# 1 Задание

Цифровая схема нарисованная в draw.io.

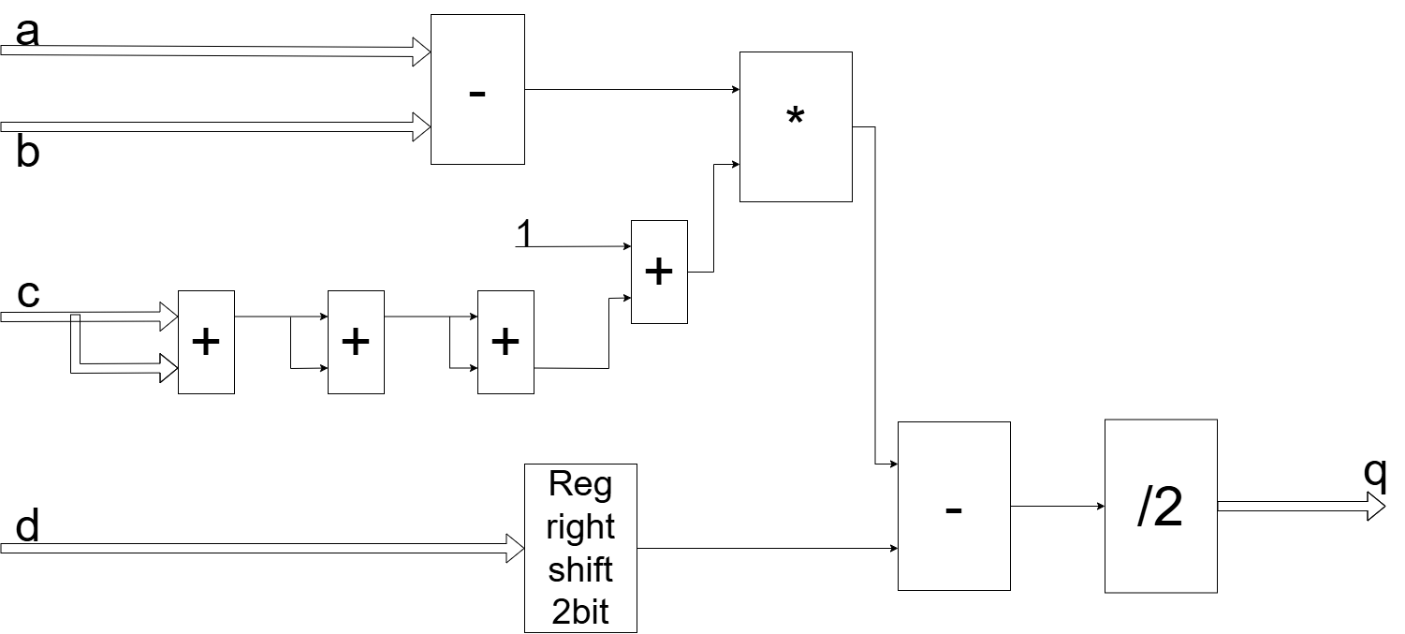


Рисунок – цифровая схема нарисованная в draw.io

Сдвиг на 2 бита – замена умножения на 4

# 2 Задание

Плис – Artix 7, xc7a25tcsg325-2L. Частота - 100 мГЦ

## Приложение – содержание main.vhd

### library IEEE;

### use IEEE.STD\_LOGIC\_1164.ALL;

### use ieee.numeric\_std.all;

### entity main is

### Generic

### (

### bit\_width : integer := 10

### );

### Port

### (

### clk : in STD\_LOGIC;

### a,b,c,d :in signed(bit\_width-1 downto 0);

### q : out signed(bit\_width-1 downto 0)

### );

### end main;

### architecture Behavioral of main is

### signal part\_1 : signed(bit\_width downto 0);

### signal part\_2 : signed(bit\_width+1 downto 0);

### signal part\_3 : signed(bit\_width\*2+2 downto 0);

### signal part\_4 : signed(bit\_width\*2+2 downto 0);

### signal part\_5 : signed(bit\_width\*2+2 downto 0);

### signal upper\_part : signed(bit\_width\*2+2 downto 0);

### signal q\_reg : signed(bit\_width\*2+2 downto 0);

### begin

### process (clk)

### begin

### if rising\_edge(clk) then

### part\_1 <= resize(a, bit\_width+1) - resize(b, bit\_width+1);

### part\_2 <= resize(to\_signed(1, 2), bit\_width+2) +

### resize(c & '0' + c, bit\_width+2);

### part\_3 <= part\_1 \* part\_2;

### part\_4 <= part\_3 - resize(d & "00", bit\_width\*2+3);

### q\_reg <= part\_4 / 2;

### end if;

### end process;

### q <= resize(q\_reg, bit\_width);

### end Behavioral;

## Приложение – содержание model.vhd

### LIBRARY ieee;

### USE ieee.std\_logic\_1164.ALL;

### USE ieee.numeric\_std.ALL;

### LIBRARY UNISIM;

### USE UNISIM.Vcomponents.ALL;

### ENTITY testbench IS

### END testbench;

### ARCHITECTURE behavorial OF

### testbench IS

### constant bit\_width : integer := 10;

### COMPONENT main

### Port

### (

### clk : in STD\_LOGIC;

### a,b,c,d :in signed(bit\_width-1 downto 0);

### q : out signed(bit\_width-1 downto 0)

### );

### END COMPONENT;

### signal clk :std\_logic;

### signal a : signed(bit\_width-1 downto 0);

### signal b : signed(bit\_width-1 downto 0);

### signal c : signed(bit\_width-1 downto 0);

### signal d : signed(bit\_width-1 downto 0);

### signal q : signed(bit\_width-1 downto 0);

### constant PERIOD : TIME := 10 ns;

### constant DUTY\_CYCLE : real := 0.5;

### constant OFFSET : time := 3 ns;

### BEGIN

### UUT: main

### PORT MAP

### (

### clk => clk,

### a => a,

### b => b,

### c => c,

### d => d,

### q => q

### );

### clk\_gen: PROCESS

### BEGIN

### WAIT for OFFSET;

### CLOCK\_LOOP : LOOP

### clk <= '1';

### WAIT FOR (PERIOD - (PERIOD \* DUTY\_CYCLE));

### clk <= '0';

### WAIT FOR (PERIOD \* DUTY\_CYCLE);

### END LOOP CLOCK\_LOOP;

### END PROCESS;

abcd\_gen : PROCESS

BEGIN

wait for PERIOD\*15;

wait until rising\_edge(clk);

a <= to\_signed(50, bit\_width);

b <= to\_signed(45, bit\_width);

c <= to\_signed(25, bit\_width);

d <= to\_signed(10, bit\_width);

wait for PERIOD\*8;

wait until rising\_edge(clk);

a <= to\_signed(15, bit\_width);

b <= to\_signed(22, bit\_width);

c <= to\_signed(3, bit\_width);

d <= to\_signed(11, bit\_width);

wait for PERIOD\*8;

wait until rising\_edge(clk);

a <= to\_signed(23, bit\_width);

b <= to\_signed(5, bit\_width);

c <= to\_signed(4, bit\_width);

d <= to\_signed(10, bit\_width);

wait for PERIOD\*8;

wait until rising\_edge(clk);

a <= to\_signed(10, bit\_width);

b <= to\_signed(40, bit\_width);

c <= to\_signed(4, bit\_width);

d <= to\_signed(110, bit\_width);

wait for PERIOD\*8;

wait until rising\_edge(clk);

a <= to\_signed(-10, bit\_width);

b <= to\_signed(41, bit\_width);

c <= to\_signed(-4, bit\_width);

d <= to\_signed(100, bit\_width);

wait for PERIOD\*8;

wait until rising\_edge(clk);

a <= to\_signed(-11, bit\_width);

b <= to\_signed(100, bit\_width);

c <= to\_signed(-6, bit\_width);

d <= to\_signed(110, bit\_width);

WAIT;

END PROCESS;

END;

# Требования:

“c. входные параметры *a, b, c, d* являются целыми числами со знаком (signed)”

За это отвечает выделенная строчка в файле main.vhd. Так как параметры signed, выделяется -1 от заданной разрядности(bit\_width).

