ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Институт информатики и кибернетики

Кафедра радиотехники

Отчет по индивидуальному домашнему заданию:
”Разработка цифровых устройств на базе ПЛИС”
Вариант №15

Студент: Согонов Е.А.

Преподаватель: Корнилин Д.В.

Группа: 6364-120304D

Самара 2022

СОДЕРЖАНИЕ

1. Формализация текстового задания.

Исходное задание звучит так:

Часы с индикацией минут и секунд на четырехзначном семисегментном индикаторе

Схематически цифровое устройство, реализующее данный функционал можно изобразить с помощью блок-схемы, изображенной на рисунке 1.

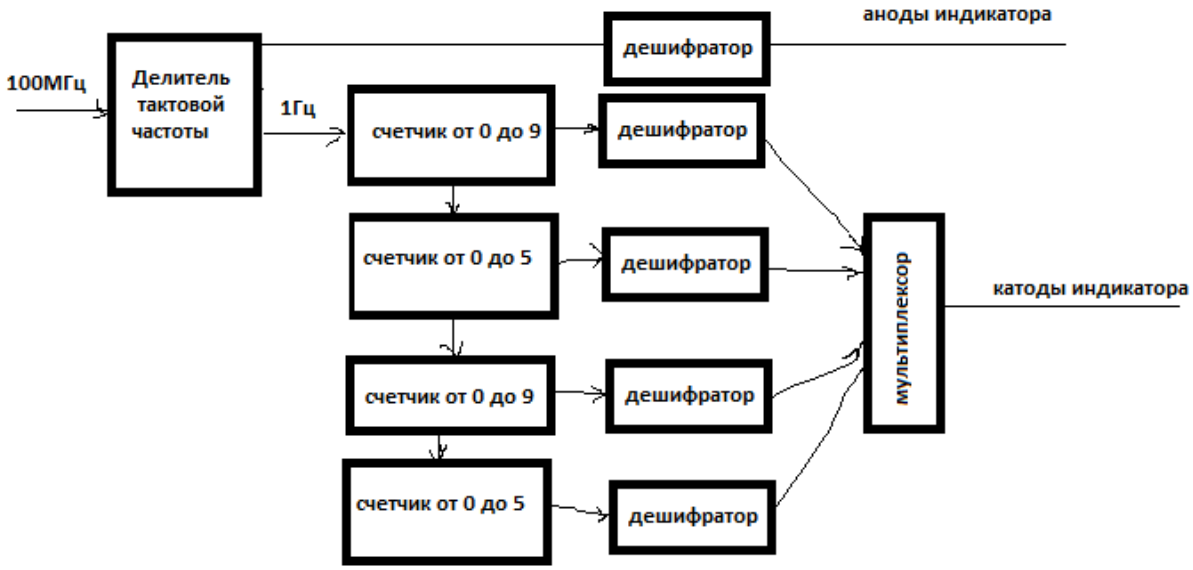

Рисунок 1 – Блок-схема часов

Первый счетчик делит частоту встроенного в отладочную плату осциллятора со 100МГц до 1 Гц, что соответствует периоду в одну секунду, следующий счетчик считает количество этих импульсов, считая от 0 до 9. Следующий счетчик считает от 0 до 6, что соответствует десяткам секунд. Аналогично следующие два счетчика соответствуют единицам и десяткам минут. Текущее значение каждого из счетчиков поступает на дешифратор, и затем на мультиплексор, который управляет катодами семисегментного индикатора. На аноды каждого из 4 индикаторов напряжение подается поочередно, с частотой незаметной глазу. Реализовать это можно с помощью автомата с конечным числом состояний, но в текущей ситуации проще взять 2 меняющихся бита какого-то из счетчиков, и подать их на дешифратор и затем мультиплексор. Это и даст поочередное переключение 4 индикаторов. Функция остановки будет реализована с помощью переключателя: в одном положении счет осуществляется, в другом - нет, на индикаторе будут изображены те цифры, что были до остановки. (нихуя так работать не будет - при остановке счетчика выключится и индикатор, он будет показывать одну цифру. Фикс проблемы сделаю на паре, попробуйте пока это потестить)

