd[3]	65.241	65.394	65.455	65.608
32	A[13]	sd[4]	64.976	65.322	65.401	65.747
33	A[12]	sd[4]	64.733	65.079	64.869	65.215
34	A[11]	sd[4]	64.664	65.010	65.048	65.394
35	A[14]	sd[4]	64.614	64.960	64.748	65.094
36	A[6]	sd[1]	64.383	64.142	64.437	64.196
37	A[12]	sc[0]	63.810	64.080	64.036	64.306
38	A[7]	sd[2]	63.664	63.562	63.882	63.780
39	A[8]	sd[3]	63.627	63.780	63.898	64.051
40	A[9]	sc[0]	63.612	63.882	64.096	64.366
41	A[8]	sc[0]	63.555	63.825	63.734	64.004
42	A[11]	sc[0]	63.479	63.749	63.933	64.203
43	A[14]	sc[0]	63.372	63.642	63.582	63.852
44	A[10]	sc[0]	63.333	63.603	63.501	63.771
45	A[13]	sc[0]	63.245	63.515	63.731	64.001

Figura 6: Tempo de propagação máximo do circuito (sem alterações)

Ao verificar que o tempo de propagação estava próximo de 76us (lembrando que o tempo registrado é o máximo em qualquer caso no circuito), realizei alterações em alguns parâmetros do programa para tentar diminuir o tempo total de propagação.

Além disso, ou parâmetro que registrei a fim de verificar possíveis melhoras foi a quantidade de elementos lógicos e pinos de I/O, abaixo estão exibidos estes valores para o programa sem alterações (conforme descrito acima).

FONTE: Elaborado pelo autor

Set Operating Conditions						
<input checked="" type="radio"/> Slow 1200mV 125C Model <input type="radio"/> Slow 1200mV -40C Model <input type="radio"/> Fast 1200mV -40C Model						
Report						
Timing Analyzer Summary Advanced I/O Timing Datasheet Report Slow 1200mV 125C Model Propagation Delay Minimum Propagation Delay Slow 1200mV -40C Model Fast 1200mV -40C Model						
Tasks						
Set Operating Conditions... Reports Slack Report Setup Summary Report Hold Summary Report Recovery Summary Report Removal Summary Report Minimum Pulse Width Report Max Skew Summary Report Net Delay Summary Datasheet Report Fmax Summary Report Datasheet Device Specific Report TCCS Report RSKM Report DDR Report Metastability Summary Diagnostic Report Clocks						
	Input Port	Output Port	RR ^	RF	FR	FF
1	A[10]	sd[0]	73.269	73.232	73.285	73.248
2	A[11]	sd[0]	72.891	72.854	73.179	73.142
3	A[12]	sd[0]	72.816	72.779	72.786	72.749
4	A[13]	sd[0]	72.671	72.634	72.924	72.887
5	A[14]	sd[0]	72.656	72.619	72.643	72.606
6	A[10]	sd[1]	70.528	70.709	70.544	70.725
7	A[9]	sd[0]	70.466	70.429	70.793	70.756
8	A[11]	sd[1]	70.150	70.331	70.438	70.619
9	A[12]	sd[1]	70.075	70.256	70.045	70.226
10	A[13]	sd[1]	69.930	70.111	70.183	70.364
11	A[14]	sd[1]	69.915	70.096	69.902	70.083
12	A[8]	sd[0]	69.057	69.020	69.227	69.190
13	A[10]	sd[2]	67.751	67.664	67.767	67.680
14	A[9]	sd[1]	67.725	67.906	68.052	68.233
15	A[11]	sd[2]	67.373	67.286	67.661	67.574
16	A[12]	sd[2]	67.298	67.211	67.268	67.181
17	A[13]	sd[2]	67.153	67.066	67.406	67.319
18	A[14]	sd[2]	67.138	67.051	67.125	67.038
19	A[8]	sd[1]	66.316	66.497	66.486	66.667
20	A[10]	sd[3]	65.946	65.983	65.962	65.999
21	A[11]	sd[3]	65.568	65.605	65.856	65.893
22	A[12]	sd[3]	65.493	65.530	65.463	65.500
23	A[13]	sd[3]	65.348	65.385	65.601	65.638
24	A[14]	sd[3]	65.333	65.370	65.320	65.357
25	A[7]	sd[0]	65.189	65.152	65.849	65.812
26	A[9]	sd[2]	64.948	64.861	65.275	65.188
27	A[8]	sd[2]	63.539	63.452	63.709	63.622
28	A[9]	sd[3]	63.143	63.180	63.470	63.507
29	A[7]	sd[1]	62.448	62.629	63.108	63.289
30	A[6]	sd[0]	62.360	62.323	62.732	62.695
31	A[8]	sd[3]	61.734	61.771	61.904	61.941
32	A[10]	sd[4]	61.481	61.536	61.497	61.552
33	A[11]	sd[4]	61.103	61.158	61.391	61.446
34	A[12]	sd[4]	61.028	61.083	60.998	61.053
35	A[13]	sd[4]	60.883	60.938	61.136	61.191
36	A[14]	sd[4]	60.868	60.923	60.855	60.910

