

**INSTITUTO  
FEDERAL**  
Santa Catarina

---

Câmpus  
São José

# **Conhecendo os Dispositivos Lógicos Programáveis**

**Dispositivos Lógicos Programáveis I**

**Arthur Cadore Matuella Barcella**

10 de Agosto de 2023

# **Sumário**

<b>1</b>	<b>Orientações</b>	<b>3</b>
1.1	Objetivo . . . . .	3
1.2	Passo 1: . . . . .	3
1.3	Passo 2: . . . . .	3
1.4	Passo 3: . . . . .	3
<b>2</b>	<b>Desenvolvimento</b>	<b>4</b>
2.1	Passo 1: . . . . .	4
2.2	Passo 2: . . . . .	10
2.3	Passo 3: . . . . .	18
2.4	Passo 3 - Otimizações: . . . . .	24
<b>3</b>	<b>Referências bibliográficas</b>	<b>30</b>

# 1 Orientações

## 1.1 Objetivo

- Conhecer o Quartus Prime e as características dos dispositivos lógicos programáveis
- Analisar os tempos de propagação em um circuito combinacional
- Alterar configurações do compilador
- Fazer a simulação funcional e temporal de um circuito combinacional.

## 1.2 Passo 1:

- Ao escolher a família de FPGAS, escolha inicialmente um dispositivo da família Max II. Anote o código desse dispositivo.
- Capture as telas solicitadas e depois utilize-as no relatório da atividade.
- Anote o tempo utilizado para cada uma das etapas do processo de compilação.
- Anote o número de elementos lógicos utilizados e o número de pinos utilizados, bem com o percentual em relação ao número total do dispositivo.
- Anote algum erro (Error) ou alertas (Warnings) que o Quartus II indicar no painel de mensagens [Messages]
- Ao final salve o projeto em um arquivo QAR (sugestão PJ1.QAR)

## 1.3 Passo 2:

- Repita a atividade descrita em Conhecendo os dispositivos lógicos programáveis - QUARTUS PRIME, trocando a família e dispositivo a ser usado na implementação.
- Escolha nesta vez um dispositivos da família Cyclone IV E ou Stratix II GX. Anote o código desse dispositivo.
- Observe as mudanças que ocorrem tanto no tipo de Elemento Lógico disponível, no Chip Planner, no Pin Planner, e no circuito dos pinos de I/O.
- Note que estes FPGAs também apresenta novos componentes, tais como: Memória, Multiplicadores, DSP, PLL, DLL, etc. Verifique se consegue encontrar-los no leiaute mostrado no Chip Planner, e documente aqueles que encontrar.
- Compare os resultados obtidos nos procedimentos do PASSO 1 e PASSO 2.

## 1.4 Passo 3:

- Realize o procedimento descrito em Medição de tempos de propagação em circuitos combinacionais - Quartus Prime Ao escolher a família de FPGAS, escolha um dispositivo FPGA da família Cyclone IV E. Anote o código desse dispositivo.
- Capture as telas mostradas no roteiro e depois utilize-as no relatório da atividade.
- Anote o máximo tempo de propagação entre entrada e saída.
- Anote o número de elementos lógicos utilizados e o número de pinos utilizados, bem com o percentual em relação ao número total do dispositivo.

- Experimente modificar as configurações do compilador, conforme mostrado em Configurando o compilador. Se desejar mude a semente inicial trocando o valor de [Seed: 1]
- Experimente inserir diferentes restrições de atraso máximo para o compilador, e analise o resultado obtido.]
- Anote algum erro (Error) ou alertas (Warnings) que o Quartus II indicar no painel de mensagens [Messages]
- Ao final salve o projeto em um arquivo QAR (sugestão PJ2.QAR)

## 2 Desenvolvimento

### 2.1 Passo 1:

Conforme as orientações dadas para o desenvolvimento da atividade, a primeira etapa era selecionar um equipamento da família MAX II.

Iniciei um novo projeto e adicionei o seguinte dispositivo (EPM240F100C4) para realizar a atividade:

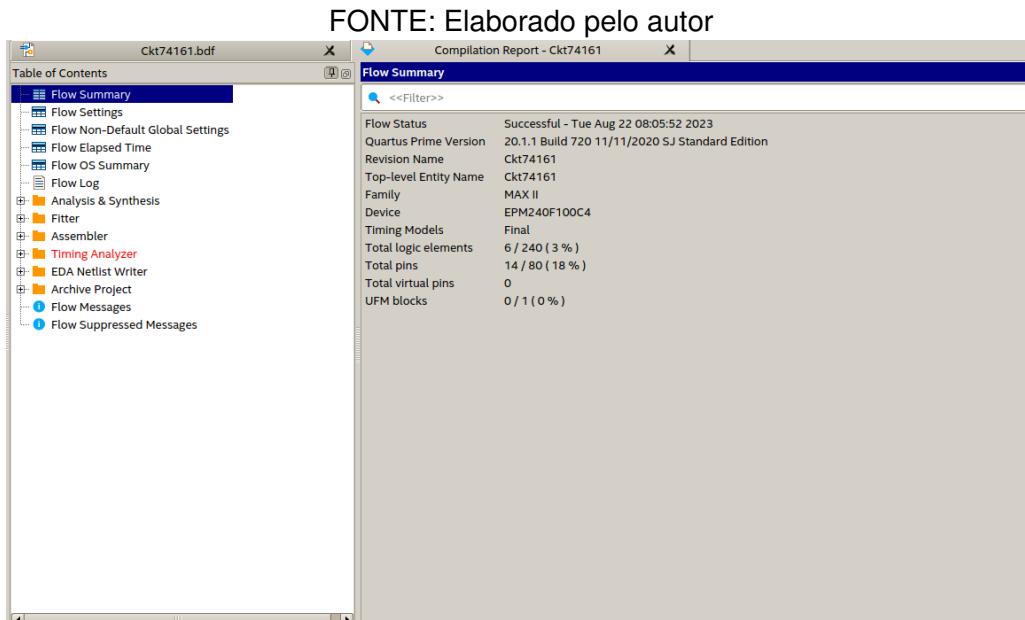


Figura 1: Configuração do projeto - Passo 1 AE2

Em seguida, adicionei o contador 74161 no diagrama de circuito, o objetivo da adição era compreender como o contador seria construído em hardware utilizando o dispositivo selecionado.

FONTE: Elaborado pelo autor

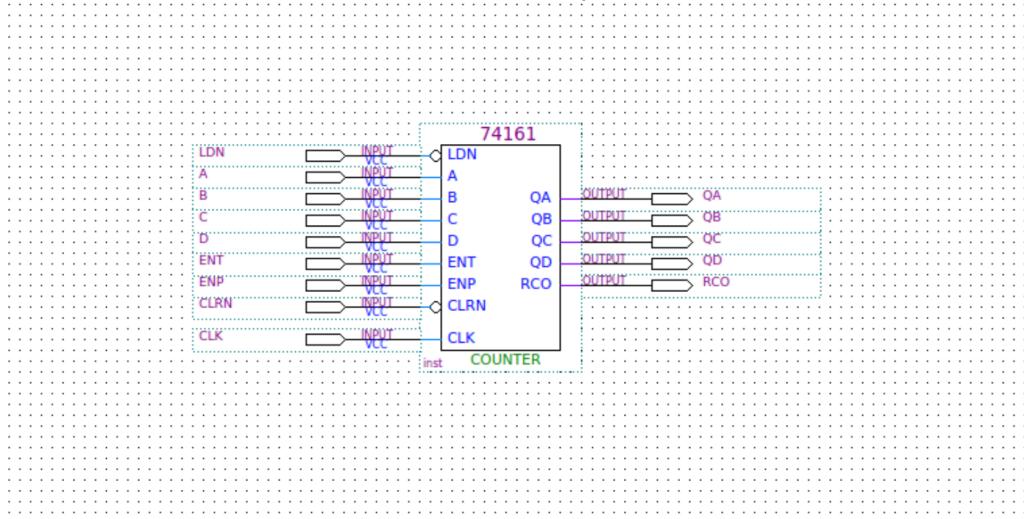


Figura 2: Adição do Contador 74161 ao diagrama de circuito

Em seguida, com salvei o diagrama de circuito/projeto como "Ckt74161.bdf", e iniciei a compilação do circuito especificado.

FONTE: Elaborado pelo autor

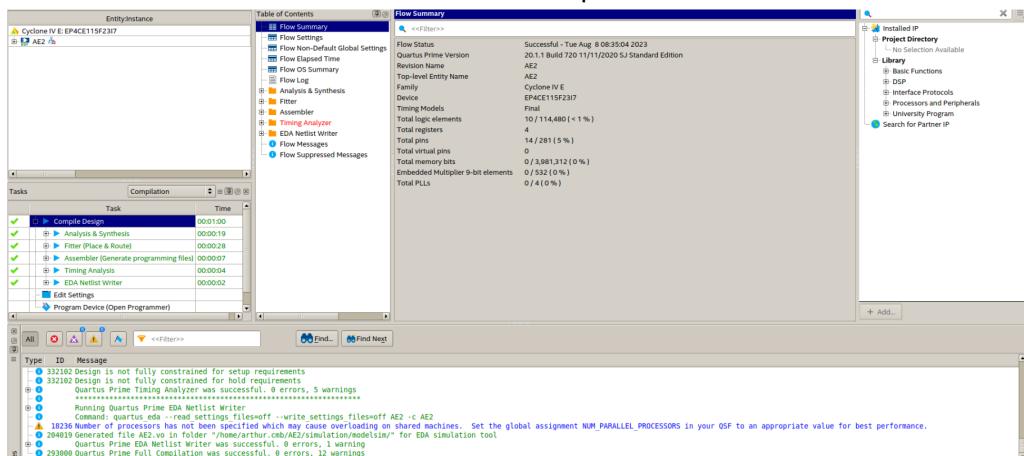


Figura 3: Compilando o diagrama do contador

A compilação é necessária para que o software faça a conexão automatizada dos componentes lógicos contidos dentro do dispositivo de maneira a criar o circuito desejado.

Uma vez com o circuito compilado, iniciei as observações partindo da ferramenta "Pin-Planner", para verificar os pinos automaticamente atribuidos ao dispositivo pelo software:

FONTE: Elaborado pelo autor

Top View - Wire Bond

MAX II - EPM240F100C4

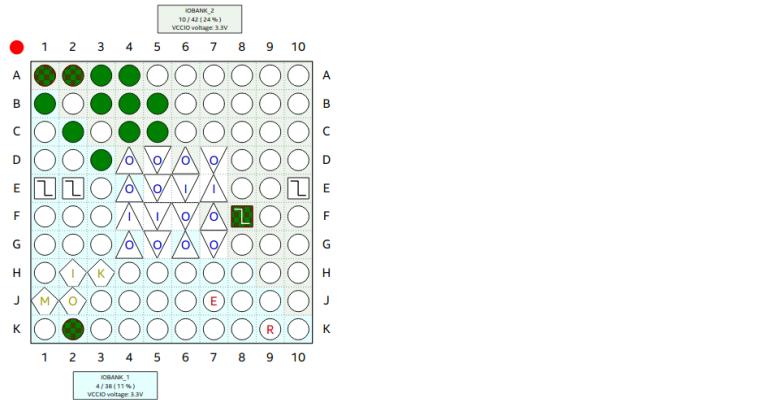


Figura 4: Visão do Pin-Planner com pinos já atribuidos

Note que os pinos apresentados em verde foram atribuídos automaticamente pelo quartus para serem utilizados como entrada e saída do dispositivo.

Abaixo está uma captura mostrando os pinos atribuídos para cada entrada do componente inserido no diagrama de circuito:

FONTE: Elaborado pelo autor

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength	strict Preservative
in A	Input	PIN_A1	2	PIN_A1	3.3-V LVTTL		16mA (default)	
in B	Input			PIN_B1	3.3-V L...efault		16mA (default)	
in C	Input			PIN_C2	3.3-V L...efault		16mA (default)	
in CLK	Input	PIN_F8	2	PIN_F8	3.3-V LVTTL		16mA (default)	
in CLRN	Input	PIN_K2	1	PIN_K2	3.3-V LVTTL		16mA (default)	
in D	Input			PIN_D3	3.3-V L...efault		16mA (default)	
in ENP	Input			PIN_B4	3.3-V L...efault		16mA (default)	
in ENT	Input	PIN_A2	2	PIN_A2	3.3-V LVTTL		16mA (default)	
in LDN	Input			PIN_C4	3.3-V L...efault		16mA (default)	
out QA	Output			PIN_B5	3.3-V L...efault		16mA (default)	
out QB	Output			PIN_B3	3.3-V L...efault		16mA (default)	
out QC	Output			PIN_A4	3.3-V L...efault		16mA (default)	

Figura 5: Tabela de pinos atribuídos no pin-planner

Em seguida, utilizei a ferramenta "Chip-planner"para observar como o contador foi adicionado ao dispositivo, ou seja, quais conexões foram feitas fisicamente no chip, além dos componentes de I/O utilizados.

