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 №4

Студенты: Гуськова К.М.,
Рожновская Д.О.
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.

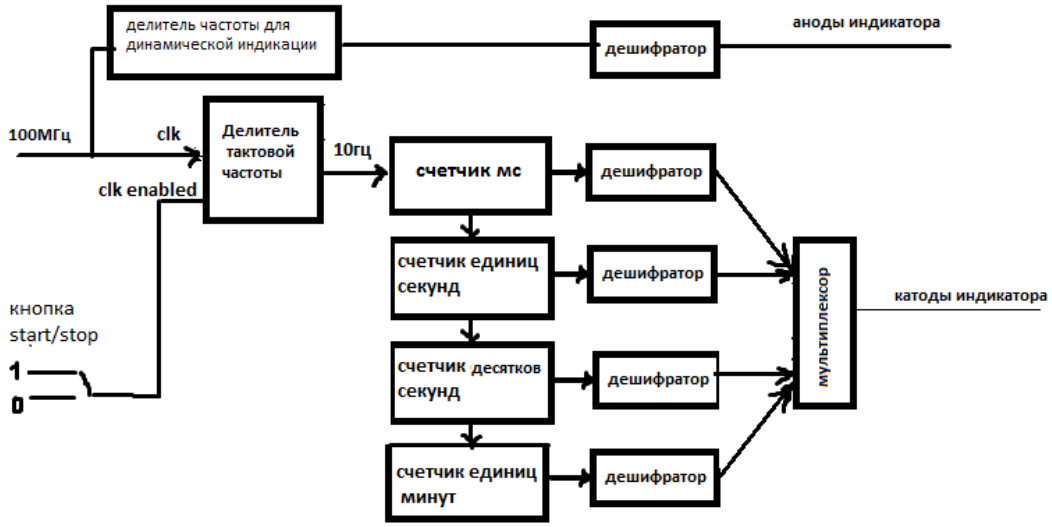

Рисунок 1 – Блок-схема секундомера

Первый счетчик делит частоту встроенного в отладочную плату осциллятора со 100 МГц до 10 Гц, что соответствует периоду в одну миллисекунду.

Счетчик миллисекунд считает количество этих импульсов, считая от 0 до 9. Следующий счетчик считает так же от 0 до 9, что соответствует единицам секунд.

Аналогично следующие два счетчика соответствуют десяткам секунд(считает от 0 до 5) и единицам минут(от 0 до 9).

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. Это обеспечивается с помощью отдельного делителя частоты. В качестве переменной величины для мультиплексора можно взять несколько меняющихся битов какого-то вектора. Это и даст поочередное переключение 4 индикаторов.

Функция start–stop будет реализована с помощью кнопки без фиксации: по первому нажатию секундомер запускается, по второму останавливается, из любого этих состояний секундомер можно сбросить, иначе говоря, в одном состоянии счет времени осуществляется, в другом - нет, на индикаторе будут изображены те цифры, что были до остановки.

