Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

Дисциплина: Проектирование цифровых систем на языках описания аппаратуры

ОТЧЕТ

к лабораторной работе №3

на тему

ОПИСАНИЕ И МОДЕЛИРОВАНИЕ РЕГУЛЯРНЫХ ЛОГИЧЕСКИХ СИСТЕМ

|  |  |
| --- | --- |
| Выполнили:  ст. гр. 850702  Турко В. Д. | Проверил:  Санько Н. С. |
|  |  |

Минск 2020

1. Цель работы

Для заданной нерегулярной логической схемы (рис. 1.1): составить иерархическое описание для заданной схемы, описать схему с использованием операторов generate, generic для произвольной разрядности N.

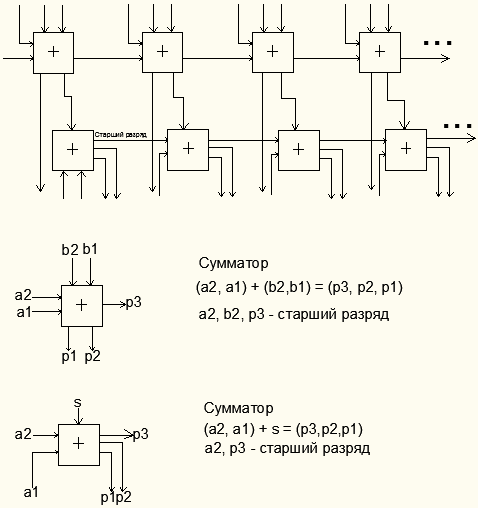


Рисунок 1.1 – Логическая схема

1. Структурная vhdl-модель схемы
2. Используемые элементы

В таблице 2.1 приведена реализация моделей используемых элементов.

Таблица 2.1 – модели элементов

|  |  |
| --- | --- |
| ENTITY add1 IS      PORT (          a1, b1 : IN STD\_LOGIC;          c1, p1 : OUT STD\_LOGIC);  END add1; | ENTITY add2 IS      PORT (          c1, a2, b2 : IN STD\_LOGIC;          p2, p3 : OUT STD\_LOGIC);  END add2; |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| ARCHITECTURE struct\_1 OF add1 IS  BEGIN      p1 <= ((a1 AND (NOT b1)) OR ((NOT a1) AND b1));      c1 <= a1 AND b1;  END struct\_1; | ARCHITECTURE struct\_1 OF add2 IS  BEGIN      p2 <= ((NOT c1) AND (NOT a2) AND b2) OR          ((NOT c1) AND a2 AND (NOT b2)) OR          (c1 AND (NOT a2) AND (NOT b2)) OR          (a2 AND b2 AND c1);      p3 <= (a2 AND c1) OR (b2 AND c1) OR (a2 AND b2);  END struct\_1; |
| ENTITY adder\_2 IS      PORT (          a2, a1 : IN STD\_LOGIC;          b : IN STD\_LOGIC\_VECTOR(2 DOWNTO 1);          p : OUT STD\_LOGIC\_VECTOR(3 DOWNTO 1));  END adder\_2;  ARCHITECTURE structure OF adder\_2 IS      COMPONENT add1          PORT (              a1, b1 : IN STD\_LOGIC;              c1, p1 : OUT STD\_LOGIC);      END COMPONENT;      COMPONENT add2          PORT (              c1, a2, b2 : IN STD\_LOGIC;              p2, p3 : OUT STD\_LOGIC);      END COMPONENT;      SIGNAL c1 : STD\_LOGIC;  BEGIN      circ1 : add1 PORT MAP(a1, b(1), c1, p(1));      circ2 : add2 PORT MAP(c1, a2, b(2), p(2), p(3));  END structure; | ENTITY adder\_2s IS      PORT (          a2, a1 : IN STD\_LOGIC;          s : IN STD\_LOGIC;          p : OUT STD\_LOGIC\_VECTOR(3 DOWNTO 1));  END adder\_2s;  ARCHITECTURE structure OF adder\_2s IS      COMPONENT add1          PORT (              a1, b1 : IN STD\_LOGIC;              c1, p1 : OUT STD\_LOGIC);      END COMPONENT;      COMPONENT add2          PORT (              c1, a2, b2 : IN STD\_LOGIC;              p2, p3 : OUT STD\_LOGIC);      END COMPONENT;      SIGNAL c1 : STD\_LOGIC;  BEGIN      circ1 : add1      PORT MAP(a1, '0', c1, p(1));      circ2 : add2      PORT MAP(c1, a2, s, p(2), p(3));  END structure; |