### entity main is

### Generic

### (

### bit\_width : integer := 10

### );

### Port

### (

### clk : in STD\_LOGIC;

### a,b,c,d :in signed(bit\_width-1 downto 0);

### q : out signed(bit\_width-1 downto 0)

### );

### end main;

“d. набор параметров a, b, c, d должен подаваться на вход схемы синхронно “

В main.vhd все операции запускаются по фронту clk :

### process(clk)

### begin

### if rising\_edge(clk) then

В model.vhd в abcd\_gen происходит синхронизация с фронтом:

### abcd\_gen : PROCESS

### BEGIN

### wait for PERIOD\*15;

### wait until rising\_edge(clk);

“e. разрядность данных должна определяться параметром”

за это отвечает generic port bit\_width в main.vhd:

### entity main is

### Generic

### (

### bit\_width : integer := 10

### );

А в model.vhd за это отвечает постоянная:

### constant bit\_width : integer := 10;

“f. схема должна обеспечивать возможность получения нового набора входных параметров a, b, c, d каждый такт.”

Это происходит в model.vhd в процессе abcd\_gen:

### wait for PERIOD\*15;

### wait until rising\_edge(clk);

### 

### a <= to\_signed(50, bit\_width);

### b <= to\_signed(45, bit\_width);

### c <= to\_signed(25, bit\_width);

### d <= to\_signed(10, bit\_width);

wait for PERIOD\*15; - необходим для задержки при post-implantation simulation, так как там учитывается реальная задержка сигналов в железе.

“g. латентность схемы должна быть оптимальной.”

Латентность равна 4-ём тактам процессора.

“h. по возможности реализовать подтверждение входных и выходных данных сигналом valid”

Приложение – содержание main.vhd с реализацией valid.