FONTE: Elaborado pelo autor

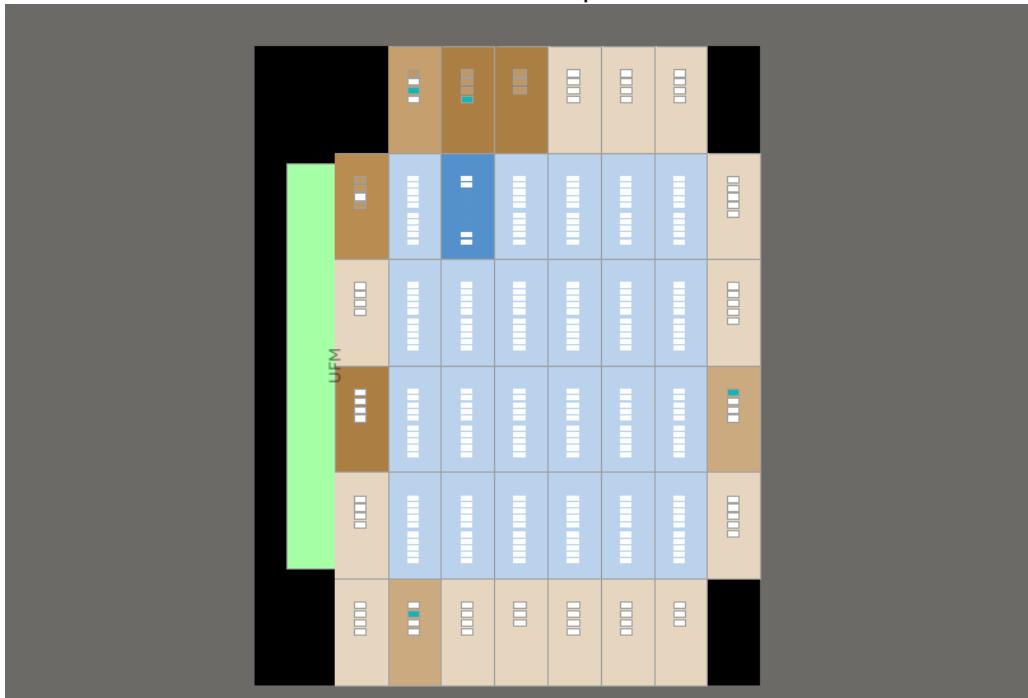


Figura 6: Visão geral da ferramenta "Chip-Planner" sobre o CI

Ao utilizar a ferramenta, é possível notar como o dispositivo será afetado pela montagem/compilação do circuito a nível de hardware.

A região azul ilustrada acima, apresenta o local onde os componentes lógicos que formam o contador foram alocados.

Ao aproximar da região destacada, fica evidente a composição do circuito, conforme ilustrado abaixo:

FONTE: Elaborado pelo autor

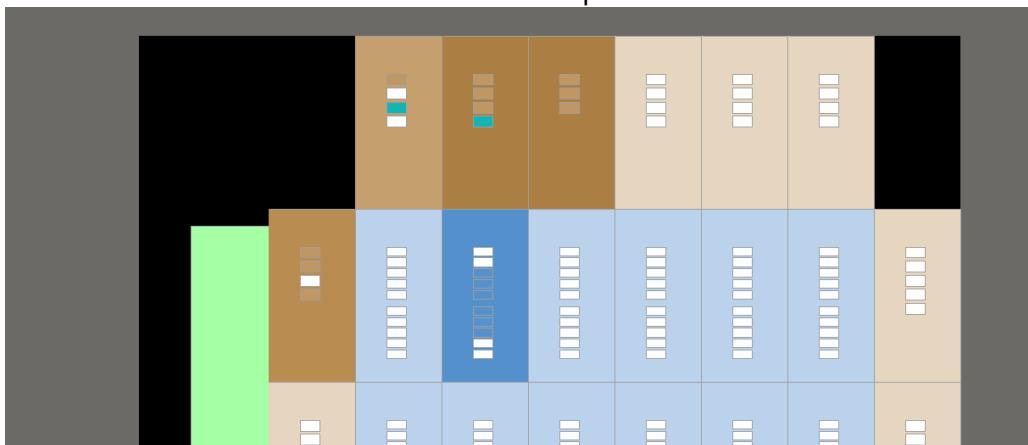


Figura 7: Visão aproximada do CI no "Chip-Planner"

Os retângulos preenchidos (em azul), representam as unidades lógicas alocadas para o circuito, enquanto que, os retângulos preenchidos (em marrom) representam as interfaces de I/O utilizadas para conexão do circuito com a área externa do dispositivo.

Ao selecionar um dos elementos lógicos e verificar suas propriedades, é possível identificar o circuito configurado pelo software. Abaixo está uma ilustração da unidade lógica utilizada no contador

(região em azul no chip-planner):

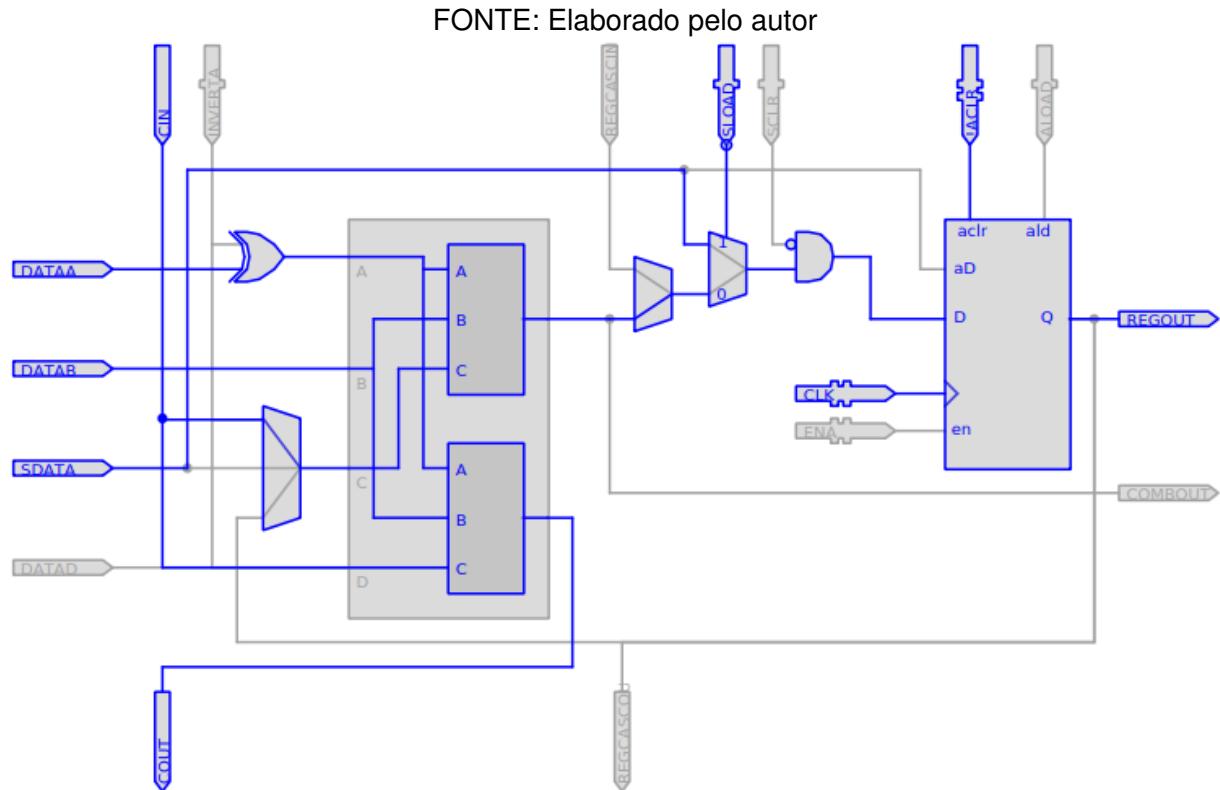


Figura 8: Circuito da unidade lógica configurada

Em seguida, repeti o procedimento para verificar o circuito de entrada e saída, abaixo está o diagrama detalhado do circuito

FONTE: Elaborado pelo autor

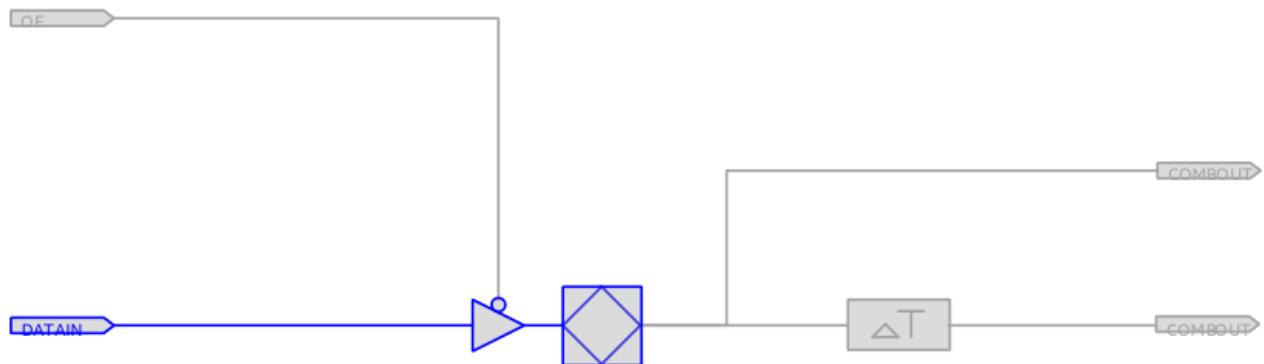


Figura 9: Circuito de I/O configurado

Em seguida, após analisar verifiquei o diagrama lógico do circuito (análogo ao circuito real), que utiliza portas lógicas e flip-flops, para isso, accesei a ferramenta "RTL-Viewer":

FONTE: Elaborado pelo autor

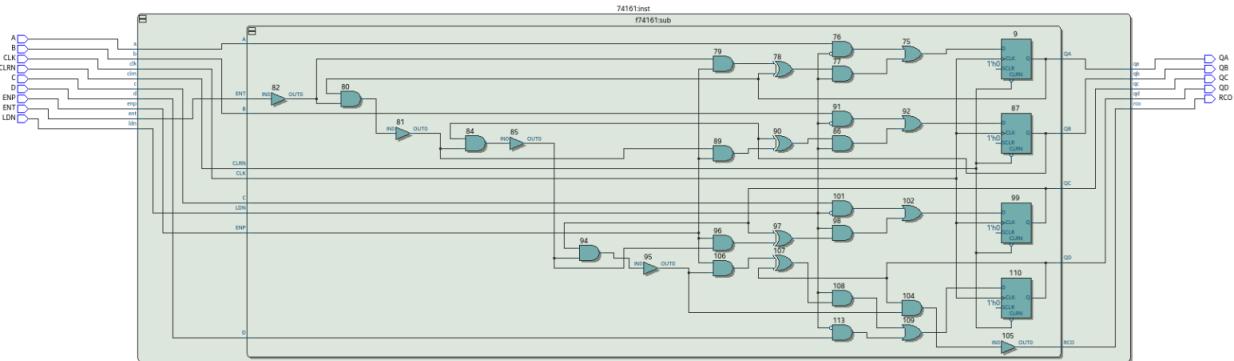


Figura 10: Visão RTL sobre o circuito configurado

Durante os testes, necessitei fazer a compilação do circuito apenas uma vez. Para o circuito apresentado o tempo de compilação total foi o seguinte:

FONTE: Elaborado pelo autor

	Task	Time
✓	Compile Design	00:01:00
✓	Analysis & Synthesis	00:00:19
✓	Fitter (Place & Route)	00:00:28
✓	Assembler (Generate programming files)	00:00:07
✓	Timing Analysis	00:00:04
✓	EDA Netlist Writer	00:00:02
	Edit Settings	
	Program Device (Open Programmer)	

Figura 11: Tempo total de compilação do circuito

Durante o tempo de compilação nenhum erro foi apresentado, apenas alguns warnings, o mais significativo está apresentado abaixo, referente a falta de especificação por parte do usuário da quantidade de processadores que o quartus poderá utilizar para realizar a compilação do projeto.

Sem essa informação, o quartus não sabe quantos processadores pode utilizar e por default, utiliza todos os dispositivos, o que pode gerar complicações para a máquina do usuário caso o projeto seja complexo ou denso.

FONTE: Elaborado pelo autor

```

Type ID Message
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
*****
Running Quartus Prime EDA Netlist Writer
Command: quartus.eda --read_settings_files=off --write_settings_files=off AE2 -c AE2
18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
204019 Generated file AE2.vo in folder "/home/arthur/cmb/AE2/simulation/modelsim/" for EDA simulation tool
Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
293000 Quartus Prime Full Compilation was successful. 0 errors, 12 warnings

```

Figura 12: Erros/Warnings exibidos em tempo de compilação

## 2.2 Passo 2:

Para a segunda atividade, criei um novo projeto, dessa vez utilizando a família CICLONE IV E, sendo o dispositivo EP4CE115, que contém muitos elementos lógicos e funcionalidades adicionais se comparado ao dispositivo anterior.

O objetivo de utilizar um dispositivo mais complexo e verificar quais circuitos/funcionalidades foram adicionados ao dispositivo:

FONTE: Elaborado pelo autor

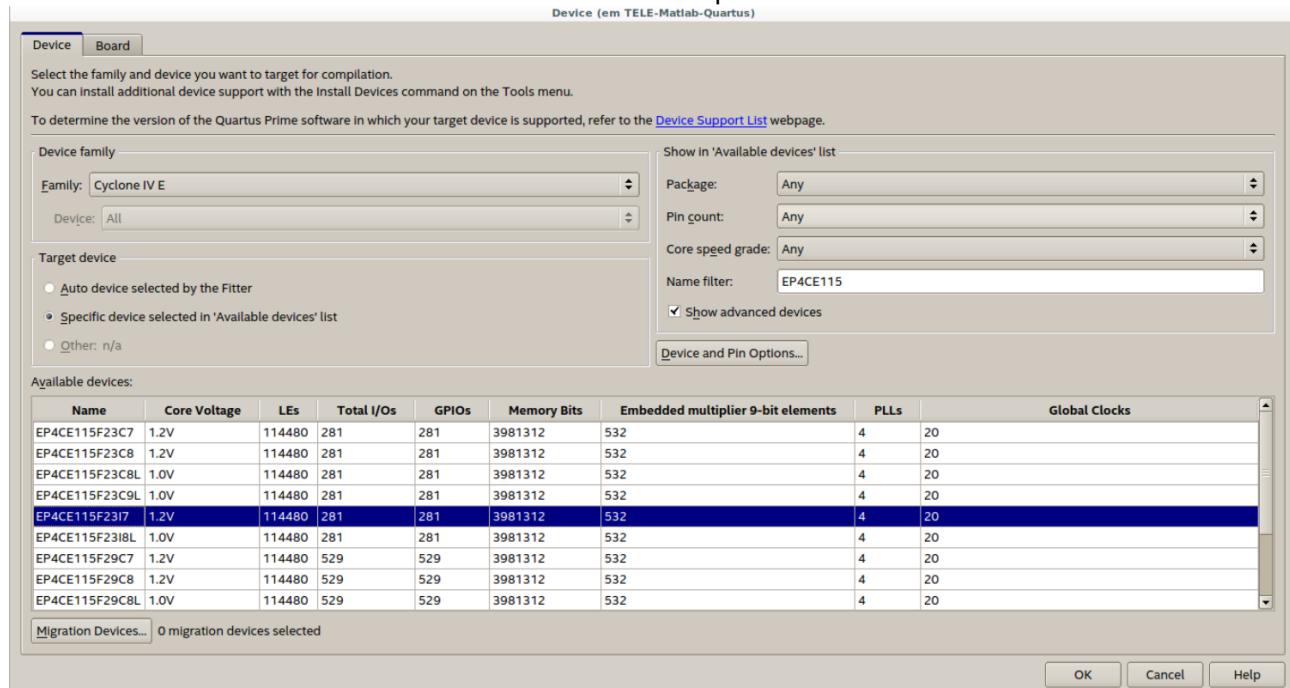


Figura 13: Criação do novo projeto com CICLONE IV

Uma vez com o novo dispositivo, adicionei novamente o contador 74161 no diagrama de circuito, o objetivo da adição era compreender quais as alterações na construção do contador a nível de hardware ao compararmos com o experimento anterior.