2. Описание устройства с помощью VHDL

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

```
--файл описания устройства
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 : in  STD_LOGIC;
                  seg : out STD_LOGIC_VECTOR (6 downto 0);
                  dig : out STD_LOGIC_VECTOR (3 downto 0)
        );
end stopwatch;

architecture Behavioral of stopwatch is
--Архитектура Behavioral для интерфейса stopwatch
--внутренние сигналы
signal cnt:   unsigned(28 downto 0);
signal msec: unsigned(3 downto 0);
signal sec_e: unsigned(3 downto 0);
signal sec_d: unsigned(3 downto 0);
signal min_e: unsigned(3 downto 0);
signal hex: unsigned(19 downto 0);
signal start_stop: std_logic;
signal Q1, Q2, Q3, start_up: std_logic;


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_vector,
--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)
 --процесс чувствителен к изменению
```

```
--сигналов clk, reset
begin
--создание отдельного сигнала, необходимого для создания
--переменного аргумента для оператора case
if reset='1' then hex <= (others => '0');
elsif rising_edge(clk) then
  if hex=to_unsigned(100_000_0,20) then hex <= (others => '0');
     else hex <= hex + 1;
  end if;
end if;
end process;

process(clk, reset, start)--debounce
begin
  if rising_edge(clk) then
    if reset = '1' then
      Q1 <= '0';
      Q2 <= '0';
      Q3 <= '0';
      start_up <='0';
    else
      Q1 <= start;
      Q2 <= Q1;
      Q3 <= Q2;
    end if;
  end if;
  start_up <= Q1 and Q2 and (not Q3);
end process;

process(clk, reset, start_up)
--процесс чувствителен к изменению сигналов clk, reset, start_up
begin
--T-триггер, запоминающий предыдущее состояние,
--при нажатии кнопки start(прошедшей через
--модуль устранения дребезга
--ставшей сигналом start_up
--триггер сбрасывается на противоположное состояние
if reset='1' then start_stop<='0';
elsif rising_edge(clk)then
  if start_up='1' then start_stop<=not (start_stop);
  end if;
end if;
end process;

process(clk, reset, start_stop)
--процесс чувствителен к изменению сигналов clk, reset, start_stop
begin
--проверка, нажата ли кнопка reset и сброс всех счетчиков
if reset='1' then
     cnt <= (others => '0');
   msec <= (others => '0');
   sec_e <= (others => '0');
   sec_d <= (others => '0');
   min_e <= (others => '0');
end if;
------Основная часть устройства
if rising_edge(CLK) then -- по переднему фронту тактового сигнала начинаем счет
   if start_stop='1' then --проверка положения нажатия кнопки старт-стоп
     if cnt=to_unsigned(10_000_000,28) then cnt <= (others => '0');
     --если досчитали до 10 миллионов, сбросить счетчик. На выходе счетчика получаем такты
```

```
      --частотой 10Гц, что соответствует периоду 1мс
         if msec=9 then msec <= x"0";
            if sec_e=9 then sec_e <= x"0";
               if sec_d=5 then sec_d <= x"0";
                  if min_e=9 then min_e <= x"0";
                  else min_e <= min_e + 1; end if;
               else sec_d <= sec_d + 1; end if;
            else sec_e <= sec_e + 1; end if;
         else msec <= msec + 1; end if;
      else cnt <= cnt + 1; end if;
   end if;
end if;
end process;
process(hex,msec,sec_e,sec_d,min_e)
--мультиплексор, занимающийся динамической индикацией
begin
case to_integer(hex(10 downto 9)) is
   when 0 => Dig <= b"0111"; seg <= (f_num_2_7seg(msec));
   when 1 => Dig <= b"1011"; seg <= (f_num_2_7seg(sec_e));
   when 2 => Dig <= b"1101"; seg <= (f_num_2_7seg(sec_d));
   when 3 => Dig <= b"1110"; seg <= (f_num_2_7seg(min_e));
   when others => Dig <= b"1111"; seg <= (others => '1');
end case;
end process;
end Behavioral;
```

Данный код полностью описывает логику требуемого устройства.

3. Симуляция устройства

Чтобы проверить, как полученное с помощью этого описания устройство работает, нужно запустить симуляцию. Для сокращения времени симуляции был изменен кусок кода со счетчиком, считающим до 10 миллионов.

```
--if cnt=to_unsigned(10_000_000,28) then cnt <= (others => '0'); --считаем до 10M
if cnt=to_unsigned(1000,28) then cnt <= (others => '0'); --считаем до 1000
```

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
-- декларация компонента для UUT
COMPONENT stopwatch
   PORT(
      clk : in  STD_LOGIC;          --описываем входы и выходы блока
      reset : in  STD_LOGIC;
      start : in  STD_LOGIC;
      seg : out STD_LOGIC_VECTOR (6 downto 0);
```

```
      dig  : out STD_LOGIC_VECTOR (3 downto 0)
      );
END COMPONENT;
--декларация сигналов и присвоение им значения "0"
signal clk : std_logic := '0';
signal reset : std_logic := '0';
signal start : std_logic := '0';