### library IEEE;

### use IEEE.STD\_LOGIC\_1164.ALL;

### use ieee.numeric\_std.all;

### entity main is

### Generic

### (

### bit\_width : integer := 10

### );

### Port

### (

### clk : in STD\_LOGIC;

### a,b,c,d :in signed(bit\_width-1 downto 0);

### q : out signed(bit\_width-1 downto 0);

### in\_valid : in STD\_LOGIC;

### out\_valid : out STD\_LOGIC

### );

### end main;

### architecture Behavioral of main is

### signal part\_1 : signed(bit\_width downto 0);

### signal part\_2 : signed(bit\_width+1 downto 0);

### signal part\_3 : signed(bit\_width\*2+2 downto 0);

### signal part\_4 : signed(bit\_width\*2+2 downto 0);

### signal q\_reg : signed(bit\_width\*2+2 downto 0);

### signal valid\_reg1 : STD\_LOGIC;

### signal valid\_reg2 : STD\_LOGIC;

### signal valid\_reg3 : STD\_LOGIC;

### signal valid\_reg4 : STD\_LOGIC;

### begin

### process (clk)

### begin

### if rising\_edge(clk) then

### part\_1 <= resize(a, bit\_width+1) - resize(b, bit\_width+1);

### 

### part\_2 <= resize(to\_signed(1, 2), bit\_width+2) + resize(c & '0' + c, bit\_width+2);

### valid\_reg1 <= in\_valid;

### 

### part\_3 <= part\_1 \* part\_2;

### valid\_reg2 <= valid\_reg1;

### 

### part\_4 <= part\_3 - resize(d & "00", bit\_width\*2+3);

### valid\_reg3 <= valid\_reg2;

### 

### q\_reg <= part\_4 / 2;

### valid\_reg4 <= valid\_reg3;

### 

### end if;

### end process;

### q <= resize(q\_reg, bit\_width);

### out\_valid <= valid\_reg4;

### end Behavioral;

Приложение – содержание model.vhd с реализацией valid