FONTE: Elaborado pelo autor

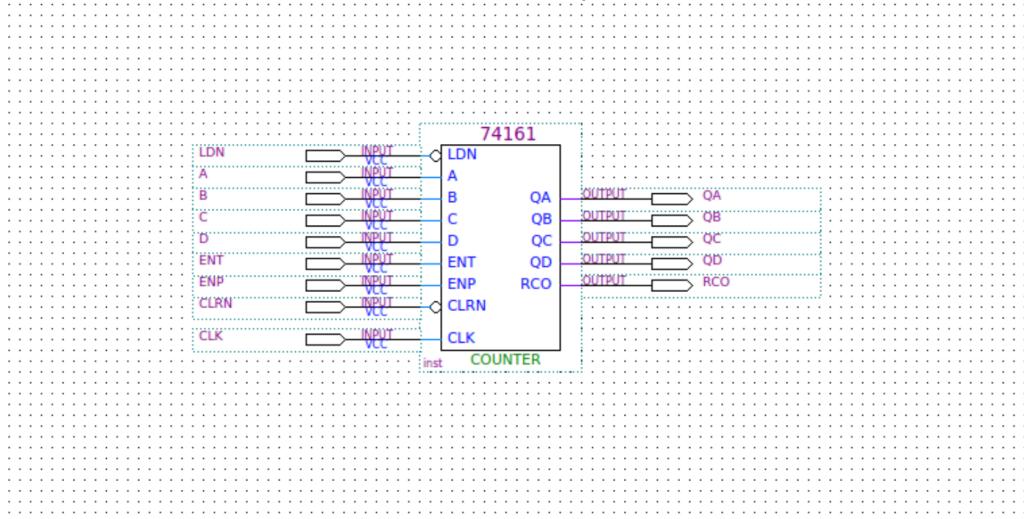


Figura 14: Adição do Contador 74161 ao diagrama de circuito

A primeira grande diferença aparente está na ferramenta "Pin-Planner", ao acessar a ferramenta utilizando o novo dispositivo nota-se uma enorme quantidade de pinos de I/O.

Diferentemente da quantidade de pinos reduzida do dispositivo usando anteriormente, este conta com 484 células nesta representação:

FONTE: Elaborado pelo autor

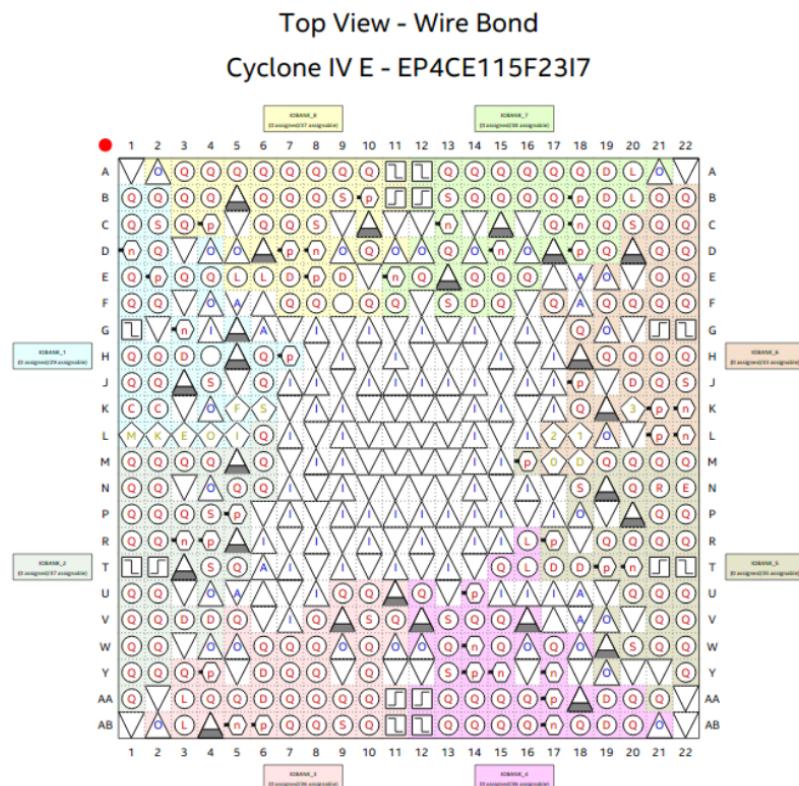


Figura 15: Visão do Pin-Planner sobre o CI

Em seguida, habilitei a visão sobre os pinos auto atribuidos a cada porta do contador inserido no circuito. Os pinos atribuidos estão representados em verde.

Note que o quartus tenta agrupa a maioria dos pinos utilizados em uma única região, a fim de simplificar a construção do circuito físico posteriormente. Além de diminuir a latência de I/O, deixando as entradas e saídas próximas.

FONTE: Elaborado pelo autor

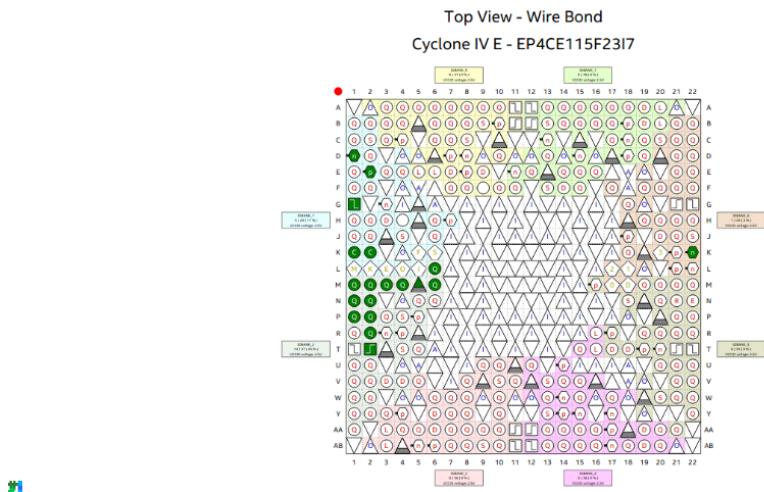


Figura 16: Visão do Pin-Planner sobre o CI - Pinos utilizados

Da mesma maneira representada anteriormente, abaixo está a tabela contendo a pinagem de I/O mapeada para o circuito, apresentada pela ferramenta "Pin-planner".

Note que devido as especificações do CI, as tensões de I/O sofreram alteração, devido a tecnologia do dispositivo.

FONTE: Elaborado pelo autor

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	er I/O Rule Resi	Differential Pair	strict Preservativ
A	Input				PIN_M3	3.2 V		8mA (default)		Pass		
B	Input				PIN_M5	2.5 V (default)		8mA (default)		Pass		
C	Input				PIN_M4	3.5 V (default)		8mA (default)		Pass		
CLK	Input				PIN_G1	2.5 V (default)		8mA (default)		Pass		
CLDN	Input				PIN_T2	2.5 V (default)		8mA (default)		Pass		
D	Input				PIN_H1	2.5 V (default)		8mA (default)		Pass		
ENP	Input				PIN_M2	2.5 V (default)		8mA (default)		Pass		
ENT	Input				PIN_N2	2.5 V (default)		8mA (default)		Pass		
LDN	Input				PIN_M1	2.5 V (default)		8mA (default)		Pass		
QA	Output				PIN_R2	2.5 V (default)	2 (default)	8mA (default)		Pass		
QB	Output				PIN_M6	2.5 V (default)	2 (default)	8mA (default)		Pass		
QC	Output				PIN_L6	2.5 V (default)	2 (default)	8mA (default)		Pass		

Figura 17: Tabela de pinos utilizados Pin-Planner

Em seguida, como no exercício anterior verifiquei a representação do dispositivo físico, através da ferramenta "Chip-planner".

Note que para este segundo dispositivo, a quantidade de elementos lógicos é muito maior, tornando a região azul do chip muito maior e com muitos segmentos que no caso anterior.

FONTE: Elaborado pelo autor

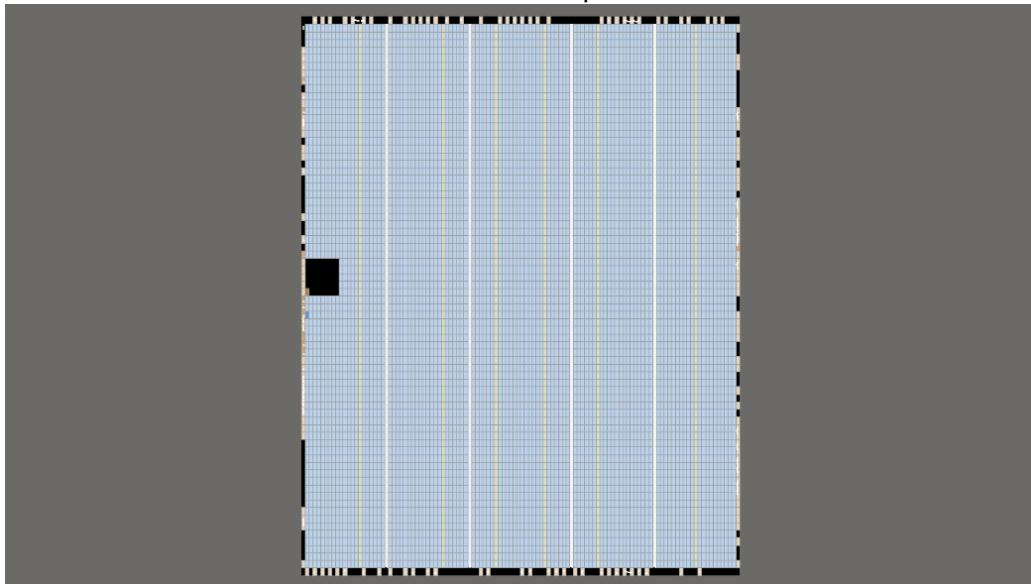


Figura 18: Representação do Chip-planner sobre o CI

Em seguida, aumentei o zoom para identificar as unidades lógicas utilizadas pelo circuito para construção do contador:

FONTE: Elaborado pelo autor

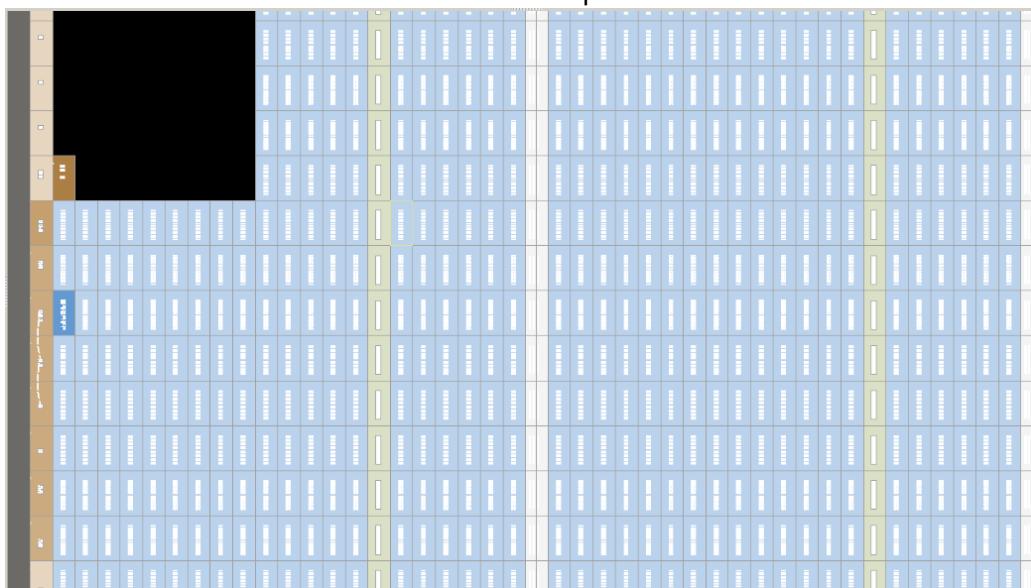


Figura 19: Representação do Chip-planner sobre o CI

Ao aproximar suficientemente, os componentes representativos do hardware começam a se tornar evidentes, no caso abaixo, os componentes na extrema esquerda (em marrom) são responsáveis por entrada/saída do chip.