2. Описание устройства с помощью VHDL

Описаннный в предыдущем пункте функционал реализуется с помощью следующего описания устройства на языке VHDL:

```
library IEEE;--подключение библиотек
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

entity stopwatch is    --Декларация entity и описание портов
      Port (
            clk : in  STD_LOGIC;
            reset : in  STD_LOGIC;
            start_stop : in  STD_LOGIC;
            seg : out STD_LOGIC_VECTOR (7 downto 0);
     dig  : out STD_LOGIC_VECTOR (3 downto 0)
            );
end stopwatch;
-- архитектурное тело
architecture Behavioral of stopwatch is  --Архитектура Behavioral для интерфейса stopwatch
--внутренние сигналы
signal cnt:   unsigned(25 downto 0);
signal sec_e: unsigned(3 downto 0);
signal sec_d: unsigned(3 downto 0);
signal min_e: unsigned(3 downto 0);
signal min_d: unsigned(3 downto 0);

function f_num_2_7seg (num : in unsigned(3 downto 0)) --функция, получающая на вход значение счетчика типа unsigned
return std_logic_vector is       variable seg7 : std_logic_vector(6 downto 0); --возвращающая значение типа std_logic_vec
--необходимое для управления катодами семисегментного индикатора
begin
     if num = X"0" then seg7 := b"1000000";
   elsif num = X"1" then seg7 := b"1111001";
   elsif num = X"2" then seg7 := b"0100100";
   elsif num = X"3" then seg7 := b"0110000";
   elsif num = X"4" then seg7 := b"0011001";
   elsif num = X"5" then seg7 := b"0010010";
   elsif num = X"6" then seg7 := b"0000010";
   elsif num = X"7" then seg7 := b"1111000";
   elsif num = X"8" then seg7 := b"0000000";
   elsif num = X"9" then seg7 := b"0010000";
   else seg7 := (others => '1');
   end if;
   return std_logic_vector(seg7);
end;

begin
process(clk, reset,start_stop) --процесс чувствителен к изменению сигналов clk, reset, start_stop
begin
if start_stop='1' then --проверка положения переключателя старт-стоп
   if reset='1' then --проверка сброса
        cnt <= (others => '0');
      sec_e <= (others => '0');
      sec_d <= (others => '0');
      min_e <= (others => '0');
      min_d <= (others => '0');
   elsif rising_edge(CLK) then --если пришел передний фронт тактового сигнала
      --if cnt=to_unsigned(100000000,26) then cnt <= (others => '0'); --считаем до 100 миллионов(для синтеза)
      if cnt=to_unsigned(100,26) then cnt <= (others => '0'); --считаем до 100(преобразуем число 100 в тип unsigned разме
         if sec_e=9 then sec_e <= x"0"; --счетчик единиц секунд, если уже имеет значение 9, сброс и передача импульса сл
```

```
            if sec_d=5 then sec_d <= x"0";
               if min_e=9 then min_e <= x"0";
                  if min_d=5 then min_d <= x"0";
                  else min_d <= min_d + 1; end if;
               else min_e <= min_e + 1; end if;
            else sec_d <= sec_d + 1; end if;
         else sec_e <= sec_e + 1; end if; --иначе (если на счетчике единиц не 9), увеличиваем значение счетчика на 1.
      else cnt <= cnt + 1; end if; --аналогично
      --case to_integer(cnt(15 downto 14)) is
      case to_integer(cnt(6 downto 5)) is --в качестве переменной, которая будет меняться, выбраны два бита из unsigne
         when 0 => Dig <= b"1000"; seg <= ("1"    & f_num_2_7seg(sec_e)); --на самый правый индикатор вывести точку(
         when 1 => Dig <= b"0100"; seg <= ("1"    & f_num_2_7seg(sec_d)); -- аналогично
         when 2 => Dig <= b"0010"; seg <= (cnt(25)& f_num_2_7seg(min_e)); --аналогично, только тут точка будет мига
         when 3 => Dig <= b"0001"; seg <= ("1"    & f_num_2_7seg(min_d));
         when others => Dig <= b"0000"; seg <= (others => '1');
      end case;
   end if;
 end if;
end process;
end Behavioral;
```

Данный код полностью описывает логику требуемого устройства.

