UTFPR
Engenharia de Computação
Lógica Reconfigurável

Relatório da Atividade 2: Introdução à placa DE10-Lite

Aluno: Deivid da Silva Galvão
Professor orientador: Marcelo de Oliveira

Outubro
2024

UTFPR

Engenharia de Computação

Lógica Reconfigurável

Relatório

Relatório do Trabalho Prático Disciplinar apresentado como requisito parcial à obtenção de nota na disciplina de Lógica Reconfigurável do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Aluno: Deivid da Silva Galvão

Professor orientador: Marcelo de Oliveira

Outubro

2024

Conteúdo

1 Introdução **1**

2 Implementação **1**

2.1 Parte 1 1

2.2 Parte 2 3

3 Resultados obtidos **4**

1 Introdução

Na atividade 2, vamos aplicar os conceitos aprendidos anteriormente sobre o portas lógicas, focando na aplicação na linguagem VHDL, onde o objetivo foi realizar as operações logicas NOT, AND, OR, NAND, NOR, XOR, XNOR a partir das entradas "a"e "b", porém dessa vez implementando diretamente na placa e também vamos implementar um circuito especificado descobrindo sua equação por meio da tabela verdade e do mapa de Karnaugh e fazendo sua implementação diretamente na placa também.

2 Implementação

2.1 Parte 1

```
library ieee ;   -- Importando a biblioteca ieee
use ieee . std_logic_1164 . all ;
-- ------------------------------------------
entity projeto1 is  -- entidade do projeto
port (
a , b: in bit ;  -- declaracao das entradas
z, h, k, w, v,m,p,x: out bit -- declaracao das saidas
);
end entity ;
-- ------------------------------------------
architecture projeto1 of projeto1 is -- codigo do programa
begin
--z esta recebendo a saida da operacao "a" and "b"
z <= a and b;
--v esta recebendo a saida da operacao negacao "a"
v <= not a;
--p esta recebendo a saida da operacao negacao "b"
p <= not b;
--x esta recebendo a saida da operacao "a" or "b"
x <= a or b;
--w esta recebendo a saida da operacao "a" negacao and "b"
w <= a nand b;
--h esta recebendo a saida da operacao "a" negacao or "b"
h <= a nor b;
--m esta recebendo a saida da operacao "a" or exclusivo "b"
m <= a xor b;
```

```
--k esta recebendo a saida da operacao "a" or exclusivo negado "b"
k <= a xnor b;

end architecture ;
-- ---------------------------------------------------------------
```