1. Иерархическое описание схемы

ENTITY SchemeModel IS

    PORT (

        x1 : IN STD\_LOGIC\_VECTOR(1 TO 4);

        x2 : IN STD\_LOGIC\_VECTOR(1 TO 4);

        x3 : IN STD\_LOGIC\_VECTOR(1 TO 4);

        x4 : IN STD\_LOGIC\_VECTOR(1 TO 4);

        y1 : OUT STD\_LOGIC\_VECTOR(1 TO 3);

        y2 : OUT STD\_LOGIC\_VECTOR(1 TO 3);

        y3 : OUT STD\_LOGIC\_VECTOR(1 TO 3);

        y4 : OUT STD\_LOGIC\_VECTOR(1 TO 3);

        p : OUT STD\_LOGIC\_VECTOR(0 TO 1));

END SchemeModel;

ARCHITECTURE SchemeArch OF SchemeModel IS

    SIGNAL adder1\_1\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder1\_2\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder1\_3\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder1\_4\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder2\_1\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder2\_2\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder2\_3\_out : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL adder2\_4\_out : STD\_LOGIC\_VECTOR(1 TO 3);

BEGIN

    adder1\_1 : adder\_2 PORT MAP(x1(1), '0', x1(2 TO 3), adder1\_1\_out);

    adder1\_2 : adder\_2 PORT MAP(x2(1), adder1\_1\_out(1), x2(2 TO 3), adder1\_2\_out);

    adder1\_3 : adder\_2 PORT MAP(x3(1), adder1\_2\_out(1), x3(2 TO 3), adder1\_3\_out);

    adder1\_4 : adder\_2 PORT MAP(x4(1), adder1\_3\_out(1), x4(2 TO 3), adder1\_4\_out);

    adder2\_1 : adder\_2s PORT MAP('0', x1(4), adder1\_1\_out(2), adder2\_1\_out);

    adder2\_2 : adder\_2s PORT MAP(adder2\_1\_out(1), x2(4), adder1\_2\_out(2), adder2\_2\_out);

    adder2\_3 : adder\_2s PORT MAP(adder2\_2\_out(1), x3(4), adder1\_3\_out(2), adder2\_3\_out);

    adder2\_4 : adder\_2s PORT MAP(adder2\_3\_out(1), x4(4), adder1\_4\_out(2), adder2\_4\_out);

    y1 <= (adder1\_1\_out(3), adder2\_1\_out(2), adder2\_1\_out(3));

    y2 <= (adder1\_2\_out(3), adder2\_2\_out(2), adder2\_2\_out(3));

    y3 <= (adder1\_3\_out(3), adder2\_3\_out(2), adder2\_3\_out(3));

    y4 <= (adder1\_4\_out(3), adder2\_4\_out(2), adder2\_4\_out(3));

    p <= (adder1\_4\_out(1), adder2\_4\_out(1));

END ARCHITECTURE SchemeArch;

1. Описание с использованием операторов generate, generic

ENTITY SchemeModelGen IS

    GENERIC (N : POSITIVE := 4);

    PORT (

        x : IN STD\_LOGIC\_VECTOR(1 TO 4 \* N);

        y : OUT STD\_LOGIC\_VECTOR(1 TO 3 \* N);

        p : OUT STD\_LOGIC\_VECTOR(0 TO 1));