3. Симуляция устройства

Чтобы проверить, насколько полученное описание хорошо работает, нужно запустить симуляцию. Для ускорения симуляции был изменен кусок кода со счетчиком, считающим до 100 миллионов.

```
--if cnt=to_unsigned(100000000,26) then cnt <= (others => '0'); --считаем до 100 миллионов
if cnt=to_unsigned(100,26) then cnt <= (others => '0'); --считаем до 100
```

Так же для симуляции необходимо создать файл симуляции (Test bench) SIMM.vhd на языке VHDL, содержимое которого можно увидеть ниже:

```
--тестовый файл для симуляции устройства
library IEEE;--подключение библиотек
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
COMPONENT stopwatch -- декларация компонента для UUT
   PORT(
      clk : in  STD_LOGIC;          --описываем входы и выходы блока
      reset : in  STD_LOGIC;
      start_stop : in  STD_LOGIC
      );
END COMPONENT;

signal clk : std_logic := '0'; --декларация сигналов и присвоение им значения "0"
signal reset : std_logic := '0';
signal start_stop : std_logic := '0';

begin
```

```
uut: stopwatch --конкретизация компонента для Unit Under Test (UUT)
PORT MAP ( --подключение выводов
   clk => clk,  reset => reset,start_stop => start_stop
);

clock: process --создание тактового сигнала
begin
   clk <= '0'; wait for 5 ns;
   clk <= '1'; wait for 5 ns;
end process;

reseting : process--создание тестового сигнала(симуляция нажатия на кнопку reset)
begin
   reset <= '0'; wait for 1 ns;
   reset <= '1'; wait for 5 ns;
   reset <= '0'; wait for 150 ms;
end process;

stop_start : process--создание тестового сигнала(симуляция нажатия на кнопку reset)
begin
   start_stop <= '0'; wait for 1 ns;
   start_stop <= '1'; wait for 3 ms;
   start_stop <= '0'; wait for 1 ms;
   start_stop <= '1'; wait for 10 ns;
end process;

end Behavioral;
```

После запуска симуляции были получены временные диаграммы, изображенные на рисунках 2, 3, 4, 5, 6, 7

В начальный момент времени значения всех счетчиков не определены. При сбросе с помощью сигнала reset значения счетчиков обнуляются, после чего начинается счет посредством счетчика cnt. Досчитав до 100(вообще он должен до 100 миллионов считать, но для симуляции ждать такое значение бессмысленно), счетчик сбрасывается, передавая импульс следующему счетчику, считающего единицы секунд. Он в свою очередь считает до 9, и тактует счетчик десятков секунд, считающего до 5. Так же, можно просмотреть момент, когда при нажатии кнопок, прибавляющих по секунде и минуте соответственно, происходит изменение значения счетчиков.