### LIBRARY ieee;

### USE ieee.std\_logic\_1164.ALL;

### USE ieee.numeric\_std.ALL;

### LIBRARY UNISIM;

### USE UNISIM.Vcomponents.ALL;

### ENTITY testbench IS

### END testbench;

### ARCHITECTURE behavorial OF

### testbench IS

### constant bit\_width : integer := 10;

### COMPONENT main

### Port

### (

### clk : in STD\_LOGIC;

### a,b,c,d :in signed(bit\_width-1 downto 0);

### q : out signed(bit\_width-1 downto 0);

### in\_valid : in STD\_LOGIC;

### out\_valid : out STD\_LOGIC

### );

### END COMPONENT;

### signal clk :std\_logic;

### signal rst : std\_logic;

### signal a : signed(bit\_width-1 downto 0);

### signal b : signed(bit\_width-1 downto 0);

### signal c : signed(bit\_width-1 downto 0);

### signal d : signed(bit\_width-1 downto 0);

### signal q : signed(bit\_width-1 downto 0);

### signal in\_valid : std\_logic;

### signal out\_valid: std\_logic;

### constant PERIOD : TIME := 10 ns;

### constant DUTY\_CYCLE : real := 0.5;

### constant OFFSET : time := 3 ns;

### BEGIN

### UUT: main

### PORT MAP

### (

### clk => clk,

### a => a,

### b => b,

### c => c,

### d => d,

### q => q,

### in\_valid => in\_valid,

### out\_valid => out\_valid

### );

### clk\_gen: PROCESS

### BEGIN

### WAIT for OFFSET;

### CLOCK\_LOOP : LOOP

### clk <= '1';

### WAIT FOR (PERIOD - (PERIOD \* DUTY\_CYCLE));

### clk <= '0';

### WAIT FOR (PERIOD \* DUTY\_CYCLE);

### END LOOP CLOCK\_LOOP;

### END PROCESS;

### abcd\_and\_valid\_gen : PROCESS

### BEGIN

### wait until rising\_edge(clk);

### in\_valid <= '0';

### 

### a <= to\_signed(50, bit\_width);

### b <= to\_signed(45, bit\_width);

### c <= to\_signed(25, bit\_width);

### d <= to\_signed(10, bit\_width);

### in\_valid <= '1';

### wait until out\_valid <= '1';

### in\_valid <= '0';

### 

### wait until rising\_edge(clk);