Enquanto que, os dispositivos apresentados na região azul-escuro, estão sendo utilizados para a construção do contador a nível de hardware:

FONTE: Elaborado pelo autor

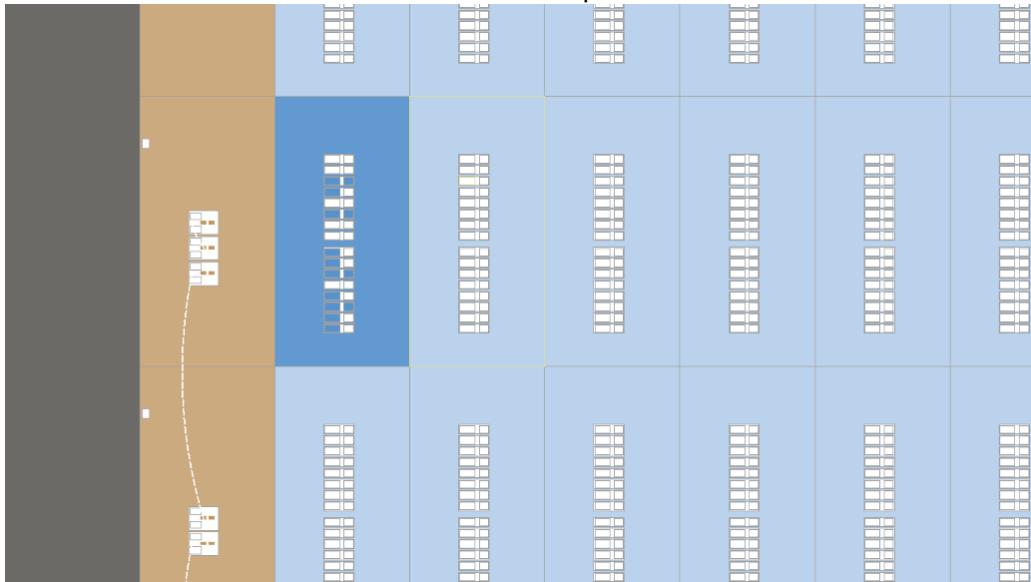


Figura 20: Representação do Chip-planner sobre o CI

Ao alterar as labels do visualizador, é possível habilitar a visualização de conexões entre as unidades lógicas e também conexões com os componentes de I/O, conforme representado abaixo:

FONTE: Elaborado pelo autor



Figura 21: Representação das conexões entre as unidades lógicas

As conexões em amarelo apresentadas na figura 21 representam as utilizadas pelo CI para ligar todos os componentes e formar o contador.

As conexões representadas em branco não estão sendo utilizadas pelo circuito, e dessa forma, estão livres para utilização para montar outros circuitos.

Ao aproximar ainda mais o visualizador, é exibido o rótulo identificador do dispositivo de I/O, conforme ilustrado abaixo:

FONTE: Elaborado pelo autor

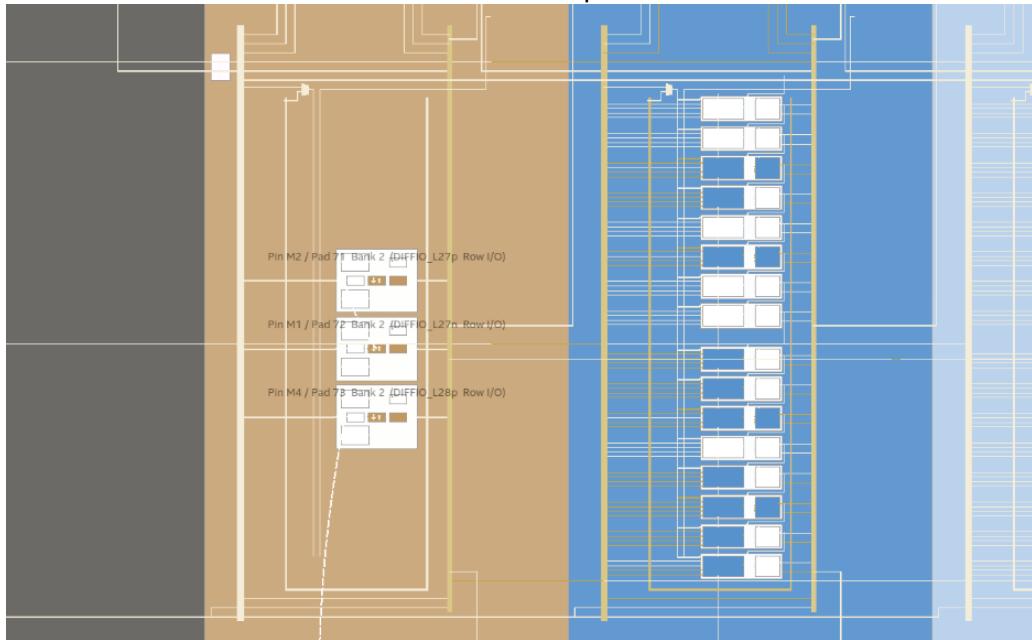


Figura 22: Rótulo identificador da unidade de I/O

Em seguida, accesei ao clicar sobre um dos componentes lógicos, verifiquei as diferenças entre o circuito deste dispositivo ao compara-lo com o dispositivo anterior.

O circuito abaixo está contido em cada unidade lógica do dispositivo, note que a poucas diferenças associadas especificamente a esse cirucito, apenas a quantidade de entradas é maior e mux adicionais para configurações no flip-flop.

FONTE: Elaborado pelo autor

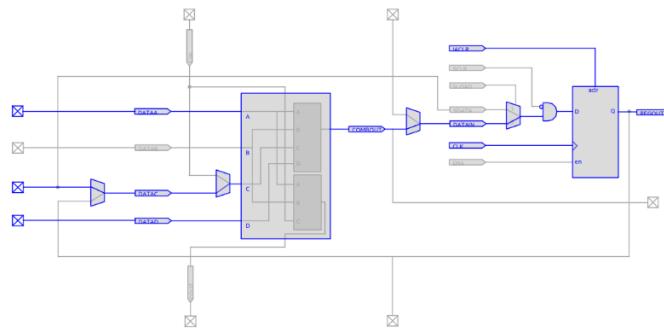


Figura 23: Circuito de unidade lógica do dispositivo

Para o circuito de I/O entretanto, diversas mudanças ocorreram, adicionando diversos sub-circuitos além do principal para receber o sinal de entrada.

FONTE: Elaborado pelo autor

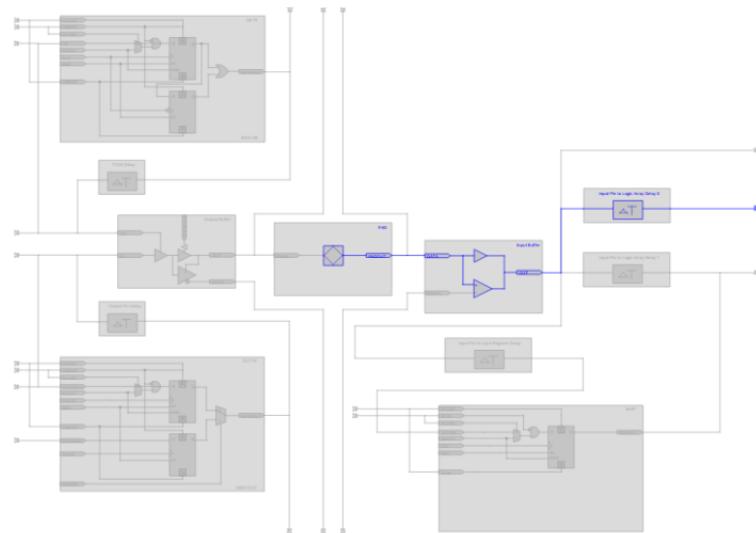


Figura 24: circuito de I/O do dispositivo - visão total

Cada bloco de circuito contém uma descrição de funcionalidade, os circuitos que contém Flip-flop como na ilustração abaixo estão conectados diretamente a um circuito de buferização para a saída do sinal no pino.

FONTE: Elaborado pelo autor

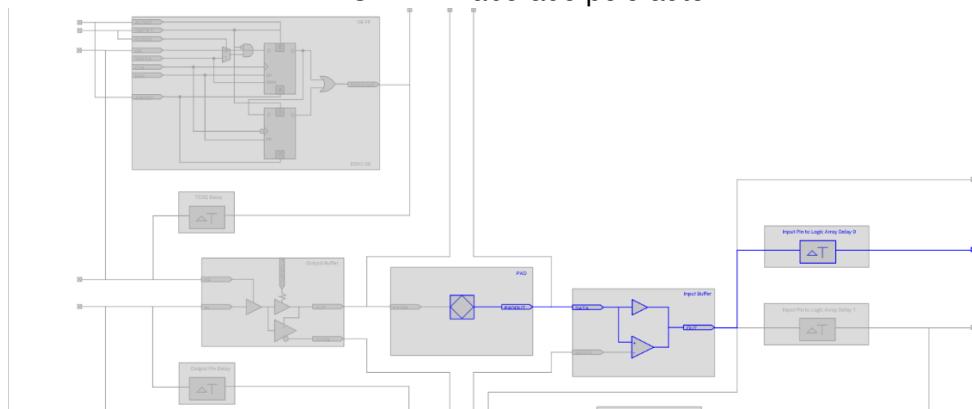


Figura 25: circuito de I/O do dispositivo - visão superior

Na vizão inferior do diagrama, outros dois blocos são apresentados, ambos os blocos apresentam flip-flops com diversos pinos para configuração via software, além de que a saída do flip-flop está conectada a um circuito de buferização de entrada e a um circuito denominado "Logic Array Relay", possivelmente utilizado para atrasar/propagar continuamente o sinal de entrada por um período de tempo.

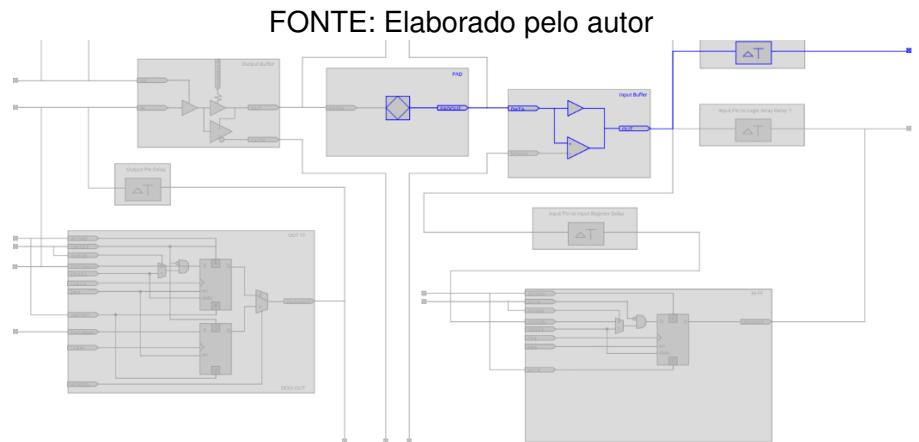


Figura 26: circuito de I/O do dispositivo - visão inferior

Em seguida, após verificar as alterações de cada componente físico, verifiquei novamente o diagrama lógico do contador, utilizado visão RTL sobre o circuito, para essa representação, nenhuma mudança significativa foi encontrada.

Para verificar as diferenças entre o mapeamento tecnológico do dispositivo anterior para esse, acessei novamente a ferramenta "Technology Map".

Nesta representação foram adicionados alguns componentes extras ao circuito, de acordo com a descrição do software, são células lógicas combinacionais. Provavelmente foram adicionadas a visão de tecnologia pois são parte do hardware base do dispositivo.

FONTE: Elaborado pelo autor

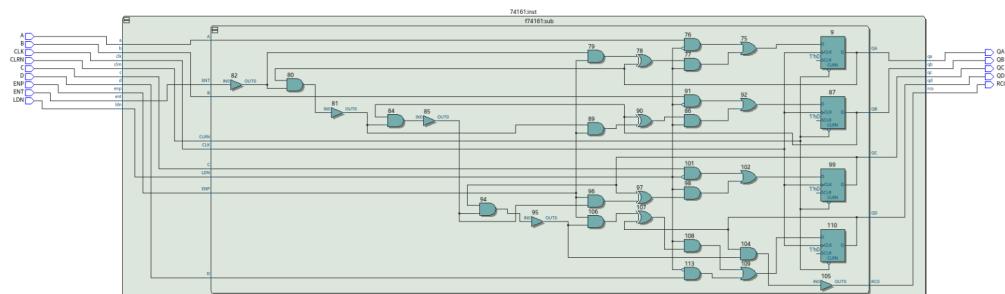


Figura 27: Diagrama RTL do circuito

FONTE: Elaborado pelo autor

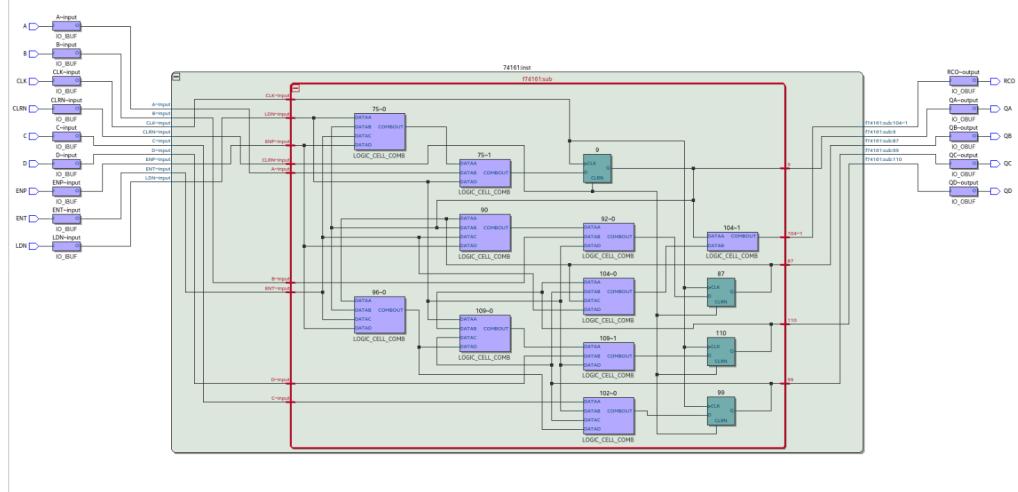


Figura 28: Visualização do circuito através do Technology Map

### 2.3 Passo 3:

Para o terceiro laboratório, é necessário criar um novo projeto para iniciar o desenvolvimento de circuitos através da escrita em VHDL. Para isso, vou criar um projeto denominado "hamming-distance":

FONTE: Elaborado pelo autor

New Project Wizard (em TELE-Matlab-Quartus)