дальше все картинки вам не подходят

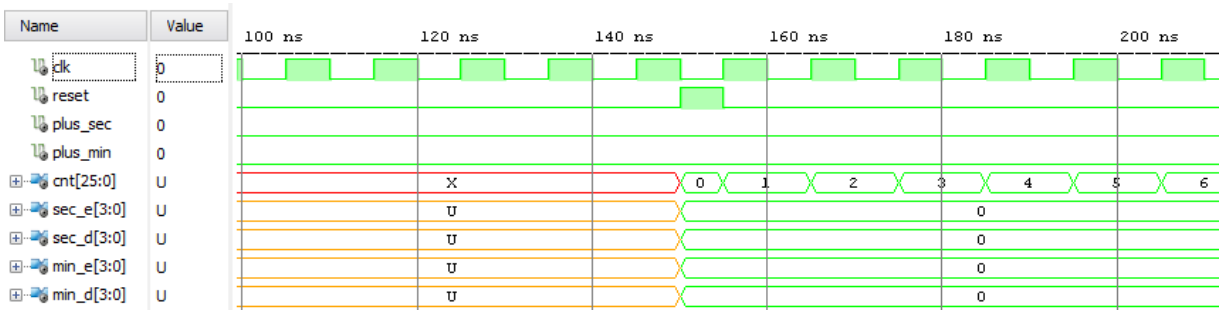

Рисунок 2 – Нулевой момент времени

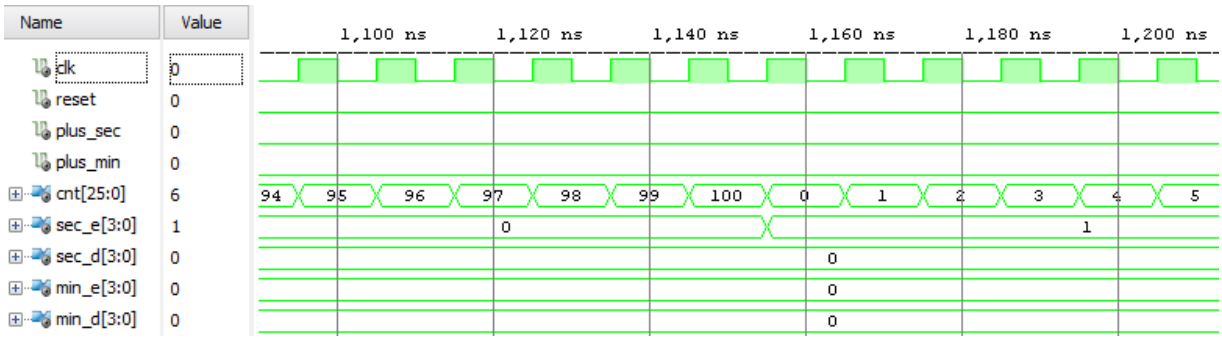

Рисунок 3 – Счетчик единиц секунд при приходе импульса от делителя тактовой частоты