### 

### a <= to\_signed(15, bit\_width);

### b <= to\_signed(22, bit\_width);

### c <= to\_signed(3, bit\_width);

### d <= to\_signed(11, bit\_width);

### wait for offset;

### in\_valid <= '1';

### wait until out\_valid <= '1';

### in\_valid <= '0';

### 

### wait until rising\_edge(clk);

### 

### a <= to\_signed(23, bit\_width);

### b <= to\_signed(5, bit\_width);

### c <= to\_signed(4, bit\_width);

### d <= to\_signed(10, bit\_width);

### wait for offset;

### in\_valid <= '1';

### wait until out\_valid <= '1';

### in\_valid <= '0';

### 

### wait until rising\_edge(clk);

### 

### a <= to\_signed(10, bit\_width);

### b <= to\_signed(40, bit\_width);

### c <= to\_signed(4, bit\_width);

### d <= to\_signed(110, bit\_width);

### wait for offset;

### in\_valid <= '1';

### wait until out\_valid <= '1';

### in\_valid <= '0';

### 

### wait until rising\_edge(clk);

### 

### a <= to\_signed(-10, bit\_width);

### b <= to\_signed(41, bit\_width);

### c <= to\_signed(-4, bit\_width);

### d <= to\_signed(100, bit\_width);

### wait for offset;

### in\_valid <= '1';

### wait until out\_valid <= '1';

### in\_valid <= '0';

### 

### wait until rising\_edge(clk);

### 

### a <= to\_signed(-11, bit\_width);

### b <= to\_signed(100, bit\_width);

### c <= to\_signed(-6, bit\_width);

### d <= to\_signed(110, bit\_width);

### wait for offset;

### in\_valid <= '1';

### wait until out\_valid <= '1';

### WAIT;

### END PROCESS;

### END;

# Задание 9

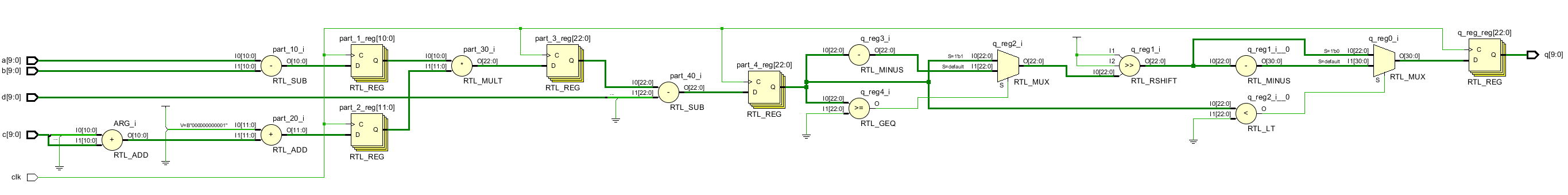


Рисунок – Elaborated Design схемы без реализации valid

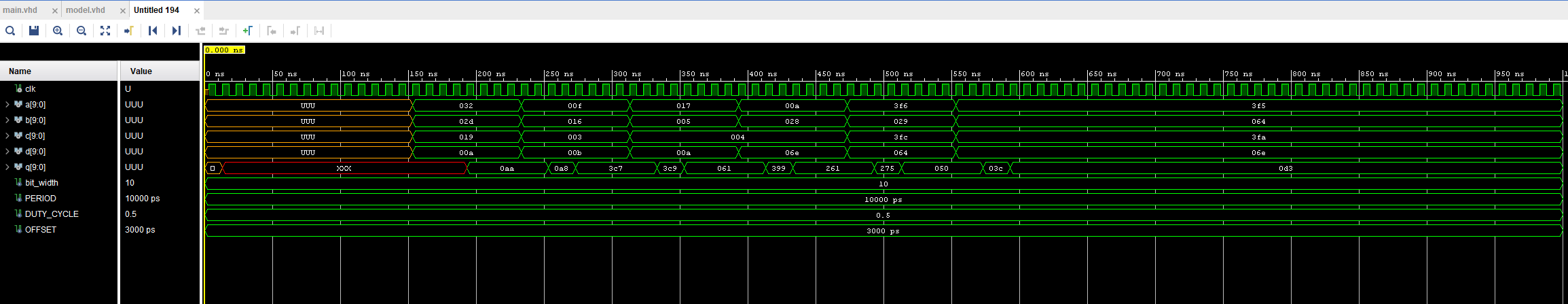


Рисунок – behavioral simulation без valid