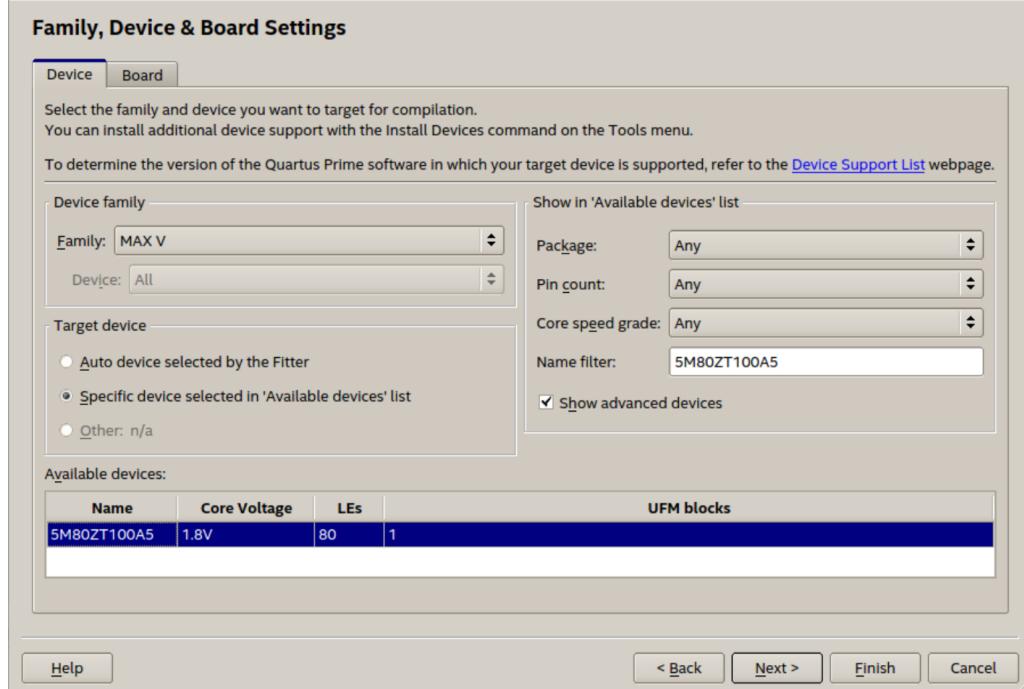


Figura 29: Configurações do projeto hamming\_distance

Ao criar um projeto, vou adicionar um novo arquivo HDL e adicionar o código abaixo (VHDL) que descreverá o circuito a ser implementado em hardware:

FONTE: Elaborado pelo autor

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity hamming_distance is
    generic (
        N:     natural := 25;
        M:     natural := 5); -- M = ceil(log2(N))
    port(
        a, b :  in std_logic_vector (N-1 downto 0);
        y :    out std_logic_vector (M-1 downto 0));
end entity;