END SchemeModelGen;

ARCHITECTURE SchemeArch OF SchemeModelGen IS

    SIGNAL adder1\_out : STD\_LOGIC\_VECTOR(1 TO 3 \* N);

    SIGNAL adder2\_out : STD\_LOGIC\_VECTOR(1 TO 3 \* N);

BEGIN

    GEN\_ADD : FOR I IN 0 TO N - 1 GENERATE

        LOWER\_BIT : IF I = 0 GENERATE

            U11 : adder\_2 PORT MAP(x(1), '0', x(2 TO 3), adder1\_out(1 to 3));

            U12 : adder\_2s PORT MAP('0', x(4), adder1\_out(2), adder2\_out(1 to 3));

        END GENERATE LOWER\_BIT;

        UPPER\_BITS : IF I > 0 GENERATE

            U11 : adder\_2 PORT MAP(x(I \* 4 + 1), adder1\_out(I \* 3 + 1), x(I \* 4 + 2 TO I \* 4 + 3),

adder1\_out(I \* 3 + 1 to I \* 3 + 3));

            U12 : adder\_2s PORT MAP(adder2\_out(I \* 3 + 1), x(I \* 4 + 4), adder1\_out(I \* 3 + 2),

adder2\_out(I \* 3 + 1 to I \* 3 + 3));

        END GENERATE UPPER\_BITS;

        y(I \* 3 + 1 TO I \* 3 + 3) <= (adder1\_out(I \* 3 + 3), adder2\_out(I \* 3 + 2),

adder2\_out(I \* 3 + 3));

    END GENERATE GEN\_ADD;

    p <= (adder1\_out((N - 1) \* 3 + 1), adder2\_out((N - 1) \* 3 + 1));

END ARCHITECTURE SchemeArch;

1. Моделирование
2. Тестирующая программа

ENTITY scheme\_test IS

END scheme\_test;

ARCHITECTURE MainBehavior OF scheme\_test IS

    COMPONENT SchemeModel IS

        PORT (

            x1 : IN STD\_LOGIC\_VECTOR(0 TO 3);

            x2 : IN STD\_LOGIC\_VECTOR(0 TO 3);

            x3 : IN STD\_LOGIC\_VECTOR(0 TO 3);

            x4 : IN STD\_LOGIC\_VECTOR(0 TO 3);

            y1 : OUT STD\_LOGIC\_VECTOR(0 TO 2);

            y2 : OUT STD\_LOGIC\_VECTOR(0 TO 2);

            y3 : OUT STD\_LOGIC\_VECTOR(0 TO 2);

            y4 : OUT STD\_LOGIC\_VECTOR(0 TO 2);

            p : OUT STD\_LOGIC\_VECTOR(0 TO 1));

    END COMPONENT SchemeModel;

    COMPONENT SchemeModelGen IS

        GENERIC (N : POSITIVE := 4);

        PORT (

            x : IN STD\_LOGIC\_VECTOR(1 TO 4 \* N);

            y : OUT STD\_LOGIC\_VECTOR(1 TO 3 \* N);

            p : OUT STD\_LOGIC\_VECTOR(0 TO 1));

    END COMPONENT SchemeModelGen;

    CONSTANT COUNT : INTEGER := 4;

    SIGNAL x1 : STD\_LOGIC\_VECTOR(1 TO 4) := (OTHERS => '0');

    SIGNAL x2 : STD\_LOGIC\_VECTOR(1 TO 4) := (OTHERS => '0');

    SIGNAL x3 : STD\_LOGIC\_VECTOR(1 TO 4) := (OTHERS => '0');

    SIGNAL x4 : STD\_LOGIC\_VECTOR(1 TO 4) := (OTHERS => '0');

    SIGNAL y1 : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y2 : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y3 : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y4 : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL p : STD\_LOGIC\_VECTOR(0 TO 1);