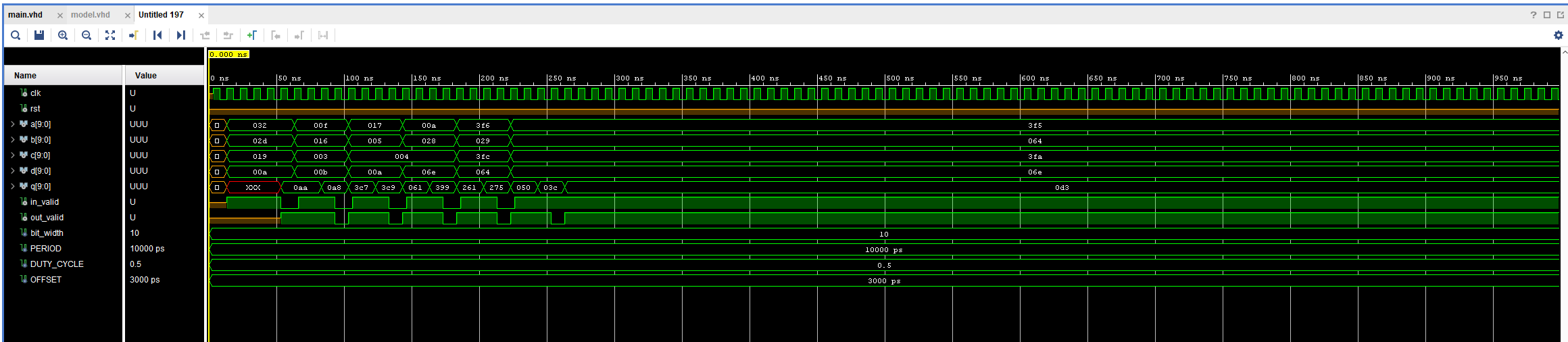


Рисунок – behavioral simulation с valid

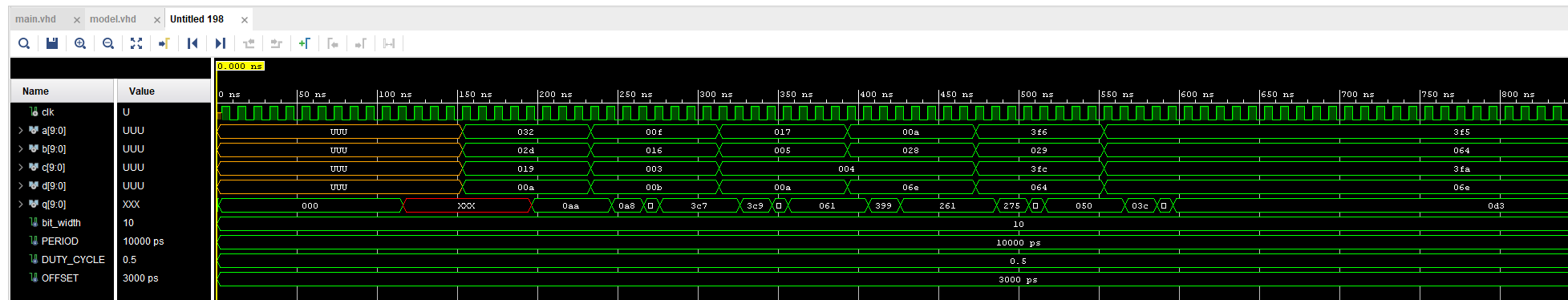


Рисунок – post-synthesis simulation без valid

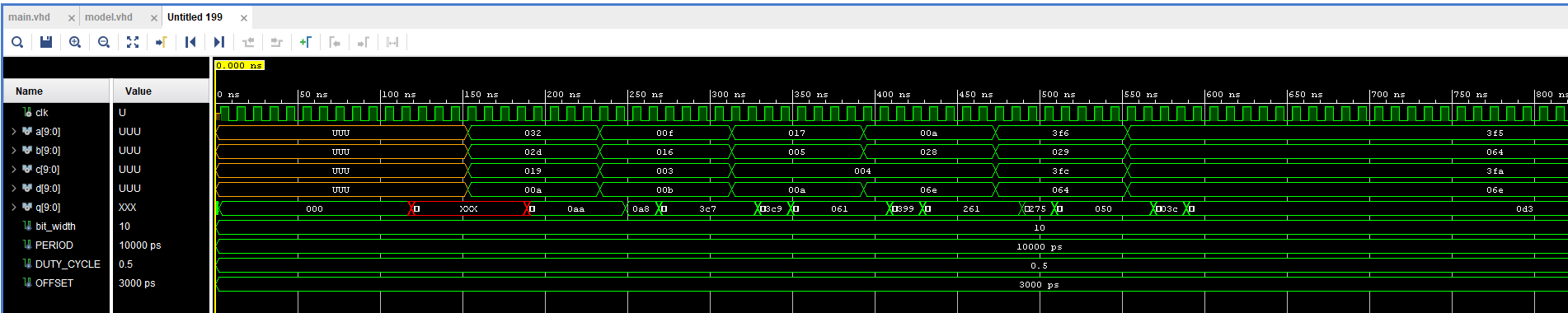


Рисунок – post-implementation simulation без valid

# Задание 10

Приложение – содержание math\_equation.py

### def main ():

### n = int(input("Bit depth:"))

### a = int(input("Input a:"))

### b = int(input("Input b:"))

### c = int(input("Input c:"))

### d = int(input("Input d:"))

### part\_1 = int(a - b)

### part\_2 = int (1 + (3 \* c))

### part\_3 = int (4 \* d)

### part\_4 = int((part\_1 \* part\_2) - part\_3)

### q = int(part\_4 / 2)

### print("int representation:", q)

### if (q < 0):

### print("binary representation:", bin(q + (1 << n))[2:])

### print("hex representation:", hex(int(bin(q + (1 << n))[2:],2))[2:])

### else:

### print("binary representation:", bin(q + (1 << n))[3:])

### print("hex representation:", hex(int(bin(q + (1 << n))[3:],2))[2:])

### if \_\_name\_\_ == "\_\_main\_\_":

### main()

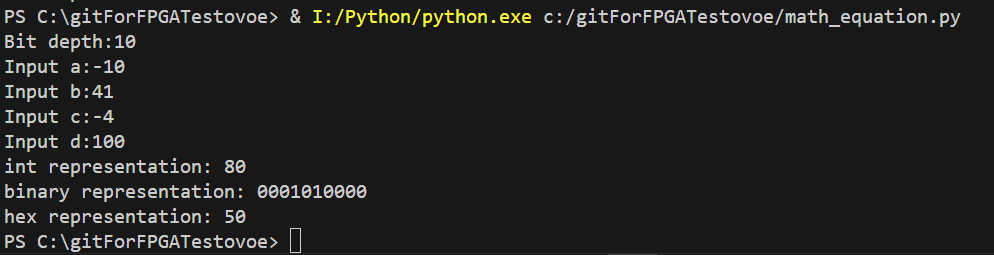


Рисунок – результат работы math\_equation.py

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Python | | | | |
| n | 10 | 10 | 10 | 10 | 10 |
| a | 50 | 15 | 23 | 10 | -10 |
| b | 45 | 22 | 5 | 40 | 41 |
| c | 25 | 3 | 4 | 4 | -4 |
| d | 10 | 11 | 10 | 110 | 100 |
| int | 170 | -57 | 97 | -415 | 80 |
| binary | 0010101010 | 1111000111 | 0001100001 | 1001100001 | 0001010000 |
| Hex | aa | 3c7 | 61 | 261 | 50 |

# Задание 11

Защита от переполнения может быть реализована с помощью временной переменной temp, которая будет проверять переполнения после умножения, и после вычитания