Figura 7: Simulação do código desenvolvido para validar a saída de A

A primeira alteração realizada foi o tipo de compilação utilizada pelo programa para gerar o circuito a partir do código VHDL, como nessa implementação o tempo de propagação e o número de elementos lógicos é essencial, alterei as configurações do programa para tentar uma maior performance do circuito resultante durante a compilação, independentemente do tamanho do circuito.

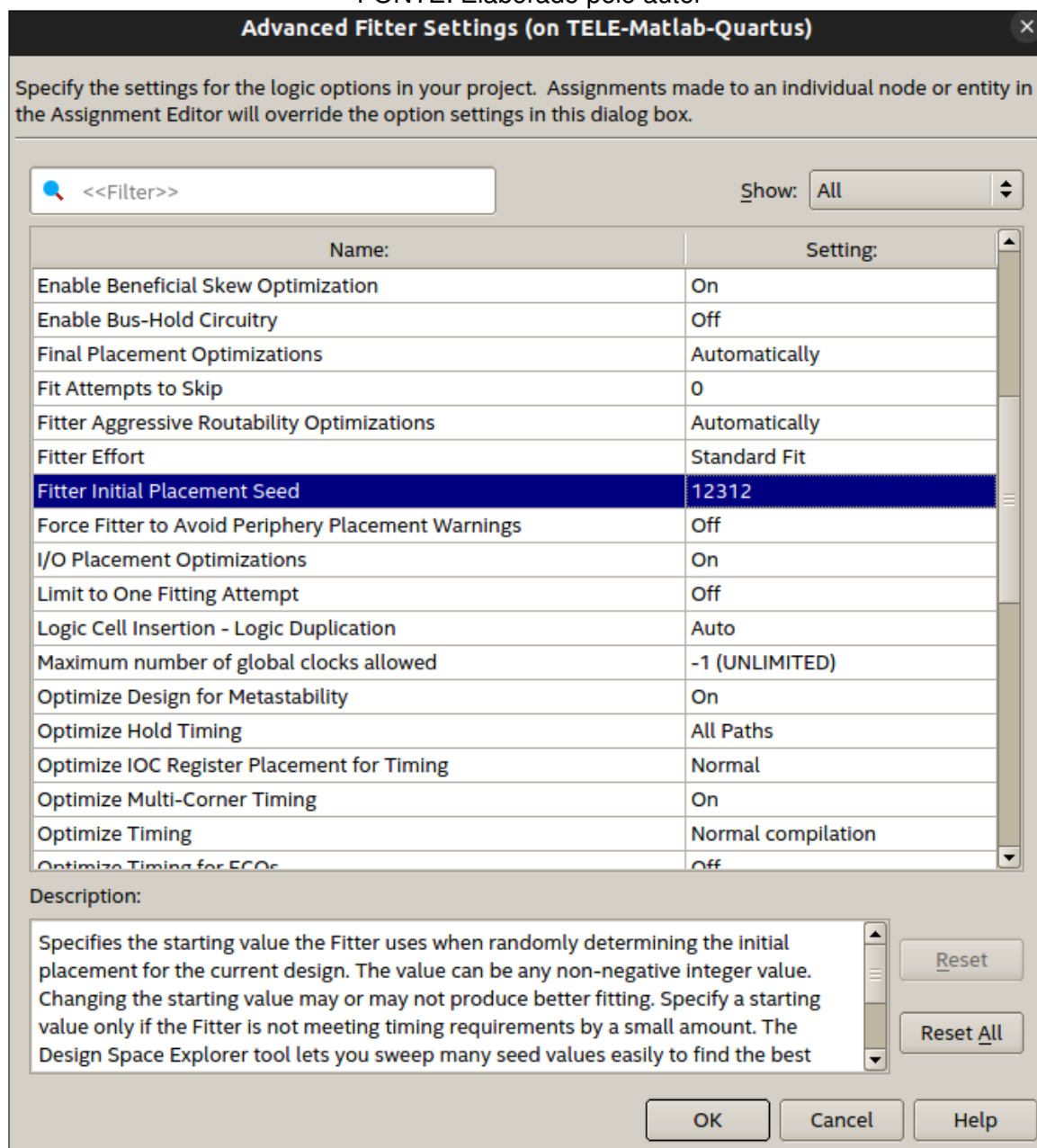


Figura 8: Tempo de propagação máximo do circuito (alteraa seed e método de compilação)

EM seguida, verifiquei que as alterações feitas não tiveram mudança significativa no valor de propagação, portanto, alterei também a seed utilizada no processo de filter do quartus), com isso tive um ganho aproximado de 3us entre este caso e o primeiro caso registrado.

Dessa forma, a tabela de propagação ficou da seguinte maneira:

FONTE: Elaborado pelo autor

Set Operating Conditions							
<input checked="" type="radio"/> Slow 1200mV 125C Model <input type="radio"/> Slow 1200mV -40C Model <input type="radio"/> Fast 1200mV -40C Model		Input Port	Output Port	RR ^	RF	FR	FF
		1 A[10]	sd[0]	73.269	73.232	73.285	73.248
		2 A[11]	sd[0]	72.891	72.854	73.179	73.142
		3 A[12]	sd[0]	72.816	72.779	72.786	72.749
		4 A[13]	sd[0]	72.671	72.634	72.924	72.887
		5 A[14]	sd[0]	72.656	72.619	72.643	72.606