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

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 6 ns;
   reset <= '1'; wait for 25 ns;
   reset <= '0'; wait for 1500 ms;
end process;

stop_start : process
--создание тестового сигнала(симуляция нажатия на кнопку start/stop)
begin
   start <= '0'; wait for 100 ns;
   start <= '1'; wait for 20 ns;
   start <= '0'; wait for 30 ms;
end process;

end Behavioral;
```

После запуска симуляции были получены временные диаграммы, изображенные на рисунках 2, 4, 5, 6, 7, 8

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 1000(вообще он должен до 100 миллионов считать, но для симуляции ждать такое значение бессмысленно), счетчик сбрасывается, передавая импульс следующему счетчику, считающего миллисекунды. Он, в свою очередь считает до 9, и тактует счетчик единиц секунд, считающего до 9. Аналогично обстоит дело со следующими счетчиками. Так же, можно просмотреть момент, когда при нажатии кнопки start-stop запускается и останавливается счет времени.

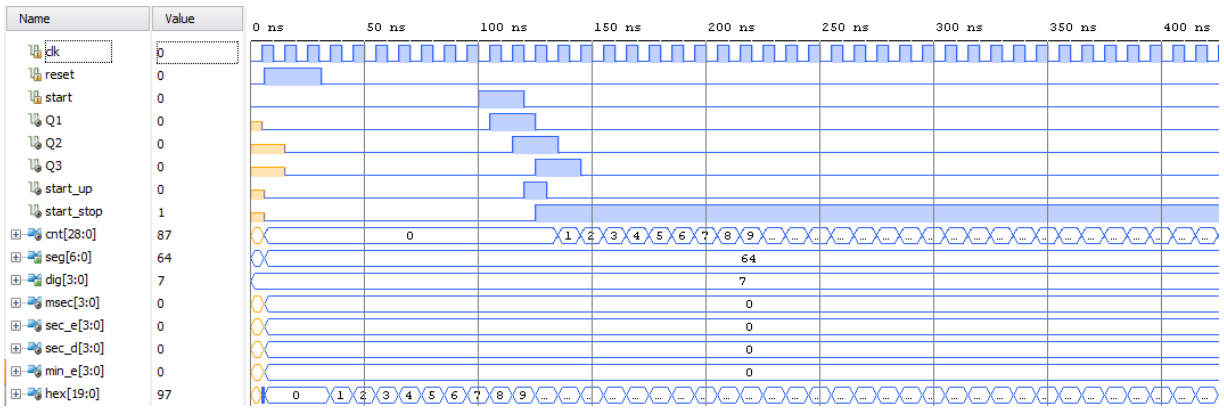

Рисунок 2 – Нулевой момент времени, сброс всех сигналов сигналом reset, начало отсчета счетчика cnt

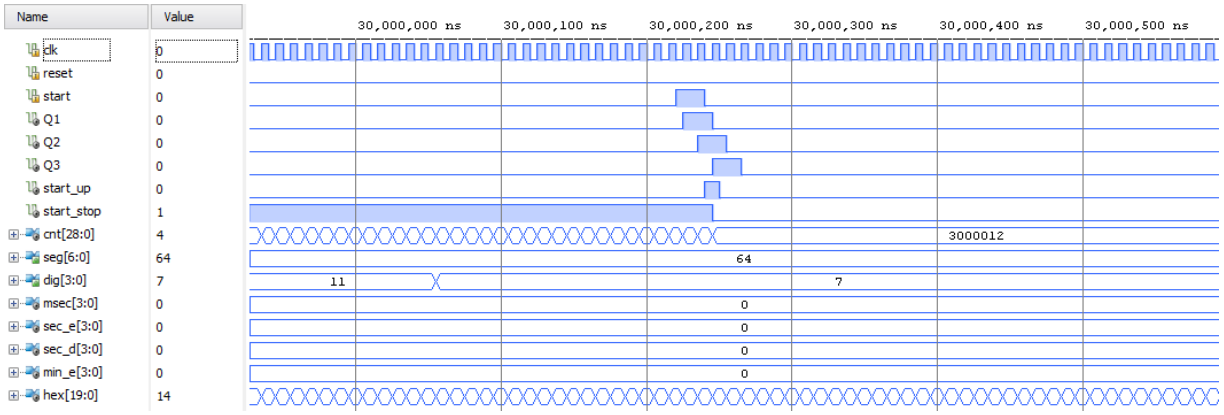

Рисунок 3 – Остановка счета времени повторным нажатием кнопки start-stop

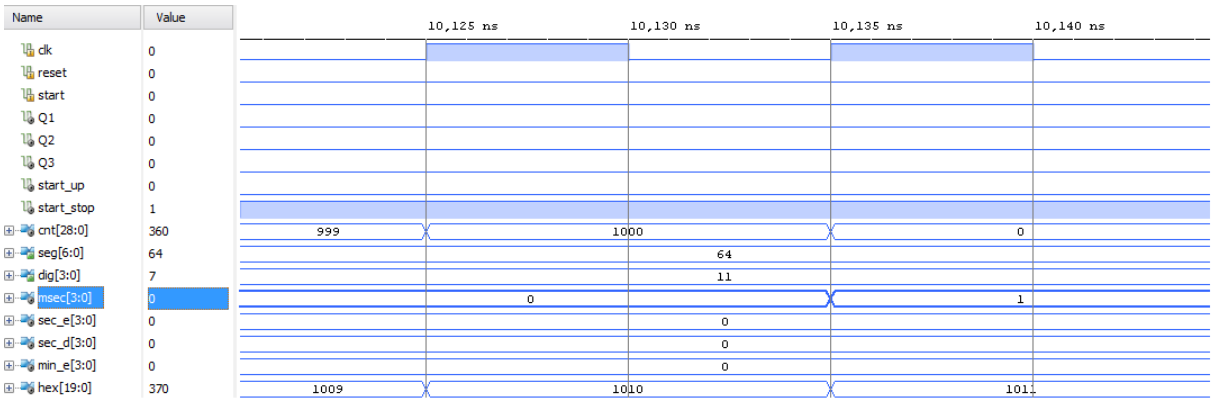