architecture ifsc_arch_gen of hamming_distance is
    signal diff: unsigned (N-1 downto 0);
    signal sum: unsigned (M-1 downto 0);
begin
    diff <= unsigned(a xor b);
    process (diff)
        variable tmp : integer range 0 to N;
    begin
        tmp := 0;
        for i in diff'range loop
            tmp := tmp + to_integer(unsigned('0' & diff(i)));
        end loop;
        sum <= to_unsigned(tmp,M);
    end process;
    y <= std_logic_vector(sum);
end architecture;

```

Figura 30: Código VHDL utilizado no projeto

O código apresentado acima realiza o calculo de hamming para duas entradas inseridas, de maneira simples, retorna um valor que simboliza quão distante um sinal está do outro, ou quão diferente um sinal se parece com outro.

Em seguida, vou realizar a compilação do projeto com o código VHDL devidamente inserido no projeto e salvo. Um vez com o projeto em compilação, é importante verificar que não há nenhum erro encontrado pelo compilador durante a interpretação do código, conforme abaixo:

FONTE: Elaborado pelo autor

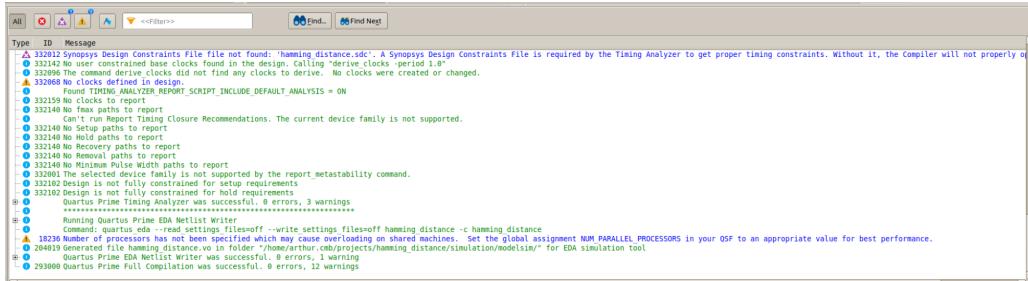


Figura 31: Resultados da compilação do projeto

No caso ilustrado, novamente o warning mais preocupante está relacionado com a quantidade de processadores utilizados pelo software no momento da compilação do projeto, entretanto, durante os teste que realizei, não necessitei alterar a configuração, portanto mantive como padrão.

Ao verificar os resultados da compilação, podemos notar que 68 de 80 segmentos lógicos estão sendo utilizados pelo dispositivo, além disso 55 dos 79 pinos de I/O estão em uso.

O objetivo da otimização do circuito é diminuir a quantidade de componentes (caso seja possível), e diminuir o tempo de propagação entre os pinos de entrada e saída o maximo possivel, podendo assim aumentar a eficiência do dispositivo e circuito.

FONTE: Elaborado pelo autor

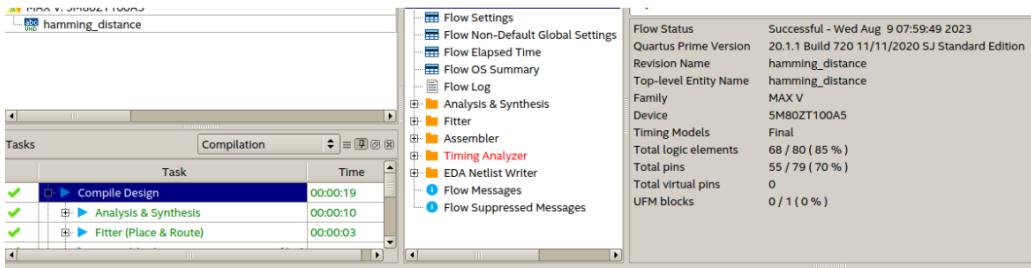


Figura 32: Status geral de compilação do projeto

Uma vez com o projeto compilado, accesei novamente a ferramenta Pin-Planner para verificar o status da pinagem do dispositivo. Note que neste exemplo, diversos foram necessários para a configuração solicitada.

Portanto, ao analisar a distribuição automatizadas dos pinos no CI, nota-se que a maioria dos pinos foram encaixados de maneira proxima para diminuir a latência gerada por propagação:

FONTE: Elaborado pelo autor

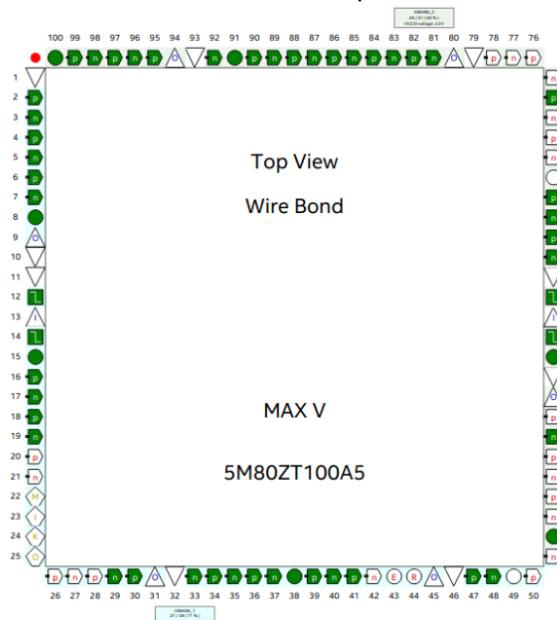


Figura 33: Visão do dispositivo lógico - Pin Planner

Da mesma maneira como nos laboratórios anteriores, coletei a tabela de pinagem gerada pelo quartus para observar as diferentes faixas de tensão em cada pindo devido a diferença e modelos/familias.

FONTE: Elaborado pelo autor

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair	strict Preservative
in_a[24]	Input			PIN_81	3.3-V L...efault		16mA (default)		
in_a[23]	Input			PIN_15	3.3-V L...efault		16mA (default)		
in_a[22]	Input			PIN_5	3.3-V L...efault		16mA (default)		
in_a[21]	Input			PIN_8	3.3-V L...efault		16mA (default)		
in_a[20]	Input			PIN_90	3.3-V L...efault		16mA (default)		
in_a[19]	Input			PIN_91	3.3-V L...efault		16mA (default)		
in_a[18]	Input			PIN_89	3.3-V L...efault		16mA (default)		
in_a[17]	Input			PIN_57	3.3-V L...efault		16mA (default)		
in_a[16]	Input			PIN_67	3.3-V L...efault		16mA (default)		
in_a[15]	Input			PIN_47	3.3-V L...efault		16mA (default)		
in_a[14]	Input			PIN_66	3.3-V L...efault		16mA (default)		
in_a[13]	Input			PIN_69	3.3-V L...efault		16mA (default)		

Figura 34: Visão do dispositivo lógico - Pin Planner

Em seguida, accesei novamente a ferramenta chip-planner para verificar o status de ocupação de cada porta. Nota-se que a maioria dos elementos de I/O estão alocados para o circuito apontado.

FONTE: Elaborado pelo autor

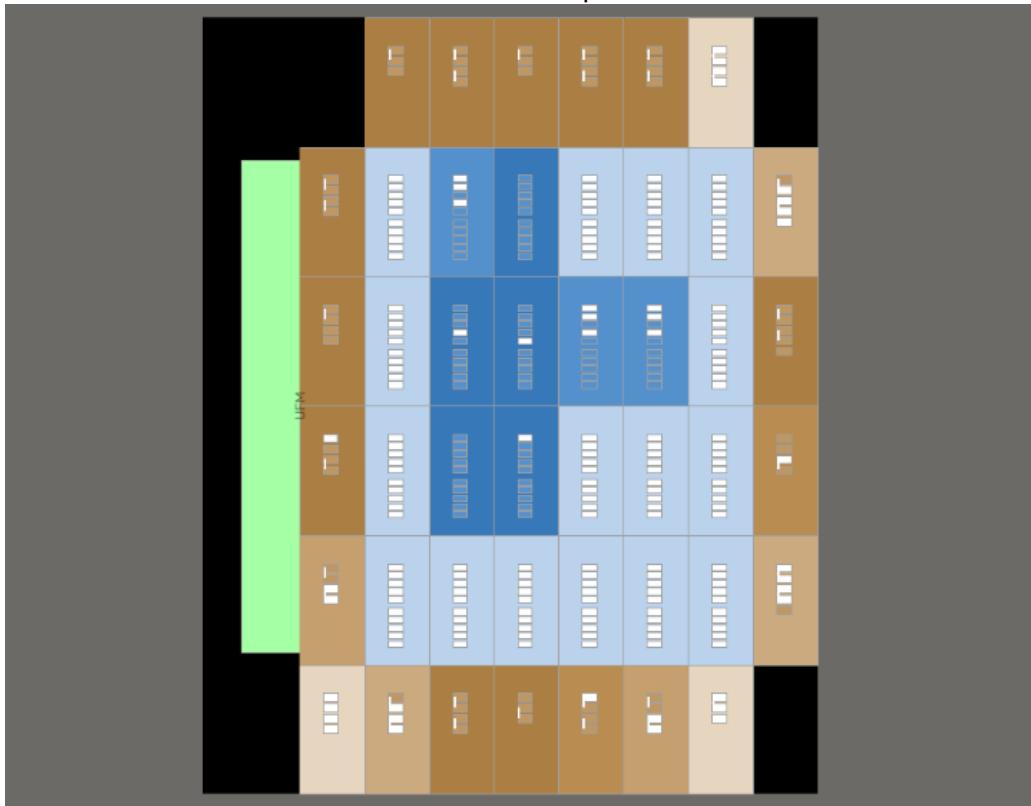


Figura 35: Visão do dispositivo lógico - Chip Planner

Note que na disposição dos elementos lógicos não há um padrão ou mesmo sequencia a ser seguida. Dessa forma, para otimizar o uso do dispositivo e maximizar sua eficiência, retornarei a quartus para verificar as possíveis opções.

Para tentar solucionar a dificuldade no tempo de propagação, vamos analisar os possíveis casos através da ferramenta "Propagation Relay":

Na configuração da ferramenta, é possível calcular usando dois modos o modo "fast" que prioriza tempo de propagação minimo e o modo "slow" que prioriza eficiência, área ocupada, etc.

Neste primeiro exemplo, vamos ver o tempo de resposta padrão para o modo slow:

FONTE: Elaborado pelo autor



Figura 36: Tempo de propagação Slow - Sem alterações

Em seguida, sem alterar qualquer outro parâmetro no quartus, vou alterar o modo da ferramenta

para "Fast" e verificar o resultado:

FONTE: Elaborado pelo autor

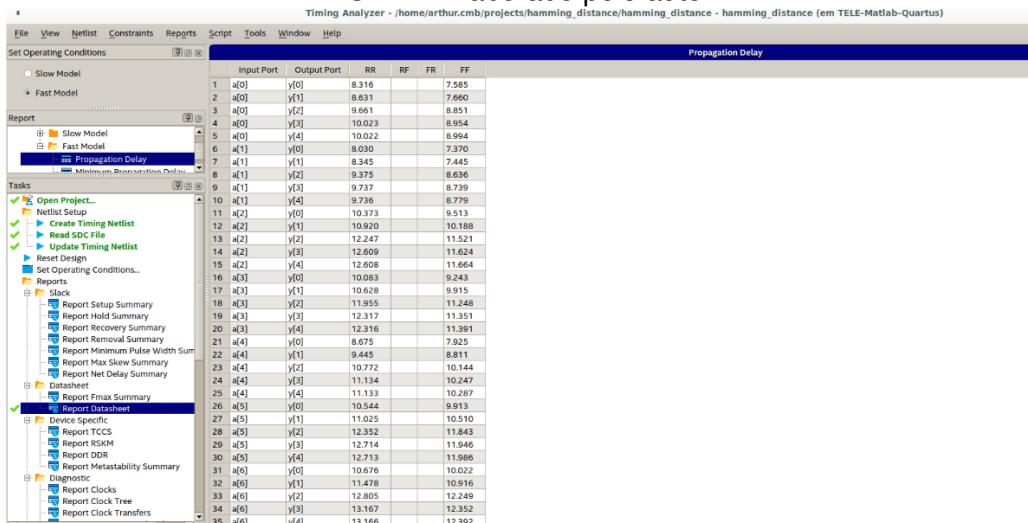


Figura 37: Tempo de propagação Fast - Sem alterações

Note que a mudança foi bastante significativa, o primeiro teste resultou em aproximadamente 27ms no melhor caso, já o teste em fast, retorno 8ms em média.

Em seguida, apenas para entender a extensão do circuito, vou verificar o "Technology-Map" do circuito e verificar a quantidade de elementos lógicos e componentes de hardware que estão utilizando o dispositivo, uma ilustração é apresentada abaixo:

FONTE: Elaborado pelo autor

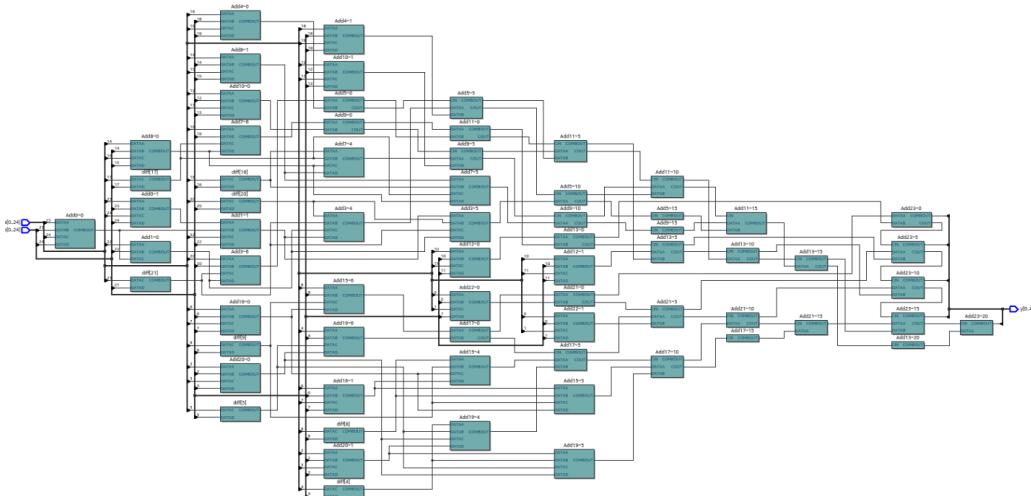
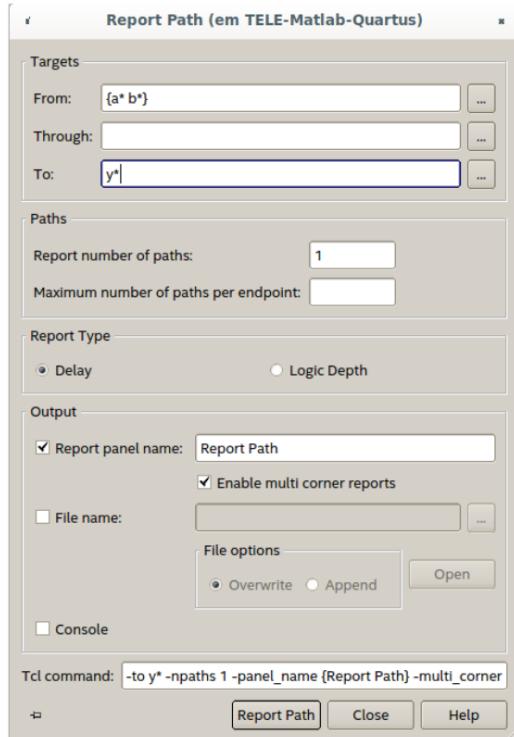


Figura 38: Technology Map - comparativo

Note que devido a complexidade do circuito e sua semântica ter sido alterada, a quantidade de componentes de hardware gerados na ferramenta subiu significativamente.

Uma vez com os valores padrão de propagação já mapeados, vou verificar como esses valores são obtidos, ou seja, qual rota está sendo tomada do inicio ao fim do circuito para gerar um valor específico de propagação.

Para isso, na ferramenta "Time Analyser" vou criar um report de caminho, que irá exibir o caminho das entradas (neste caso "a" e "b"), e da saída do circuito (neste caso "y"). Conforme a captura ilustra abaixo:



FONTE: Elaborado pelo autor

Figura 39: Configuração do Report de caminho (Report Path)

Após fazer os calculos, a ferramenta irá trazer os possíveis resultados (de acordo com o número de entradas inserido), neste exemplo inicial, apenas 1 rota foi solicitada para a criação.

	Total	Incr	RF	Type	Fanout	Location	Element
1	16.289	1.150	RR	CELL	0	PN_62	y[4]
2	0.000	0.000					data path
3	0.761	0.761	RR	CELL	2	IOC_X1_Y1_N0	b[24]-0dataout
4	3.372	2.611	RR	IC	1	LC_X4_Y2_N8	Add11-0dataout
5	3.068	0.581	RR	CELL	2	LC_X3_Y2_N9	Add11-0combout
6	4.437	0.469	RR	IC	1	LC_X3_Y3_N4	Add11-0dataout
7	4.562	0.125	RR	CELL	3	LC_X3_Y3_N4	Add11-0combout
8	-6.001	1.439	RR	IC	1	LC_X4_Y4_N3	Add11-0dataaa