		6 A[10]	sd[1]	70.528	70.709	70.544	70.725
		7 A[9]	sd[0]	70.466	70.429	70.793	70.756
		8 A[11]	sd[1]	70.150	70.331	70.438	70.619
		9 A[12]	sd[1]	70.075	70.256	70.045	70.226
		10 A[13]	sd[1]	69.930	70.111	70.183	70.364
		11 A[14]	sd[1]	69.915	70.096	69.902	70.083
		12 A[8]	sd[0]	69.057	69.020	69.227	69.190
		13 A[10]	sd[2]	67.751	67.664	67.767	67.680
		14 A[9]	sd[1]	67.725	67.906	68.052	68.233
		15 A[11]	sd[2]	67.373	67.286	67.661	67.574
		16 A[12]	sd[2]	67.298	67.211	67.268	67.181
		17 A[13]	sd[2]	67.153	67.066	67.406	67.319
		18 A[14]	sd[2]	67.138	67.051	67.125	67.038
		19 A[8]	sd[1]	66.316	66.497	66.486	66.667
		20 A[10]	sd[3]	65.946	65.983	65.962	65.999
		21 A[11]	sd[3]	65.568	65.605	65.856	65.893
		22 A[12]	sd[3]	65.493	65.530	65.463	65.500
		23 A[13]	sd[3]	65.348	65.385	65.601	65.638
		24 A[14]	sd[3]	65.333	65.370	65.320	65.357
		25 A[7]	sd[0]	65.189	65.152	65.849	65.812
		26 A[9]	sd[2]	64.948	64.861	65.275	65.188
		27 A[8]	sd[2]	63.539	63.452	63.709	63.622
		28 A[9]	sd[3]	63.143	63.180	63.470	63.507
		29 A[7]	sd[1]	62.448	62.629	63.108	63.289
		30 A[6]	sd[0]	62.360	62.323	62.732	62.695
		31 A[8]	sd[3]	61.734	61.771	61.904	61.941
		32 A[10]	sd[4]	61.481	61.536	61.497	61.552
		33 A[11]	sd[4]	61.103	61.158	61.391	61.446
		34 A[12]	sd[4]	61.028	61.083	60.998	61.053
		35 A[13]	sd[4]	60.883	60.938	61.136	61.191
		36 A[14]	sd[4]	60.868	60.923	60.855	60.910