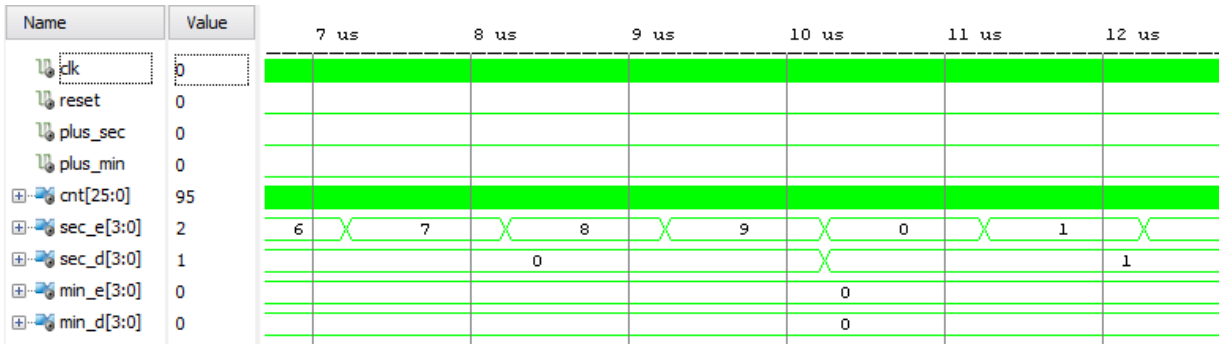

Рисунок 4 – Переключение счетчика десятков секунд счетчиком единиц секунд

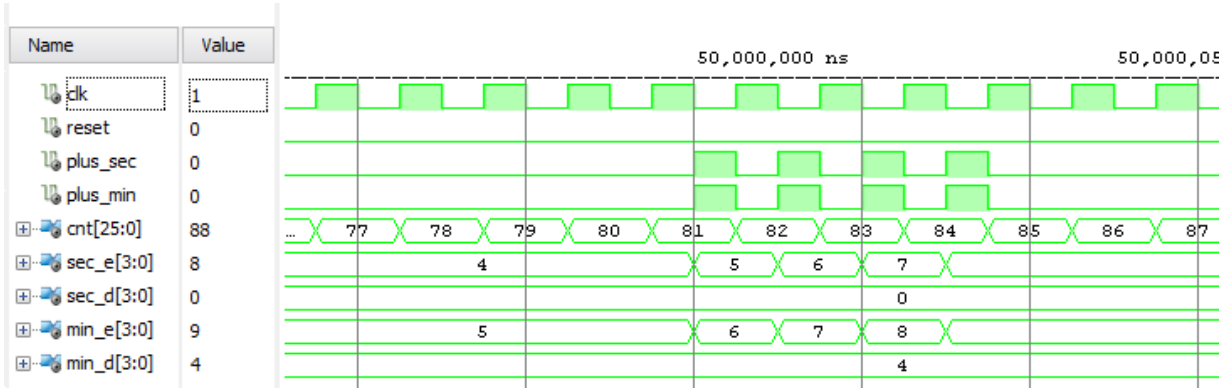

Рисунок 7 – Нажатие plus_sec и plus_min

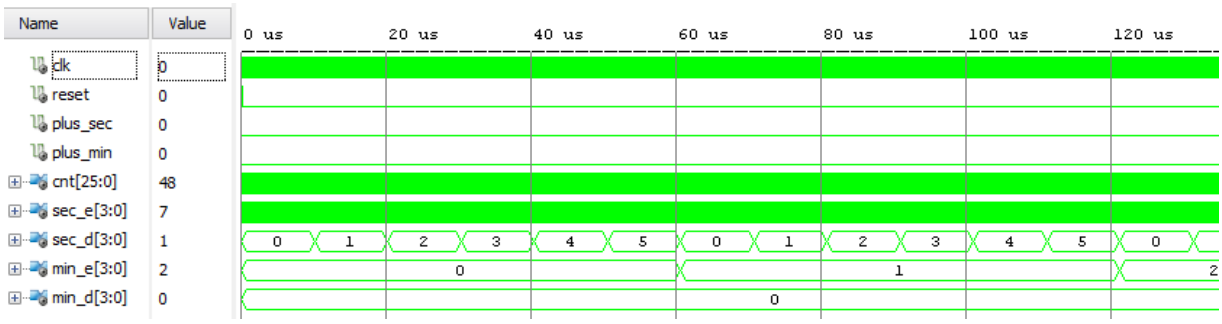

Рисунок 5 – Переключение счетчика единиц минут счетчиком десятков секунд

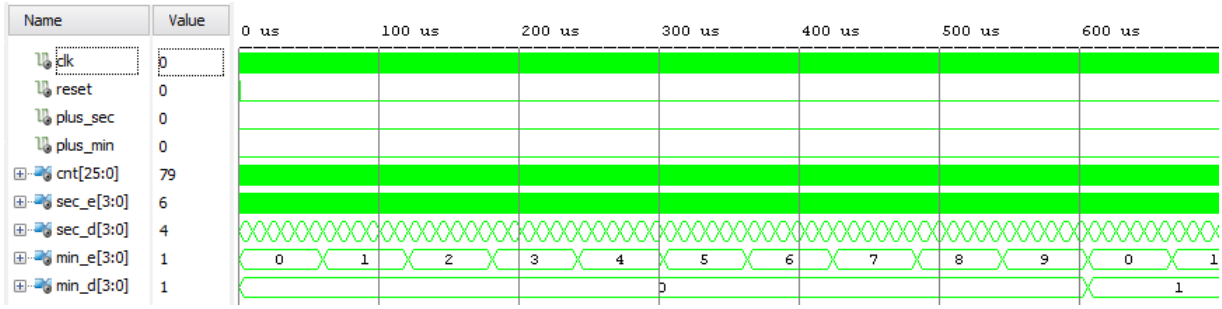

Рисунок 6 – Переключение счетчика десятков минут счетчиком единиц минут

Таким образом, симуляцию можно считать успешной.

4. Синтез цифрового устройства

Для синтеза необходимо задать временные ограничения. Создаем файл временных ограничений, в котором нужно задать частоту тактового сигнала, в окне Primary Clocks. Так же можно задать параметры и для Generated Clocks, но особого смысла в этом нет.

В результате синтеза была получена следующая схема(рисунок 8)

