Caminho Crítico dos Operadores

Dispositivos Lógicos Programáveis II - Prática

Gabriel Luiz Espindola Pedro

12 de Março de 2023

Sumário

1. Atividade

Faça a implementação de um somador para valores de 16 bits sem sinal. Escreva 5 implementações em VHDL para as operações abaixo. Sintetize usando o Quartus e o dispositivo da DE2-115.

a) a+b
b) a + “0000000000000001”
c) a + “0000000010000000”
d) a + “1000000000000000”
e) a + “1010101010101010”

Verifique a área (LE) e atraso (ns) para cada implementação.

1.1. Discussão:

Qual implementação usou menos área? Por quê? Como o delay se comportou? Comente as diferenças entre os valores de área e delay obtidos nas operações a), b) e c).

2. Desenvolvimento

2.1. Código VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity u_adder_16b is
    port (
        a    : in  std_logic_vector(15 downto 0);
        b    : in  std_logic_vector(15 downto 0);

        s    : out  std_logic_vector(15 downto 0)
    );
end entity;

architecture v1 of u_adder_16b is
begin

    s <= std_logic_vector(unsigned(a) + unsigned(b));

    -- s <= std_logic_vector(unsigned(a) + "0000000000000001");

    -- s <= std_logic_vector(unsigned(a) + "0000000010000000");

    -- s <= std_logic_vector(unsigned(a) + "1000000000000000");

    -- s <= std_logic_vector(unsigned(a) + "1010101010101010");
end v1;
```

2.2. Resultados das simulações

	Metodologia	Área (LE)	Delay (ns)
a	in_a + in_b	16	1,932
b	in_a + “0000000000000001”	15	1,826
c	in_a + “0000000010000000”	8	1,404
d	in_a + “1000000000000000”	0	0
e	in_a + “1010101010101010”	14	1,800

2.3. Análise dos Resultados

Ao realizar os experimentos e compararmos os resultados verificamos que a implementação que mais distoa das demais é a implementação **d**, que utiliza a constante “1000000000000000”. Esta implementação não utiliza nenhum LE e não possui atraso. Isso ocorre pois a constante é uma potência de 2, o que permite que o somador seja simplificado para um simples deslocamento de bits.

Como contraponto podemos verificar que a implementação **b** é a mais custosa em termos de LE e atraso. Isso se dá pela constante “0000000000000001” que é a menor possível para um número de 16 bits. Isso faz com que o somador tenha que realizar a maior quantidade de operações possíveis para realizar a soma.

Já as implementações **c** e **e** possuem um custo intermediário, com a implementação **c** utilizando a constante “0000000010000000” e a implementação **e** utilizando a constante “1010101010101010”. Ambas as constantes possuem um número intermediário de bits em 1.