### process (clk)

### variable temp : signed(bit\_width\*2+3 downto 0);

### begin

### if rising\_edge(clk) then

### temp := part\_1 \* part\_2;

### if (temp > 2\*\*(bit\_width\*2+2)-1) or (temp < -2\*\*(bit\_width\*2+2)) then

### report "Multiplication overflow detected" severity warning;

### end if;

### temp := temp - resize(d & "00", bit\_width\*2+3);

### if (temp > 2\*\*(bit\_width\*2+2)-1) or (temp < -2\*\*(bit\_width\*2+2)) then

### report "Subtraction overflow detected" severity warning;

### end if;

### end if;

### end process;

# Задание 12

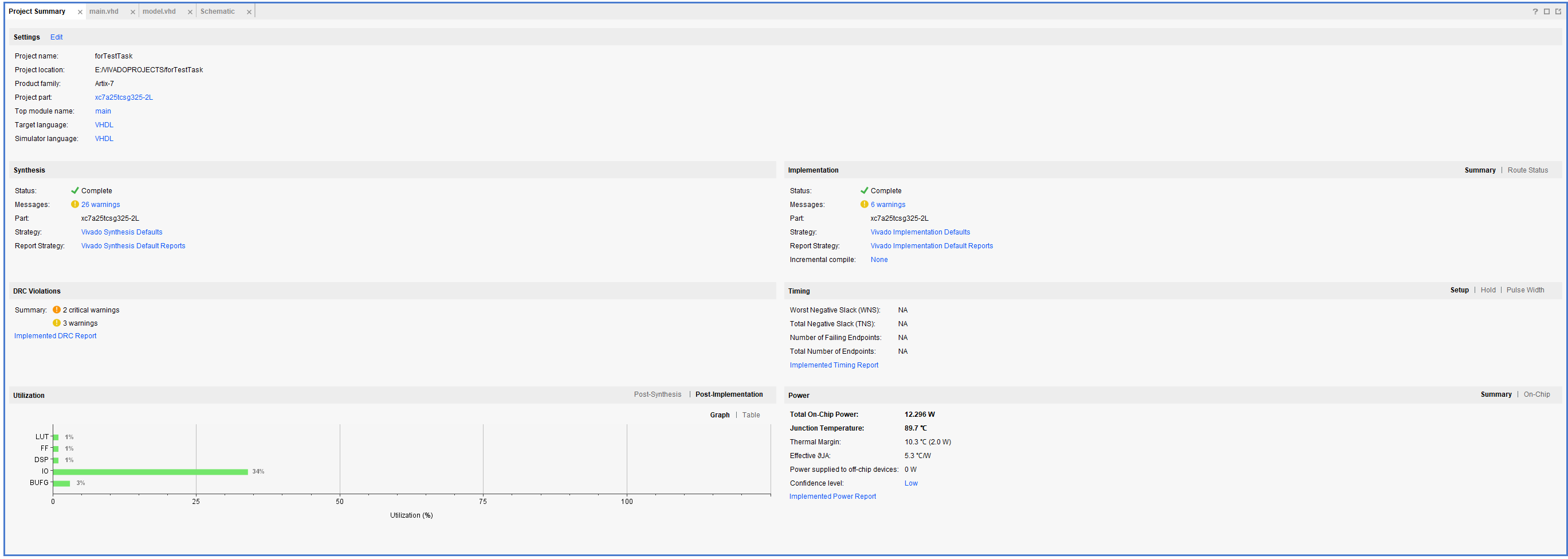


Рисунок – скриншот Project Summary

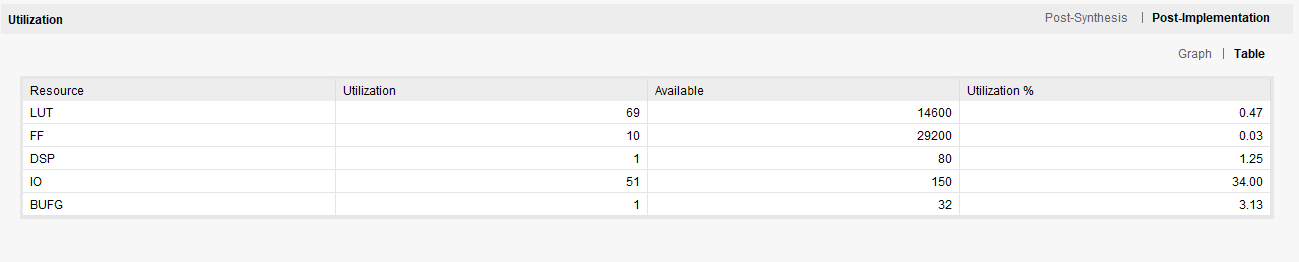


Рисунок –использование ресурсов, требуемых для реализации схемы.

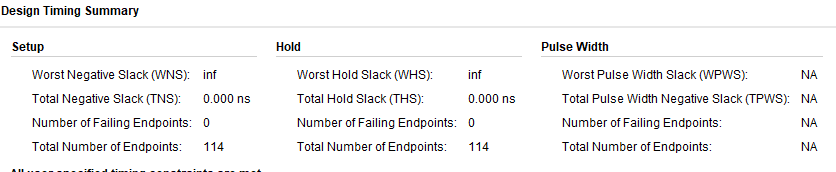


Рисунок – скриншот Report Timing Summary

Большая часть ресурсов платы ушла на выделение портов.