Figura 9: Seed de configuração do fitter

A melhora apresentada acima foi significativa para ser registrada no entanto, também apliquei um script demonstrado pelo professor á algumas aulas para tentar diminuir ainda mais o tempo de propagação, para isso configurei o arquivo "SDC1.sdc"com o seguinte conteúdo:

```
1 set_max_delay -from [get_ports *] -to [get_ports *] 40
```

O script acima faz com que a propagação de um sinal na entrada e saída do circuito não passe de um valor específico de tempo, no exemplo acima está registrando 40us como valor limite.

Entretanto conforme apresentado abaixo, por mais que o valor utilizado para "corte"seja de 40us, devido ao valor em muitos casos ser impraticavel na compilação, o tempo de propagação máximo não fica dentro do especificado no script.

FONTE: Elaborado pelo autor

	Input Port	Output Port	RR ^	RF	FR	FF
1	A[10]	sd[0]	66.234	66.371	66.009	66.146
2	A[11]	sd[0]	66.064	66.201	66.106	66.243
3	A[12]	sd[0]	65.815	65.952	65.625	65.762
4	A[13]	sd[0]	65.744	65.881	65.815	65.952
5	A[14]	sd[0]	65.712	65.849	65.531	65.668
6	A[10]	sd[1]	64.279	64.357	64.054	64.132
7	A[11]	sd[1]	64.109	64.187	64.151	64.229
8	A[12]	sd[1]	63.860	63.938	63.670	63.748
9	A[13]	sd[1]	63.789	63.867	63.860	63.938
10	A[14]	sd[1]	63.757	63.835	63.576	63.654
11	A[9]	sd[0]	63.613	63.750	64.204	64.341
12	A[9]	sd[1]	61.658	61.736	62.249	62.327
13	A[10]	sd[2]	61.484	61.795	61.259	61.570
14	A[11]	sd[2]	61.314	61.625	61.356	61.667
15	A[8]	sd[0]	61.313	61.450	60.835	60.972
16	A[12]	sd[2]	61.065	61.376	60.875	61.186
17	A[13]	sd[2]	60.994	61.305	61.065	61.376
18	A[14]	sd[2]	60.962	61.273	60.781	61.092
19	A[8]	sd[1]	59.358	59.436	58.880	58.958
20	A[9]	sd[2]	58.863	59.174	59.454	59.765
21	A[10]	sd[3]	58.807	58.872	58.582	58.647
22	A[11]	sd[3]	58.637	58.702	58.679	58.744
23	A[12]	sd[3]	58.388	58.453	58.198	58.263
24	A[13]	sd[3]	58.317	58.382	58.388	58.453
25	A[14]	sd[3]	58.285	58.350	58.104	58.169
26	A[7]	sd[0]	57.374	57.511	57.821	57.958
27	A[8]	sd[2]	56.563	56.874	56.085	56.396
28	A[9]	sd[3]	56.186	56.251	56.777	56.842
29	A[10]	sd[4]	55.792	55.839	55.567	55.614
30	A[11]	sd[4]	55.622	55.669	55.664	55.711
31	A[7]	sd[1]	55.419	55.497	55.866	55.944
32	A[12]	sd[4]	55.373	55.420	55.183	55.230
33	A[13]	sd[4]	55.302	55.349	55.373	55.420
34	A[14]	sd[4]	55.270	55.317	55.089	55.136
35	A[6]	sd[0]	55.112	55.249	54.840	54.977
36	A[8]	sd[3]	53.886	53.951	53.408	53.473

Figura 10: Tempo de propagação máximo do circuito (Alterada seed e tipo de compilação)

Por mais que o valor tenha ficado próximo de 67us, note que a configuração deu um ganho de aproximadamente 10us em relação ao caso anterior, devido as multiplas tentativas do quartus para aplicar o circuito.