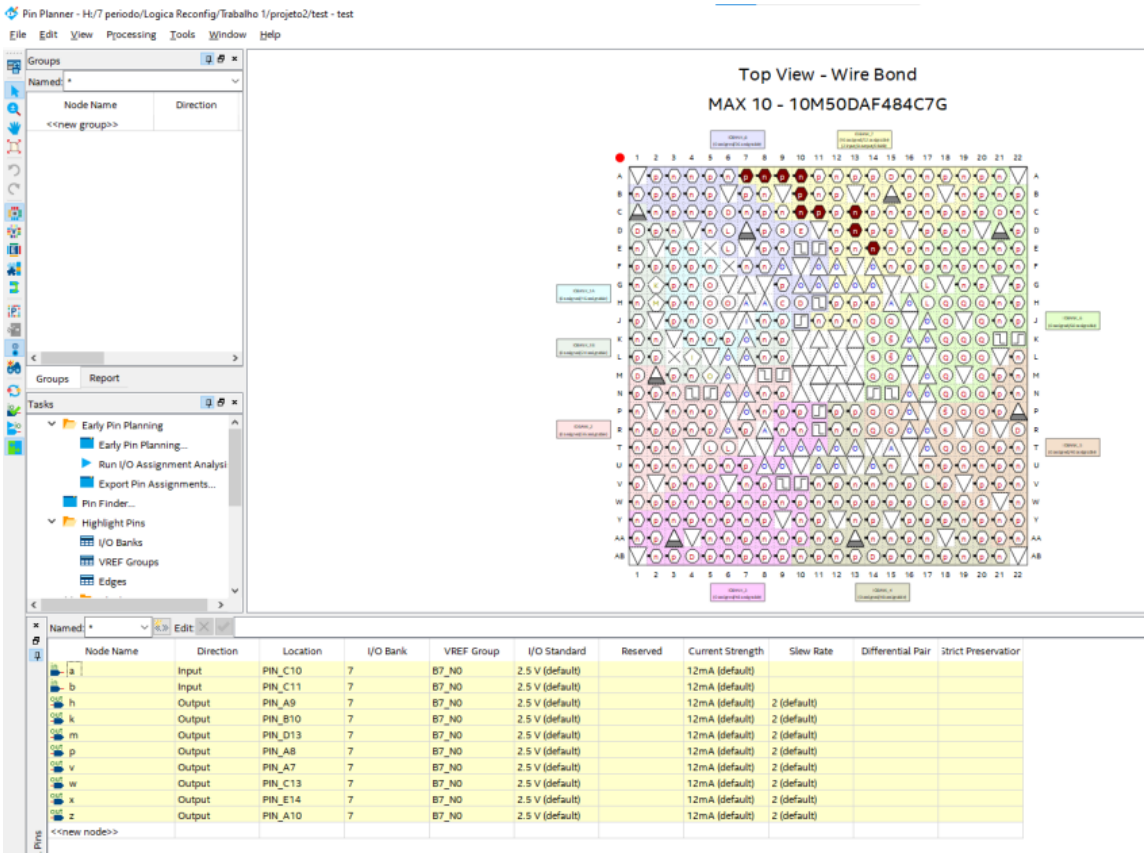

Figura 1: Pin planner

Para implementar o código na placa DE10-Lite, é necessário configurar os pinos por meio do pin planner, a cofiguração dos LEDs de acordo com o solicitado no trabalho está evidenciado na Figura 1.

2.2 Parte 2

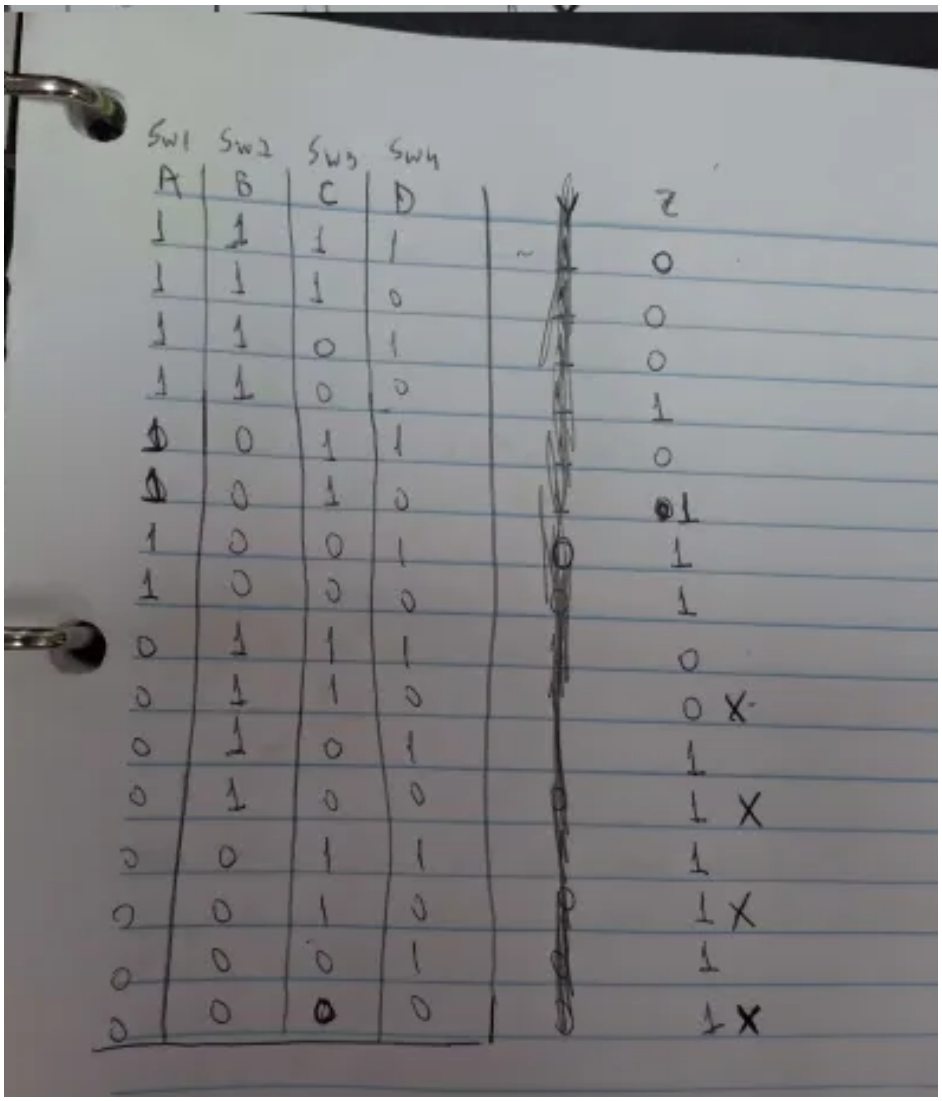

Figura 2: Tabela verdade

Para o circuito da parte 2 inicialmente foi feita a tabela verdade do circuito e o mapa de Karnaugh, onde na saída "z"é colocado alto caso duas ou mais chaves estejam em fechadas ao mesmo tempo (em 0), e para o caso das chaves "sw1"e "sw4"estarem ao mesmo tempo é colocado um "x", ou seja , a condição de "don't care", por fim ao analisar o mapa de karnaugh foi optado que todos os "x"fossem 1, pois assim foi possível agrupar um maior número de 1 e deixar a equação final mais simples possível.