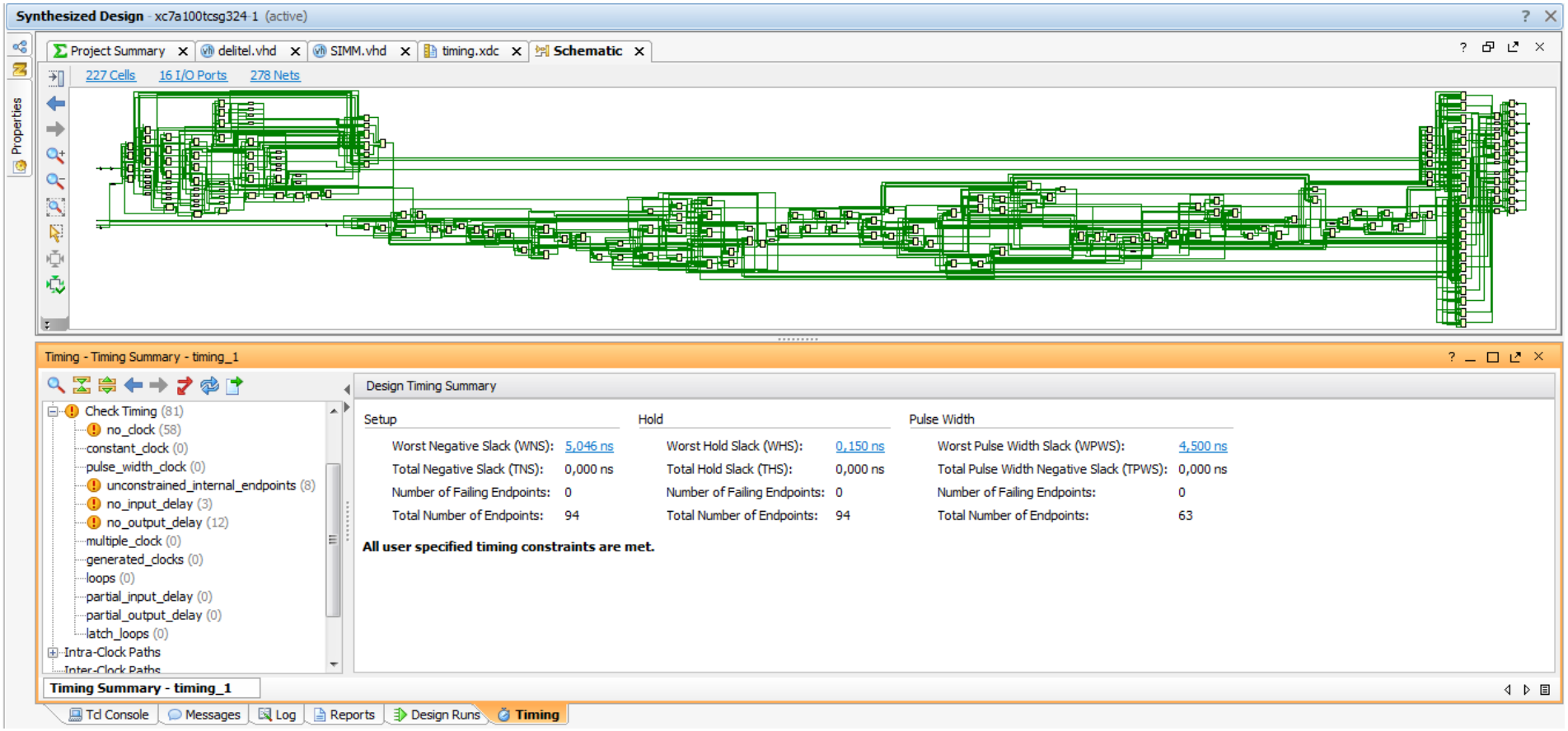

Рисунок 8 – Синтезированная схема и сводка по времени

Система отмечает отсутствие ограничений Input и Output, а также группы no clock и unconstrained elements. Причина этого в том, что все счетчики тактируется выходным сигналом другого счетчика, который не указан в качестве тактового.

Так же на этом этапе необходимо указать внешние порты ввода-вывода для размещения схемы внутри ПЛИС. Сделать это нужно вручную.