Lembrando que para atingir este valor, iniciei os valores do script em 70us e desci 5 em 5 us até atingir 40us e verificar que não haviam mais melhoras significativas após diminuição do tempo.

A configuração do script de delay máximo também serviu para diminuir a quantidade de elementos lógicos do circuito, conforme a imagem abaixo, onde passamos de 1325 para 1280 elementos, diminuindo em 45 elementos o circuito:

Fonte: Elaborado pelo autor

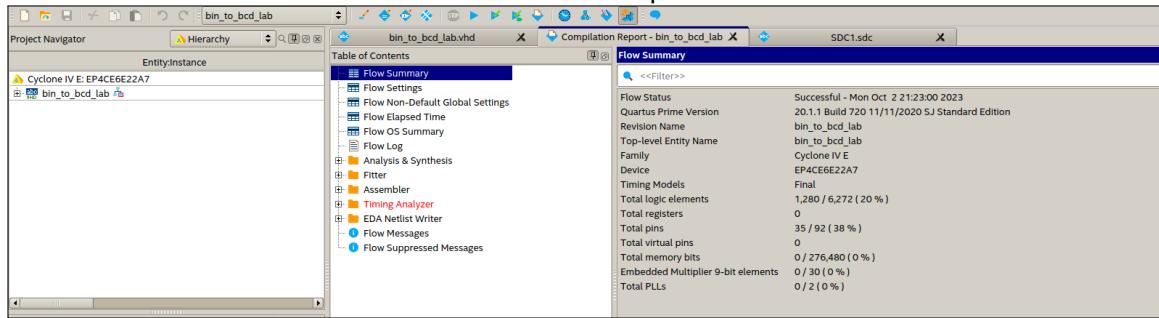


Figura 11: Tempo de propagação máximo do circuito (tempo atual)

Também verifiquei a imagem do diagrama RTL após a compilação da segunda versão do código (com script SDC habilitado), onde é possível notar a semântica do código VHDL:

Fonte: Elaborado pelo autor

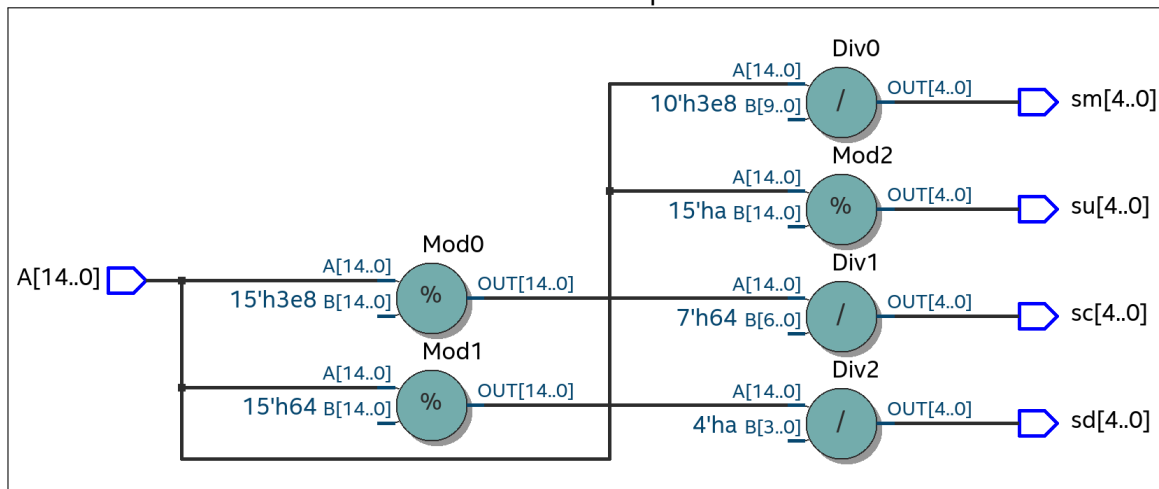


Figura 12: Diagrama RTL do circuito

3 Referências bibliográficas

Orientações do laboratório
Simulação Funcional usando o ModelSim