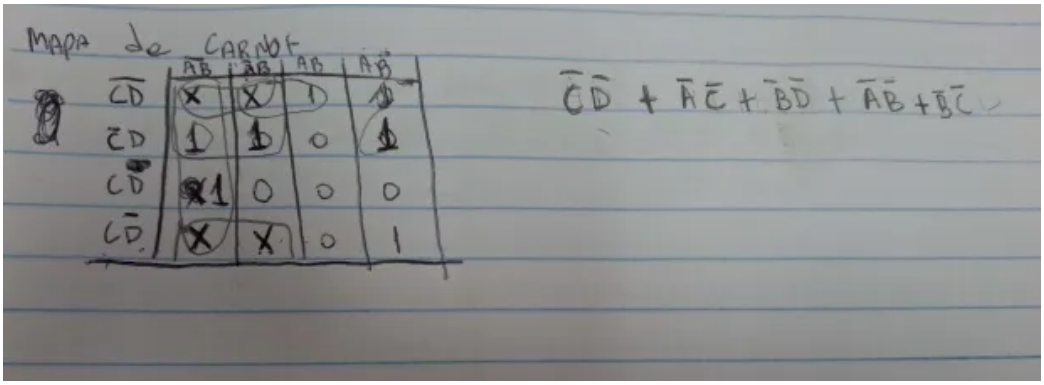

Figura 3: Mapa de Karnaugh

```
-- ------------------------------------------------
library ieee ;  -- Importando a biblioteca ieee
use ieee . std_logic_1164 . all ;
-- ------------------------------------------------
entity test is
port (
a , b, c ,d: in bit ;
z: out bit
);
end entity ;
-- ------------------------------------------------
architecture test of test is
begin
z <= (not c and not d) or (not a  and not c) or (not b and not d)...
or (not a and not b) or (not b and not c);

end architecture ;
-- ------------------------------------------------
```

As portas de entrada foram declaradas (a,b,c,d) e a saída (z), logo depois no begin foi adicionada a equação obtida pela tabela verdade e o mapa de Karnaugh.

3 Resultados obtidos

A equação gera o diagrama RTL da figura 4 e ao colocar entradas aleatórias é possivel verificar a saída por meio da simulação mostrada na figura 5.

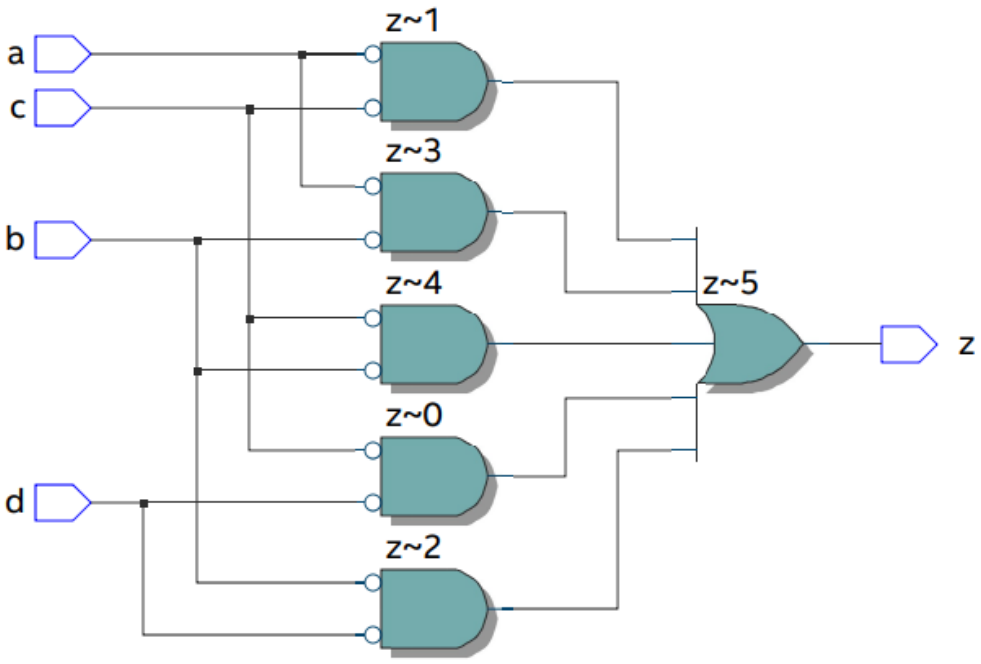

Figura 4: Diagragrama RTL

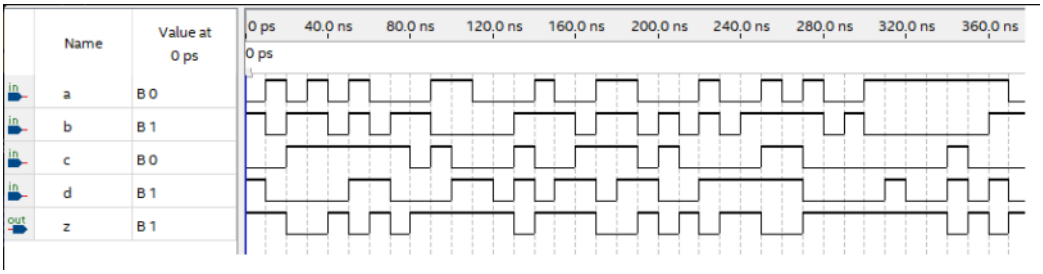

Figura 5: Simulação