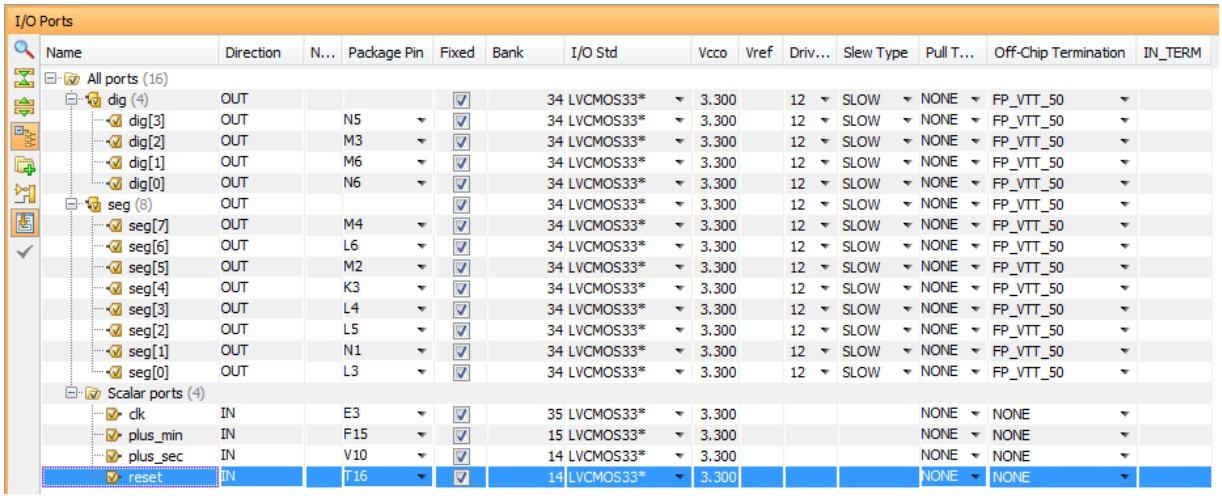

Рисунок 9 – Расположение портов ввода-вывода

5. Реализация

Следующий шаг разработки устройства – это реализация. После завершения процесса реализации нужно проверить выполнение временных ограничений (Report Timing Summary)(рисунок 10)

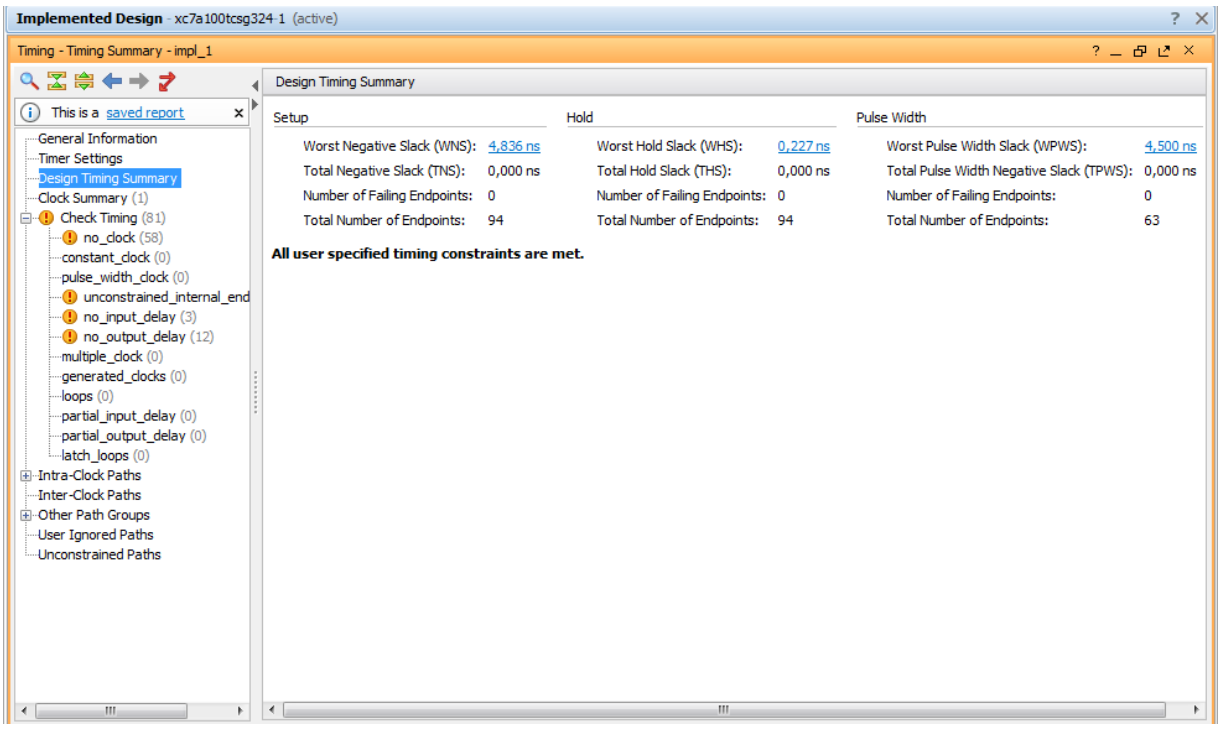

Рисунок 10 – временная сводка после реализации

6. Программирование ПЛИС

Для программирования ПЛИС нужно выбрать Generate Bitstream на левой панели (Flow Navigator), а после завершения процесса открыть Hardware Manager, подключить кабель USB к отладочной плате (разъем PROG) и затем к компьютеру. Включить плату (переключатель POWER на плате), а также проверить правильность установки перемычек JP1 и JP2.

7. Проверка функционирования устройства на отладочной плате