Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

Лабораторная работа №3

«Описание и моделирование регулярных (систолических) схем»

Вариант 6

Выполнила студент Проверил:

группы 450701 Бибило П.Н.

Никитин Г.Ю.

Минск, 2016

1. Задание:

1. Ввести имена входов и выходов схемы с использованием типа BIT\_VECTOR.

2. Составить две VHDL-модели.

Способ 1. Иерархическое описание для N=4 (рис. 1).

1-й (верхний - головной) уровень иерархии. Описание схемы в целом состоит из двух компонент, выделенных штриховой линией.

2-й уровень иерархии. Описание подсхемы, выделенной штриховой линией, состоит из четырех подсхем.

3-й (нижний – листовой) уровень иерархии. Описание подсхем, входящих в описание 2-го уровня иерархии, является структурным описанием (состоит из логических элементов и их связей), либо является функциональным описанием.

Способ 2. Описать схему (рис. 1) с использованием операторов generate, generic для произвольной разрядности N.

3. Составить тест и провести моделирование для N=4 (для способов 1,2).

1. Логическая схема (рис. 1).



Рис. 1 – Логическая схема (Вариант 6)

1. VHDL-модель логической схемы:

Файлы, общие для двух способов:

Файл sum1

entity sum1 is

port(a, b:in bit\_vector(1 to 2);

c: out bit\_vector(1 to 3));

end sum1;

architecture beh of sum1 is

begin

c(3)<=(b(2) and a(2)) or

(b(1) and a(2) and a(1)) or

(a(1) and b(2) and b(1));

c(2)<=( a(2) and not b(2) and not b(1)) or

( a(2) and not a(1) and not b(2)) or

(not a(2) and a(1) and not b(2) and b(1)) or

( a(2) and a(1) and b(2) and b(1)) or

(not a(2) and not a(1) and b(2)) or

(not a(2) and b(2) and not b(1));

c(1)<=( a(1) and not b(2) and not b(1)) or

( a(2) and not a(1) and b(1)) or

( a(1) and b(2) and not b(1)) or

(not a(2) and not a(1) and b(1));

end;

Файл umn

entity umn is

port(a, b:in bit\_vector(1 to 2);

c: out bit\_vector(1 to 4));

end umn;

architecture beh of umn is

begin

c(4)<= a(2) and a(1) and b(2) and b(1);

c(3)<= a(2) and b(2) and (a(1) nand b(1));

c(2)<=( a(2) and b(1) and ((not b(1)) nand b(2)) ) or

( a(1) and b(2) and ((not b(2)) nand b(1)) );

c(1)<=a(1) and b(1);

end;

Файл sx1

entity sx1 is

port(x: in bit\_vector(1 to 8);

z: out bit\_vector(1 to 6));

end sx1;

architecture str of sx1 is

component sum1

port(a, b:in bit\_vector(1 to 2);

c: out bit\_vector(1 to 3));

end component;

component umn

port(a, b:in bit\_vector(1 to 2);

c: out bit\_vector(1 to 4));

end component;

signal s: bit\_vector(1 to 4);

begin

m1: sum1 port map

(a(2)=>x(1),

a(1)=>x(2),

b(1)=>x(4),

b(2)=>x(3),

c(3)=>s(3),

c(2)=>s(2),

c(1)=>s(1));

m2: sum1 port map

(a(2)=>s(2),

a(1)=>s(3),

b(1)=>x(6),

b(2)=>x(5),

c(3)=>z(1),

c(2)=>z(2),

c(1)=>s(4));

m3: umn port map

(a(2)=>s(1),

a(1)=>x(7),

b(2)=>s(4),

b(1)=>x(8),

c(4)=>z(6),

c(3)=>z(5),

c(2)=>z(4),

c(1)=>z(3));

end;

Файлы первого способа:

Файл sx2p1

entity sx2p1 is

port(x: in bit\_vector(1 to 14);

y: out bit\_vector(1 to 10));

end sx2p1;

architecture str of sx2p1 is

component sx1

port(x: in bit\_vector(1 to 8);

z: out bit\_vector(1 to 6));

end component;

signal s: bit\_vector(1 to 2);

begin

m1: sx1 port map

(x(1)=>x(1),

x(2)=>x(2),

x(3)=>x(3),

x(4)=>x(4),

x(5)=>x(5),

x(6)=>x(6),

x(7)=>x(7),

x(8)=>x(8),

z(1)=>s(1),

z(2)=>s(2),

z(3)=>y(10),

z(4)=>y(9),

z(5)=>y(8),

z(6)=>y(7));

m2: sx1 port map

(x(1)=>s(1),

x(2)=>s(2),

x(3)=>x(9),

x(4)=>x(10),

x(5)=>x(11),

x(6)=>x(12),

x(7)=>x(13),

x(8)=>x(14),

z(1)=>y(1),

z(2)=>y(2),

z(3)=>y(3),

z(4)=>y(4),

z(5)=>y(5),

z(6)=>y(6));

end;

Файл test1

entity test1 is

end test1;

architecture atest of test1 is

component sx2p1

port(x: in bit\_vector(1 to 14);

y: out bit\_vector(1 to 10));

end component;

signal x:bit\_vector(1 to 14);

signal y:bit\_vector(1 to 10);

begin

p1: sx2p1 port map

(x(1)=>x(1),

x(2)=>x(2),

x(3)=>x(3),

x(4)=>x(4),

x(5)=>x(5),

x(6)=>x(6),

x(7)=>x(7),

x(8)=>x(8),

x(9)=>x(9),

x(10)=>x(10),

x(11)=>x(11),

x(12)=>x(12),

x(13)=>x(13),

x(14)=>x(14),

y(1)=>y(1),

y(2)=>y(2),

y(3)=>y(3),

y(4)=>y(4),

y(5)=>y(5),

y(6)=>y(6),

y(7)=>y(7),

y(8)=>y(8),

y(9)=>y(9),

y(10)=>y(10));

x<="00000000000000",

"00100000000001" after 100 ns,

"01000000000011" after 200 ns,

"10000000000101" after 300 ns,

"01000000001001" after 400 ns,

"00100000010001" after 500 ns,

"00010000100001" after 600 ns,

"00001001000001" after 700 ns,

"00000110000001" after 800 ns,

"00000110000001" after 900 ns,

"00001001000001" after 1000 ns,

"00010000100001" after 1100 ns,

"00100000010001" after 1200 ns,

"01000000001001" after 1300 ns,

"10000000000101" after 1400 ns,

"11111111111111" after 1500 ns;

end atest;

Файлы второго способа:

Файл sx2p2

library work;

use work.rr.all;

entity sx2p2 is

generic(N:natural);

port(x: in bit\_vector(1 to 6\*n+2);

y: out bit\_vector(1 to 4\*n+2));

end sx2p2;

architecture str of sx2p2 is

component sx1

port(x: in bit\_vector(1 to 8);

z: out bit\_vector(1 to 6));

end component;

signal s: bit\_vector(1 to 2\*n+2);

begin

g4: for i in 1 to n generate

m1: sx1 port map

(x(1)=>s(2\*i),

x(2)=>s(2\*i-1),

x(3)=>x(6\*i-3),

x(4)=>x(6\*i-2),

x(5)=>x(6\*i-1),

x(6)=>x(6\*i),

x(7)=>x(6\*i+1),

x(8)=>x(6\*i+2),

z(1)=>s(2\*i+1),

z(2)=>s(2\*i+2),

z(6)=>y(4\*i-3),

z(5)=>y(4\*i-2),

z(4)=>y(4\*i-1),

z(3)=>y(4\*i));

end generate g4;

s(1)<=x(1);

s(2)<=x(2);

y(4\*n+1)<=s(2\*n+2);

y(4\*n+2)<=s(2\*n+1);

end;

Файл test2

library work;

use work.rr.all;

entity test2 is

end test2;

architecture atest of test2 is

component sx2p2

generic(N:natural);

port(x: in bit\_vector(1 to 6\*N+2);

y: out bit\_vector(1 to 4\*N+2));

end component;

signal x:bit\_vector(1 to 6\*N+2);

signal y:bit\_vector(1 to 4\*N+2);

begin

p1: sx2p2

generic map(N)

port map

(x(1)=>x(1),

x(2)=>x(2),

x(3)=>x(3),

x(4)=>x(4),

x(5)=>x(5),

x(6)=>x(6),

x(7)=>x(7),

x(8)=>x(8),

x(9)=>x(9),

x(10)=>x(10),

x(11)=>x(11),

x(12)=>x(12),

x(13)=>x(13),

x(14)=>x(14),

y(1)=>y(1),

y(2)=>y(2),

y(3)=>y(3),

y(4)=>y(4),

y(5)=>y(5),

y(6)=>y(6),

y(7)=>y(7),

y(8)=>y(8),

y(9)=>y(9),

y(10)=>y(10));