    SIGNAL x\_gen : STD\_LOGIC\_VECTOR(1 TO COUNT \* 4);

    SIGNAL y\_gen : STD\_LOGIC\_VECTOR(1 TO COUNT \* 3);

    SIGNAL p\_gen : STD\_LOGIC\_VECTOR(0 TO 1);

    SIGNAL y1\_gen\_diag : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y2\_gen\_diag : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y3\_gen\_diag : STD\_LOGIC\_VECTOR(1 TO 3);

    SIGNAL y4\_gen\_diag : STD\_LOGIC\_VECTOR(1 TO 3);

BEGIN

    portm1 : SchemeModel PORT MAP(x1, x2, x3, x4, y1, y2, y3, y4, p);

    portm2 : SchemeModelGen GENERIC MAP(N => 4) PORT MAP(x\_gen, y\_gen, p\_gen);

    x1 <= NOT x1 AFTER interval;

    x2 <= NOT x2 AFTER interval \* 2;

    x3 <= NOT x3 AFTER interval \* 4;

    x4 <= NOT x4 AFTER interval \* 8;

    x\_gen(1 to 4) <= x1;

    x\_gen(5 to 8) <= x2;

    x\_gen(9 to 12) <= x3;

    x\_gen(13 to 16) <= x4;

    y1\_gen\_diag <= y\_gen(1 to 3);

    y2\_gen\_diag <= y\_gen(4 to 6);

    y3\_gen\_diag <= y\_gen(7 to 9);

    y4\_gen\_diag <= y\_gen(10 to 12);

    PROCESS IS

        VARIABLE count : INTEGER := 0;

    BEGIN

        IF count > 16 THEN stop;

        ELSE

            WAIT FOR interval;

            count := count + 1 ;

        END IF;

    END PROCESS;

END MainBehavior;

1. Временная диаграмма

В результате моделирования получим временную диаграмму, представленную на рис. 3.1.

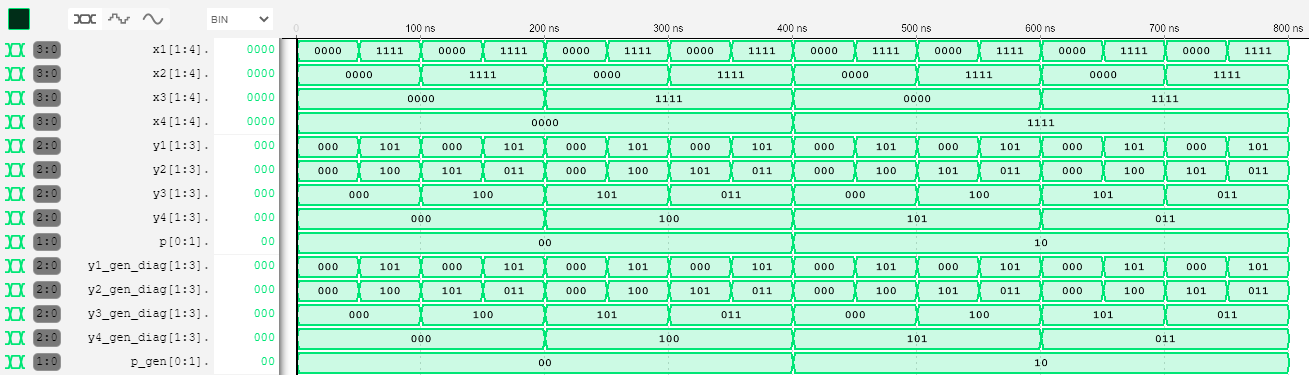


Рисунок 3.1 – Временная диаграмма

1. вывод

При выполнении лабораторной работы я составил две VHDL-модели для заданной схемы: иерархическое описание и описание с использованием операторов generate, generic. А также составил тестирующую программу и провел моделирование VHDL-моделей для N=4.