9	6.679	0.678	RR	CELL	1	LC_X4_Y4_N3	Add11-0combout
10	7.718	0.039	RR	IC	3	LC_X4_Y4_N5	Add11-0dataout
11	7.764	0.000	RR	IC	2	LC_X4_Y4_N6	Add11-5icout1
12	-7.826	0.062	RR	CELL	1	LC_X4_Y4_N6	Add11-5icout1
13	7.826	0.000	RR	IC	2	LC_X4_Y4_N7	Add11-10jicin1
14	-8.220	0.394	RR	CELL	1	LC_X4_Y4_N7	Add11-10jicout
15	9.982	1.762	RR	IC	3	LC_X4_Y3_N3	Add11-10jicata
16	-10.484	0.502	RR	CELL	1	LC_X4_Y3_N3	Add11-10jicout
17	10.628	0.000	RR	IC	1	LC_X4_Y3_N3	Add11-15jicin1
18	11.022	0.394	RR	CELL	1	LC_X4_Y3_N3	Add11-15jicombout
19	11.468	0.446	RR	IC	3	LC_X4_Y3_N3	Add11-3-j5dataaa
20	-12.146	0.678	RR	CELL	1	LC_X4_Y3_N3	Add11-3-j5combout
21	13.232	1.086	RR	IC	3	LC_X4_Y2_N8	Add11-3-j5dataab
22	-13.30	0.000	RR	CELL	1	LC_X4_Y2_N8	Add11-10jicout
23	13.735	0.000	RR	IC	1	LC_X4_Y2_N9	Add12-3-j5dataab
24	14.110	0.375	RR	CELL	1	LC_X4_Y2_N9	Add12-3-20jicout
25	15.139	1.029	RR	IC	1	IOC_X8_Y2_N0	y[4]datain
26	16.289	1.150	RR	CELL	0	PN_62	y[4]

Figura 40: Tabela de roteamento I/O - Tempos de propagação

Neste primeiro exemplo, o report foi gerado para uma única saída, ao selecionar a saída calculada e abri-la na ferramenta "Chip-Planner", vamos obter uma rota de propagação do sinal entre as entradas do circuito e a saída.

Uma vez no chip planner, o report ficará claro, é nota-se facilmente a rota de propagação de inicio ao fim do circuito e também os tempos de propagação.

NOTA: O somatório dos tempos de propagação exibido na imagem abaixo não corresponde ao valor exibido anteriormente na saída da ferramenta "Time Analyser", isso ocorre pois alguns tempos ficam ocultos devido a proximidade de texto. Ao utilizar o zoom sobre a ferramenta os tempos de propagação dos componentes menores estão medidos corretamente.

FONTE: Elaborado pelo autor

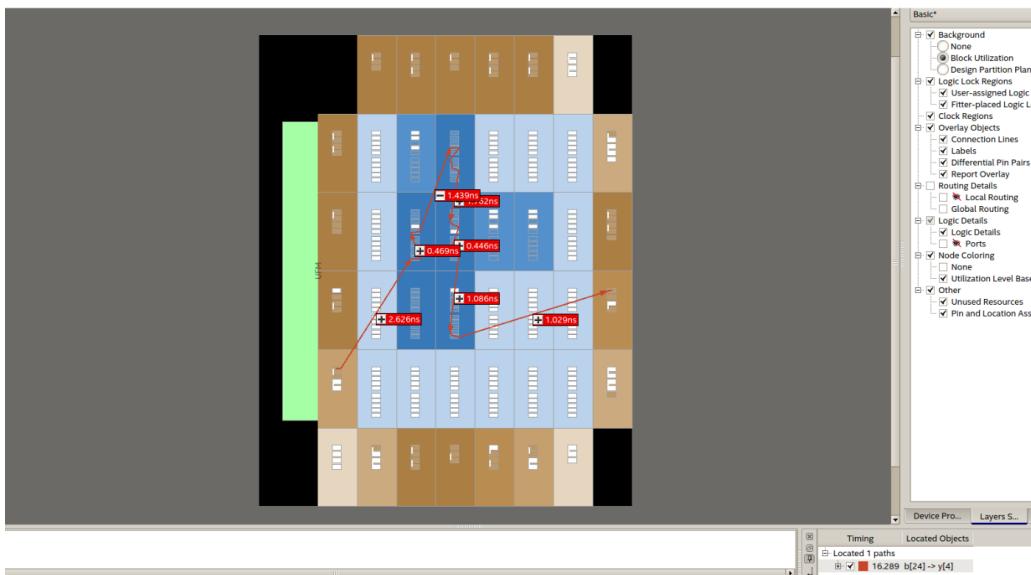


Figura 41: Mapa de roteamento I/O - Com tempos de propagação

## 2.4 Passo 3 - Otimizações:

A primeira otimização que tentarei fazer para diminuir o tempo de propagação está associada as configurações de compilação do projeto.

Por padrão, o quartus compila de maneira equilibrada entre desempenho e economia de recursos, como energia, espaço, etc. Neste exemplo, irei focar em desempenho e posteriormente área de utilização do dispositivo.

Para isso, vamos alterar as configurações do compilador para otimizar a velocidade dos programas através da compilação.

Abaixo fiz a alteração do modo "Balanced" para "Performance" de maneira agressiva, ou seja, agora o software está tentando de todas as maneiras melhorar a performance do equipamento, não se importando com demais fatores associados.

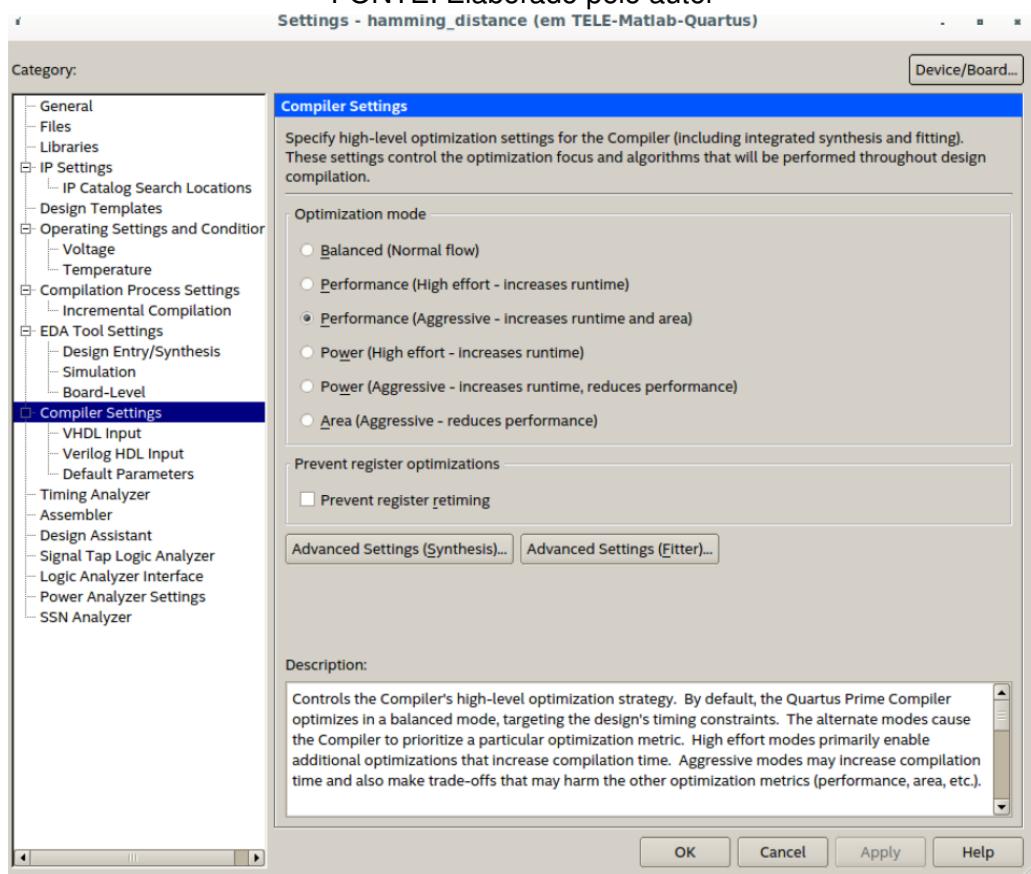


Figura 42: Configurações do compilador - Otimização por performance

Em seguida, acesei novamente as configurações de time analisys para comparar os dados anteriores de tempo de propagação com os valores atuais (com o circuito idealmente otimizado):

Utilizando o modo "Fast" nas comparações nota-se que a diferença de propagação não foi significativa, o valor apos as alterações teve uma melhora próxima de 2ms, ao comparada com o valor anterior:

FONTE: Elaborado pelo autor

Total	Incr	RF	Type	Fanout	Location	Element	
14.902	0.000			1	PIN_97	a[24]	
2	0.761	RR	CELL	2	IOC_X3_Y5_N3	a[24]  combout	
3	3.334	2.573	RR	IC	1	LC_X5_Y4_N3	Add3  odataab
4	4.391	0.937	RR	CELL	2	LC_X5_Y4_N3	Add3  odataout
5	4.379	0.483	RR	RR	1	LC_X5_Y4_N1	Add3  odataout
6	5.056	0.678	RR	CELL	3	LC_X5_Y4_N1	Add3  combout
7	5.800	0.744	RR	IC	1	LC_X6_Y4_N3	Add3  odataac
8	6.129	0.329	RR	CELL	1	LC_X6_Y4_N3	Add3  combout
9	6.568	0.439	RR	IC	3	LC_X6_Y4_N5	Add3  odataac
10	7.149	0.400	RR	CELL	1	LC_X6_Y4_N5	Add3  cout1
11	7.214	0.000	RR	RR	2	LC_X6_Y4_N6	Add3  cout1
12	7.608	0.304	RR	CELL	1	LC_X6_Y4_N6	Add3  5 combout
13	8.663	1.055	RR	IC	3	LC_X6_Y3_N1	Add3  5 datas
14	9.309	0.646	RR	CELL	1	LC_X6_Y3_N1	Add3  1 cout1
15	9.309	0.000	RR	IC	2	LC_X6_Y3_N2	Add3  1 cout1
16	9.703	0.394	RR	CELL	1	LC_X6_Y3_N2	Add3  1 5 coutout
17	10.253	0.440	RR	RR	2	LC_X6_Y3_N2	Add3  1 5 coutout
18	10.744	0.581	RR	CELL	1	LC_X6_Y2_N7	Add3  3  combout
19	11.753	1.009	RR	IC	3	LC_X5_Y3_N7	Add3  3  5 stab
20	12.256	0.503	RR	CELL	1	LC_X5_Y3_N7	Add3  3  coutD
21	12.256	0.000	RR	IC	2	LC_X5_Y3_N8	Add3  3  5 m0
22	12.322	0.066	RR	CELL	1	LC_X5_Y3_N8	Add3  3  5 coutD
23	12.322	0.000	RR	RR	2	LC_X5_Y3_N8	Add3  3  20 m0
24	12.697	0.375	RR	CELL	1	LC_X5_Y3_N8	Add3  3  20 cout
25	13.752	1.055	RR	IC	1	IOC_X1_Y2_NO	y[4] datas
26	14.902	1.150	RR	CELL	0	PIN_6	y[4]

Figura 43: Tabela de roteamento com otimização de performance

Podemos compreender como essa melhora ocorreu verificando novamente a montagem do dispositivo através do programa "Chip-Planner":

FONTE: Elaborado pelo autor

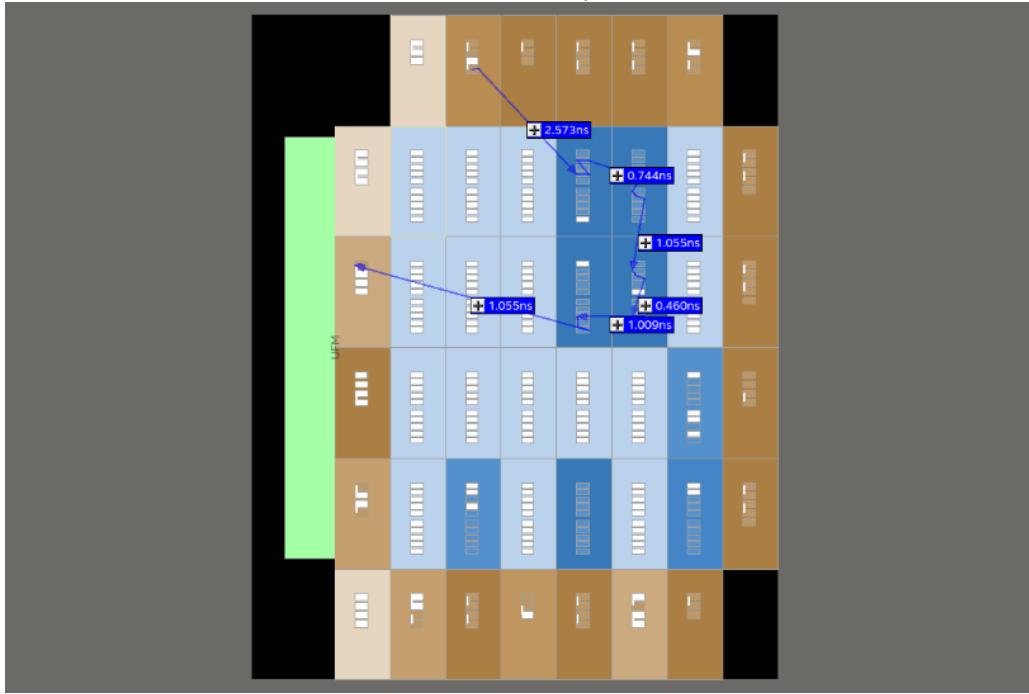


Figura 44: roteamento com otimização de performace

Note que neste segundo caso, o roteamento entre a interface de entrada e interface de saída é muito mais proximo, e as unidades lógicas utilizadas também se localizam proximas umas das outras. Isso permite um tempo menor de propagação entre o pino de entrada e o pino de saída.

É possível observar os tempos de propagação individualmente aumentando o zoom até atingir as unidades lógicas, conforme a ilustração abaixo:

FONTE: Elaborado pelo autor

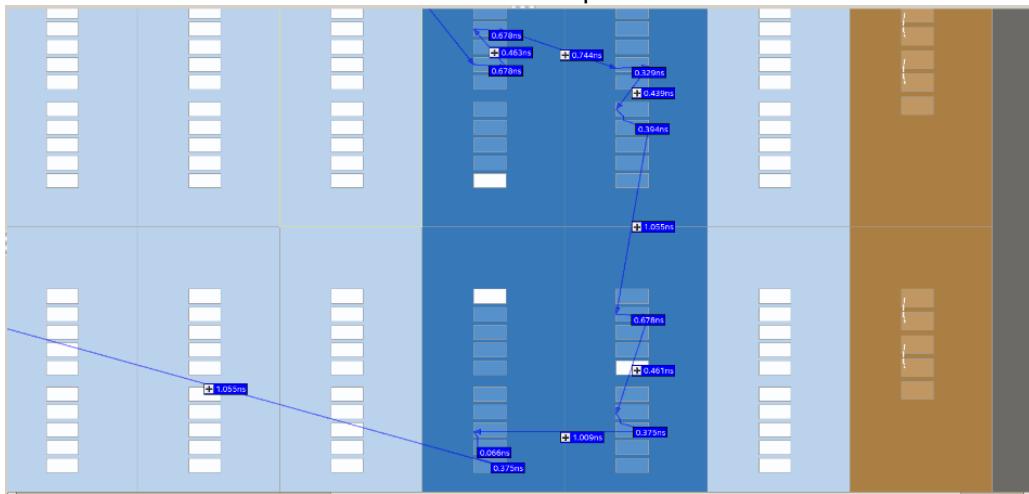


Figura 45: roteamento com otimização de performace - tempos de propagação

Para tentar melhorar de maneira mais significativa o tempo de propagação de I/O, também alterei a seed utilizada pelo compilador para determinar o caminho a ser utilizado. Lembrando que nesse passo, a configuração ainda se mantém comoaltar performace, focando em desempenho.

A ideia da alteração da seed é que eventualmente, o compilador poderá encontrar uma distribuição de roteamento com valor reduzido que poderá melhorar o tempo de propagação significativamente.

FONTE: Elaborado pelo autor

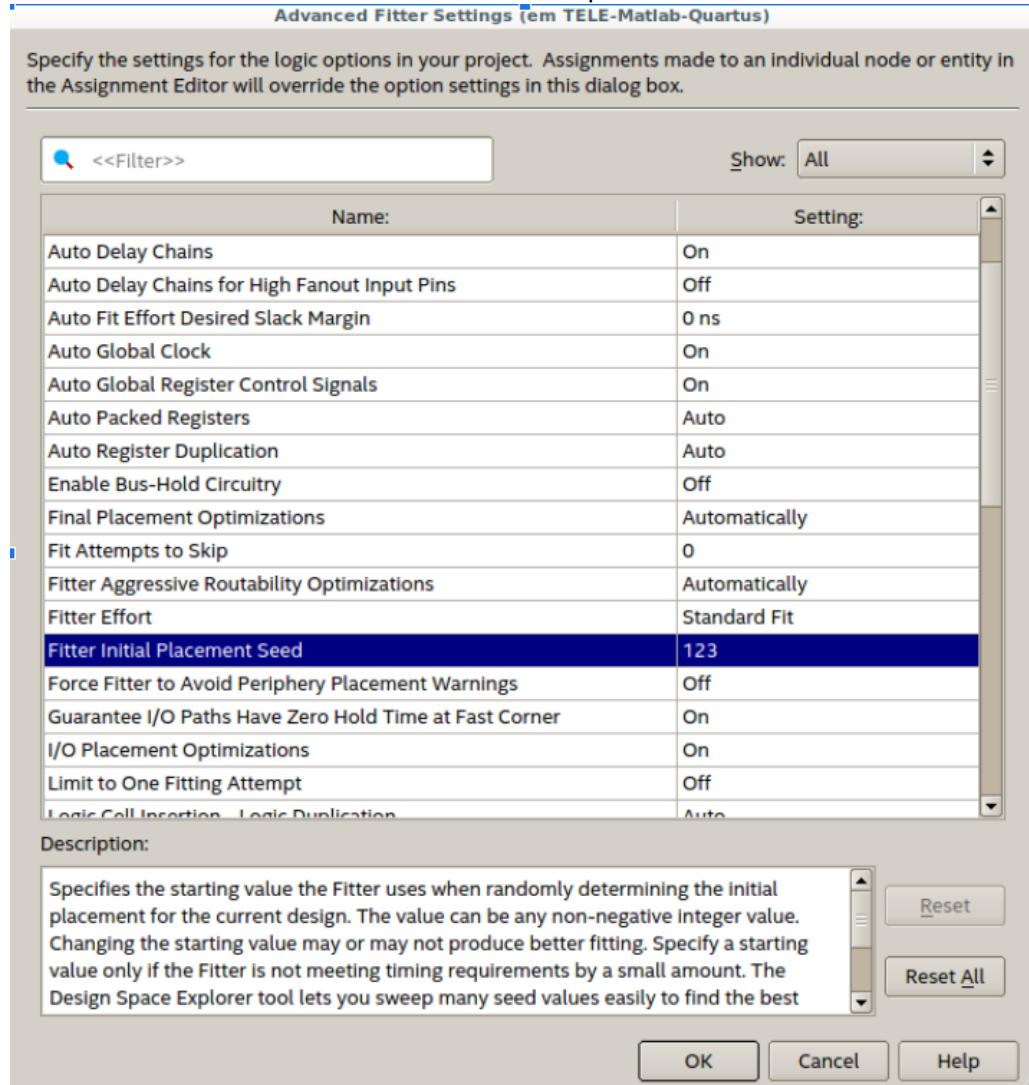


Figura 46: Configurações de compilação - Alteração da seed

Neste exemplo a seed sofreu alteração para o valor 123, conforme apresentado acima.

Após os mesmos testes apresentados acima, não obtive melhora, e sim piora de 1,2ms, com a distribuição de roteamento apresentada abaixo:

FONTE: Elaborado pelo autor

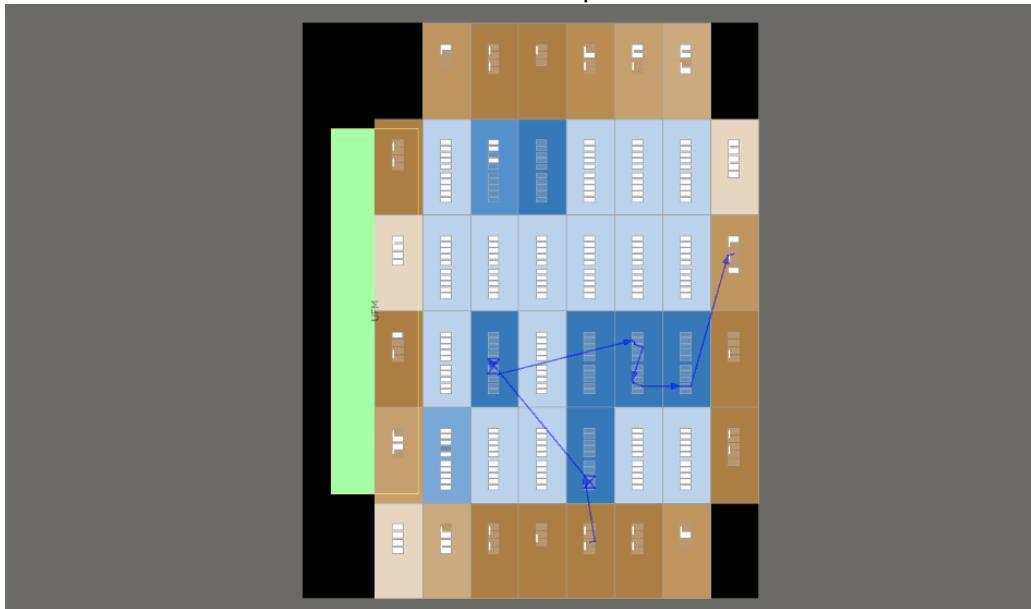


Figura 47: Roteamento com otimização por seed

Para tentar forçar o software a operar até encontrar um valor mínimo de propagação, adicionei um script para tentar orientar ao quartus a trabalhar com um valor específico de propagação.

FONTE: Elaborado pelo autor

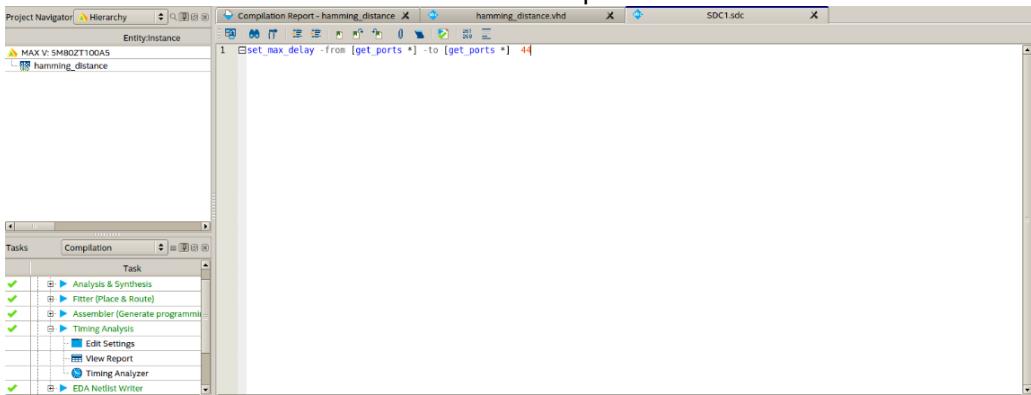


Figura 48: script de otimização forçada

Dessa forma, idealmente, o quartus deve recalcular a seed até conseguir entrar no tempo de propagação indicado. Por recomendação do professor, reduzi de 5 em 5ms o tempo de propagação até chegar em um valor mínimo de propagação total.