x<="00000000000000",

"00100000000001" after 100 ns,

"01000000000011" after 200 ns,

"10000000000101" after 300 ns,

"01000000001001" after 400 ns,

"00100000010001" after 500 ns,

"00010000100001" after 600 ns,

"00001001000001" after 700 ns,

"00000110000001" after 800 ns,

"00000110000001" after 900 ns,

"00001001000001" after 1000 ns,

"00010000100001" after 1100 ns,

"00100000010001" after 1200 ns,

"01000000001001" after 1300 ns,

"10000000000101" after 1400 ns,

"11111111111111" after 1500 ns;

end atest;

Файл test2

package rr is

constant N:natural:=2;

end rr;

В итоге у нас получились следующие временные диаграммы (Рис. 2 и 3).

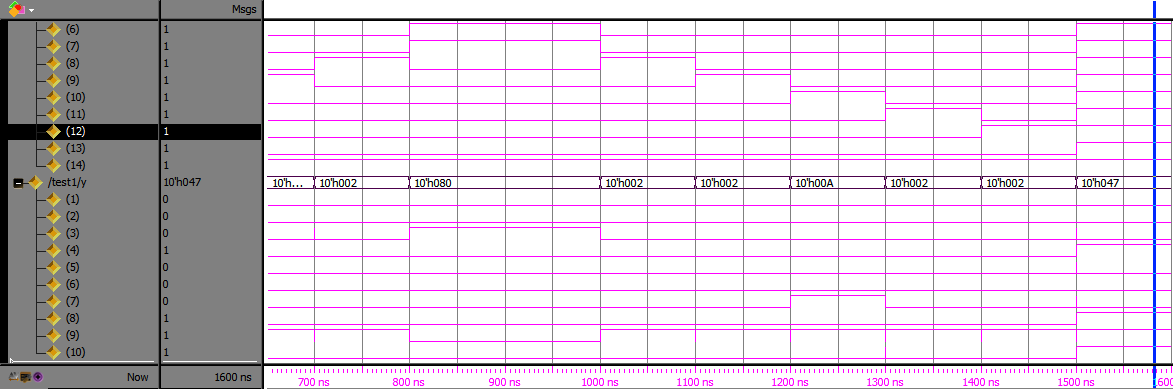


Рис. 2 – Временная диаграмма для способа №1

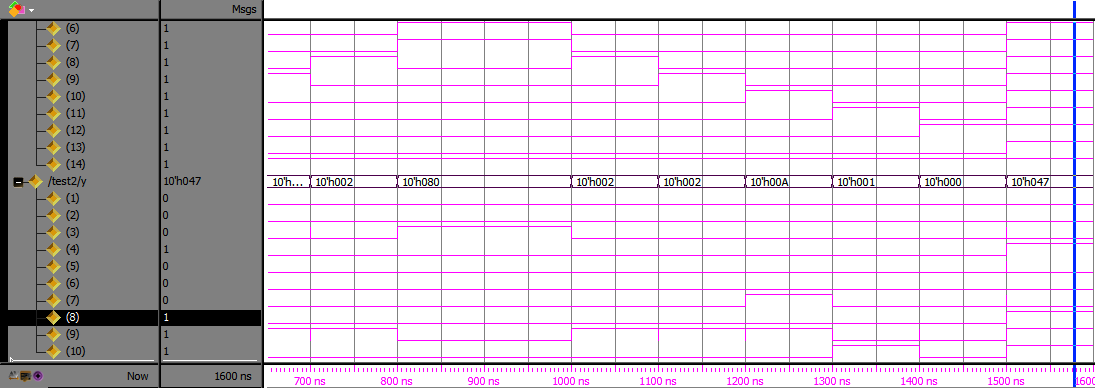


Рис. 3 – Временная диаграмма для способа №2

Как видно из приведённых выше временных диаграмм, выходные сигналы совпадают. В связи с этим можно сделать вывод, что схема была реализована правильно.

3.Анализ результатов и выводы.

Мы провели моделирование заданной логической схемы в среде Modelsim SE и успешно выполнили выданное задание. В результате была получена верная временная диаграмма работы устройства на 16 наборах входных слов.