FONTE: Elaborado pelo autor

The screenshot shows the Quartus II Timing Analyzer interface. The left pane displays a table of contents for timing analysis, including Flow Summary, Flow Settings, Flow Non-Default Global Setting, Flow Elapsed Time, Flow OS Summary, Flow Log, Analysis & Synthesis, Filter, and Timing Analyzer. The Timing Analyzer section is expanded, showing sub-options like Summary, Parallel Compilation, SDC File List, Clocks, Fmax Summary, Setup Summary, Hold Summary, Recovery Summary, Removal Summary, Minimum Pulse Width Summary, Worst-Case Timing Paths, and Setup 'n/a'. The right pane is a table titled 'Setup: 'n/a'' with columns: Slack, From Node, To Node, Launch Clock, Latch Clock, Relationship, Clock Skew, and Data Delay. The table lists 25 rows of timing data, with many entries showing negative slack values such as -1.753, -1.742, -1.23, -1.048, -0.921, -0.783, -0.658, -0.484, -0.418, -0.216, -0.188, -0.066, -0.047, 0.014, 0.089, 0.254, 0.283, 0.301, 0.424, 0.452, 0.560, 0.629, 0.639, 0.656, and 0.658.

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-1.753	a[14]	y[3]	n/a	n/a	44.000	0.000	45.753
2	-1.742	a[7]	y[3]	n/a	n/a	44.000	0.000	45.742
3	-1.23	b[24]	y[3]	n/a	n/a	44.000	0.000	45.123
4	-1.048	a[14]	y[4]	n/a	n/a	44.000	0.000	45.048
5	-0.921	a[24]	y[3]	n/a	n/a	44.000	0.000	44.921
6	-0.783	a[7]	y[4]	n/a	n/a	44.000	0.000	44.783
7	-0.658	b[6]	y[3]	n/a	n/a	44.000	0.000	44.658
8	-0.484	a[14]	y[1]	n/a	n/a	44.000	0.000	44.484
9	-0.418	b[24]	y[4]	n/a	n/a	44.000	0.000	44.418
10	-0.216	a[24]	y[4]	n/a	n/a	44.000	0.000	44.216
11	-0.188	b[24]	y[1]	n/a	n/a	44.000	0.000	44.188
12	-0.066	a[15]	y[3]	n/a	n/a	44.000	0.000	44.066
13	-0.047	b[23]	y[3]	n/a	n/a	44.000	0.000	44.047
14	0.014	a[24]	y[1]	n/a	n/a	44.000	0.000	43.986
15	0.089	a[12]	y[1]	n/a	n/a	44.000	0.000	43.911
16	0.254	a[6]	y[3]	n/a	n/a	44.000	0.000	43.746
17	0.283	a[7]	y[1]	n/a	n/a	44.000	0.000	43.717
18	0.301	b[6]	y[4]	n/a	n/a	44.000	0.000	43.699
19	0.424	a[12]	y[3]	n/a	n/a	44.000	0.000	43.576
20	0.452	a[14]	y[2]	n/a	n/a	44.000	0.000	43.548
21	0.560	b[24]	y[2]	n/a	n/a	44.000	0.000	43.440
22	0.629	a[23]	y[3]	n/a	n/a	44.000	0.000	43.371
23	0.639	a[15]	y[4]	n/a	n/a	44.000	0.000	43.361
24	0.656	b[14]	y[3]	n/a	n/a	44.000	0.000	43.344
25	0.658	b[23]	y[4]	n/a	n/a	44.000	0.000	43.342

Figura 49: timing Report - Negative Slack

Em seguida, novamente acessei a ferramenta de "Time Analysis", para validar o tempo de roteamento entre entrada e saída, conforme os testes anteriores.

Ao realizar a diminuição desta maneira, o tempo de propagação foi reduzido em quase 4ms se comparado ao tempo original (sem otimizações), no total, foi atingido 13,295ms, conforme a ilustração abaixo:

The screenshot shows the Quartus II Command Line Interface (CLI) with three tables of routing information. The top table, 'Summary of Paths', lists paths from delay 13.295 to 13.658. The middle table, 'Path #1: Delay is 13.295', shows detailed statistics for path 1 (Total: 13.295, Incr: 0.000, RF: 0, Type: IC, Fanout: 1, Location: PIN\_B9, Element: b[24]) and path 2 (Total: 13.295, Incr: 0.000, RF: 0, Type: IC, Fanout: 1, Location: IOC\_X4\_Y5\_N0, Element: b[24]lcombout). The bottom table, 'Path #1: Delay is 13.295', shows detailed statistics for path 1 (Total: 13.295, Incr: 0.000, RF: 0, Type: IC, Fanout: 1, Location: PIN\_B9, Element: b[24]) and path 2 (Total: 13.295, Incr: 0.000, RF: 0, Type: IC, Fanout: 1, Location: IOC\_X4\_Y5\_N0, Element: b[24]lcombout).

Path Summary									Path Summary									Path Summary								
	Total	Incr	RF	Type	Fanout	Location	Element		Total	Incr	RF	Type	Fanout	Location	Element		Total	Incr	RF	Type	Fanout	Location	Element			
1	13.295	13.295					data path		1	13.295	13.295						1	13.295	13.295					data path		
2	0.000	0.000			1	PIN_B9	b[24]		2	0.000	0.000			1	PIN_B9	b[24]		2	0.000	0.000			1	PIN_B9	b[24]	
3	0.761	0.761	RR	CELL	2	IOC_X4_Y5_N0	b[24]lcombout		3	0.761	0.761	RR	CELL	1	IOC_X4_Y5_N0	b[24]lcombout		3	0.761	0.761	RR	CELL	1	IOC_X4_Y5_N0	b[24]lcombout	
4	0.254	0.254	RR	CELL	2	LC_X5_Y2_N1	Add0-0 data0		4	0.254	0.254	RR	CELL	1	LC_X5_Y2_N1	Add0-0 data0		4	0.254	0.254	RR	CELL	1	LC_X5_Y2_N1	Add0-0 data0	
5	0.693	0.693	RR	CELL	2	LC_X5_Y2_N5	Add0-1 data0		5	0.693	0.693	RR	IC	1	LC_X5_Y2_N5	Add0-1 data0		5	0.693	0.693	RR	IC	1	LC_X5_Y2_N5	Add0-1 data0	
6	0.560	0.560	RR	CELL	2	LC_X5_Y2_N5	Add1-1 combout		6	0.560	0.560	RR	CELL	2	LC_X5_Y2_N5	Add1-1 combout		6	0.560	0.560	RR	CELL	2	LC_X5_Y2_N5	Add1-1 combout	
7	0.512	0.512	RR	IC	1	LC_X5_Y2_N5	Add1-1 data0		7	0.512	0.512	RR	IC	1	LC_X5_Y2_N5	Add1-1 data0		7	0.512	0.512	RR	IC	1	LC_X5_Y2_N5	Add1-1 data0	
8	0.580	0.580	RR	CELL	2	LC_X5_Y2_N5	Add2-1 combout		8	0.580	0.580	RR	CELL	1	LC_X5_Y2_N5	Add2-1 combout		8	0.580	0.580	RR	CELL	1	LC_X5_Y2_N5	Add2-1 combout	
9	0.646	0.646	RR	IC	3	LC_X6_Y2_N2	Add3-3 data0		9	0.646	0.646	RR	IC	3	LC_X6_Y2_N2	Add3-3 data0		9	0.646	0.646	RR	IC	3	LC_X6_Y2_N2	Add3-3 data0	
10	0.646	0.646	RR	CELL	1	LC_X6_Y2_N2	Add3-5 data0		10	0.646	0.646	RR	CELL	1	LC_X6_Y2_N2	Add3-5 data0		10	0.646	0.646	RR	CELL	1	LC_X6_Y2_N2	Add3-5 data0	
11	7.130	0.000	RR	IC	2	LC_X6_Y2_N3	Add5-10 cin1		11	7.130	0.000	RR	IC	2	LC_X6_Y2_N3	Add5-10 cin1		11	7.130	0.000	RR	IC	2	LC_X6_Y2_N3	Add5-10 cin1	
12	7.524	0.394	RR	CELL	1	LC_X6_Y2_N3	Add5-10 combout		12	7.524	0.394	RR	CELL	1	LC_X6_Y2_N3	Add5-10 combout		12	7.524	0.394	RR	CELL	1	LC_X6_Y2_N3	Add5-10 combout	
13	7.963	0.439	RR	IC	3	LC_X6_Y2_N7	Add11-10 data0		13	7.963	0.439	RR	IC	3	LC_X6_Y2_N7	Add11-10 data0		13	7.963	0.439	RR	IC	3	LC_X6_Y2_N7	Add11-10 data0	
14	8.609	0.646	RR	CELL	1	LC_X6_Y2_N7	Add11-10 cout1		14	8.609	0.646	RR	CELL	1	LC_X6_Y2_N7	Add11-10 cout1		14	8.609	0.646	RR	CELL	1	LC_X6_Y2_N7	Add11-10 cout1	
15	8.609	0.000	RR	IC	1	LC_X6_Y2_N8	Add11-15 cin1		15	8.609	0.000	RR	IC	1	LC_X6_Y2_N8	Add11-15 cin1		15	8.609	0.000	RR	IC	1	LC_X6_Y2_N8	Add11-15 cin1	
16	9.003	0.394	RR	CELL	1	LC_X6_Y2_N8	Add11-15 combout		16	9.003	0.394	RR	CELL	1	LC_X6_Y2_N8	Add11-15 combout		16	9.003	0.394	RR	CELL	1	LC_X6_Y2_N8	Add11-15 combout	

Figura 50: Tabela de roteamento otimizada - final

Por fim, verifiquei qual foi o caminho de roteamento utilizado para diminuir o tempo de propagação, nota-se que o quartus para diminuir o tempo, compactou o máximo de componentes do circuito em regiões próximas, como mostrado abaixo:

FONTE: Elaborado pelo autor

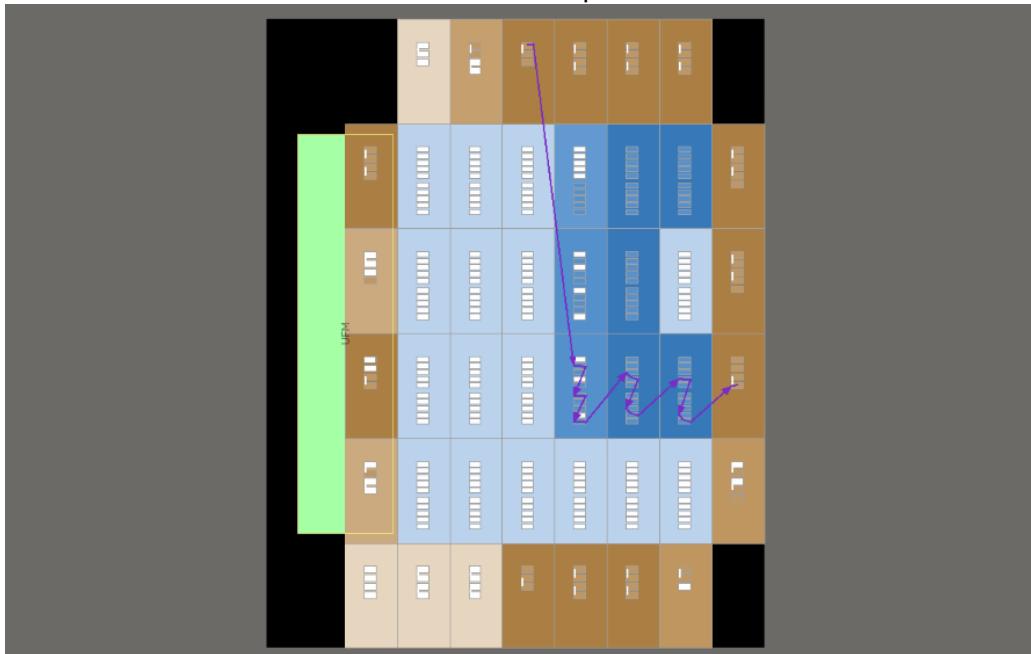


Figura 51: Roteamento com otimização forçada

### 3 Referências bibliográficas

- Wiki diária da disciplina de DLP - IFSC SJ
- Tempo de propagação em circuitos combinacionais
- Conhecendo os dispositivos lógicos programáveis
- Acesso a Nuvem do IFSC (Para realização da atividade)