Рисунок 4 – Переключение счетчика миллисекунд при приходе импульса от делителя тактовой частоты cnt

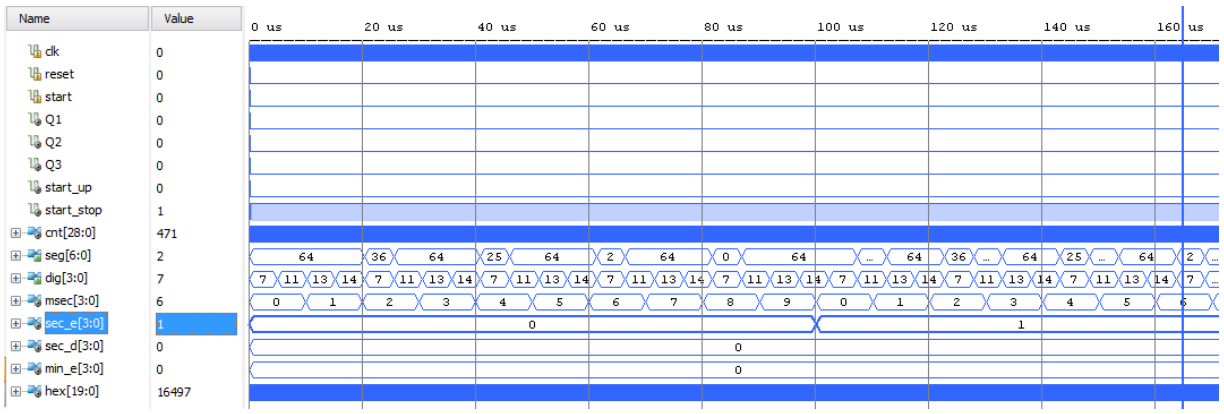

Рисунок 5 – Переключение счетчика единиц секунд счетчиком миллисекунд

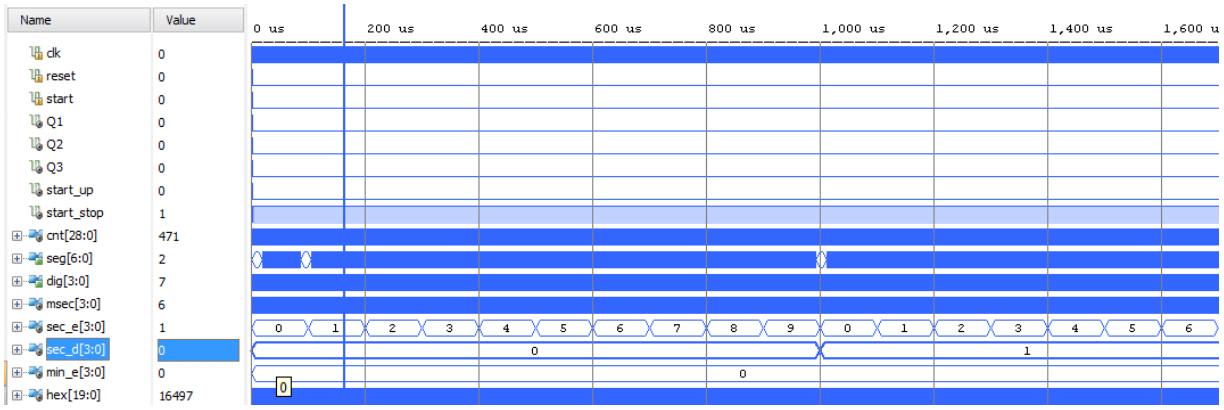

Рисунок 6 – Переключение счетчика десятков секунд счетчиком единиц секунд

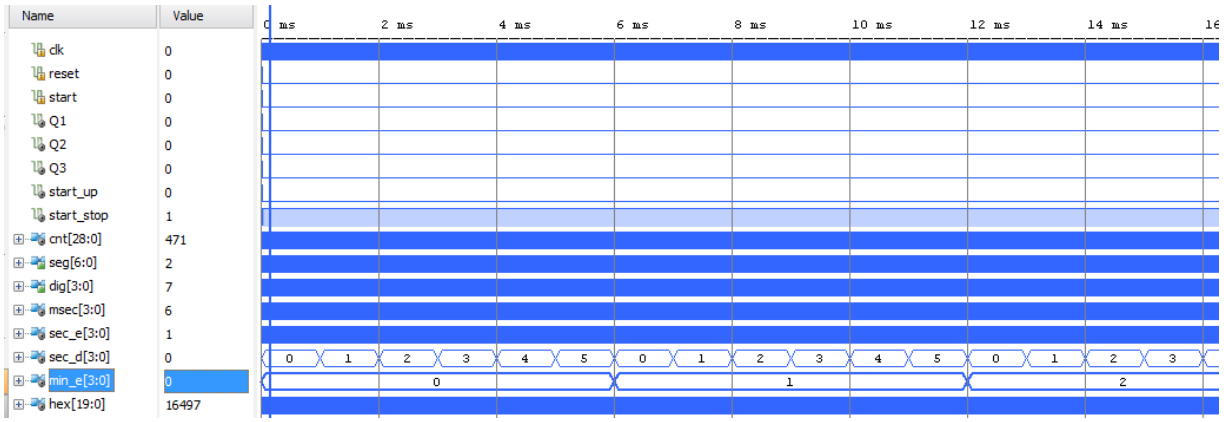

Рисунок 7 – Переключение счетчика единиц минут счетчиком десятков секунд

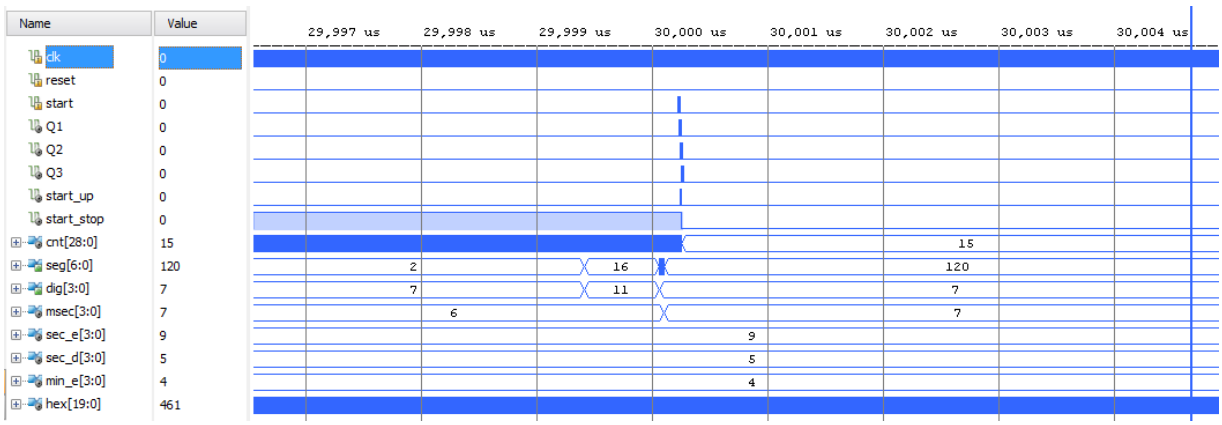

Рисунок 8 – Остановка счета нажатием кнопки start-stop – значения счетчиков зафиксировались и не меняются

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.

В результате синтеза была получена следующая схема(рисунок 9)

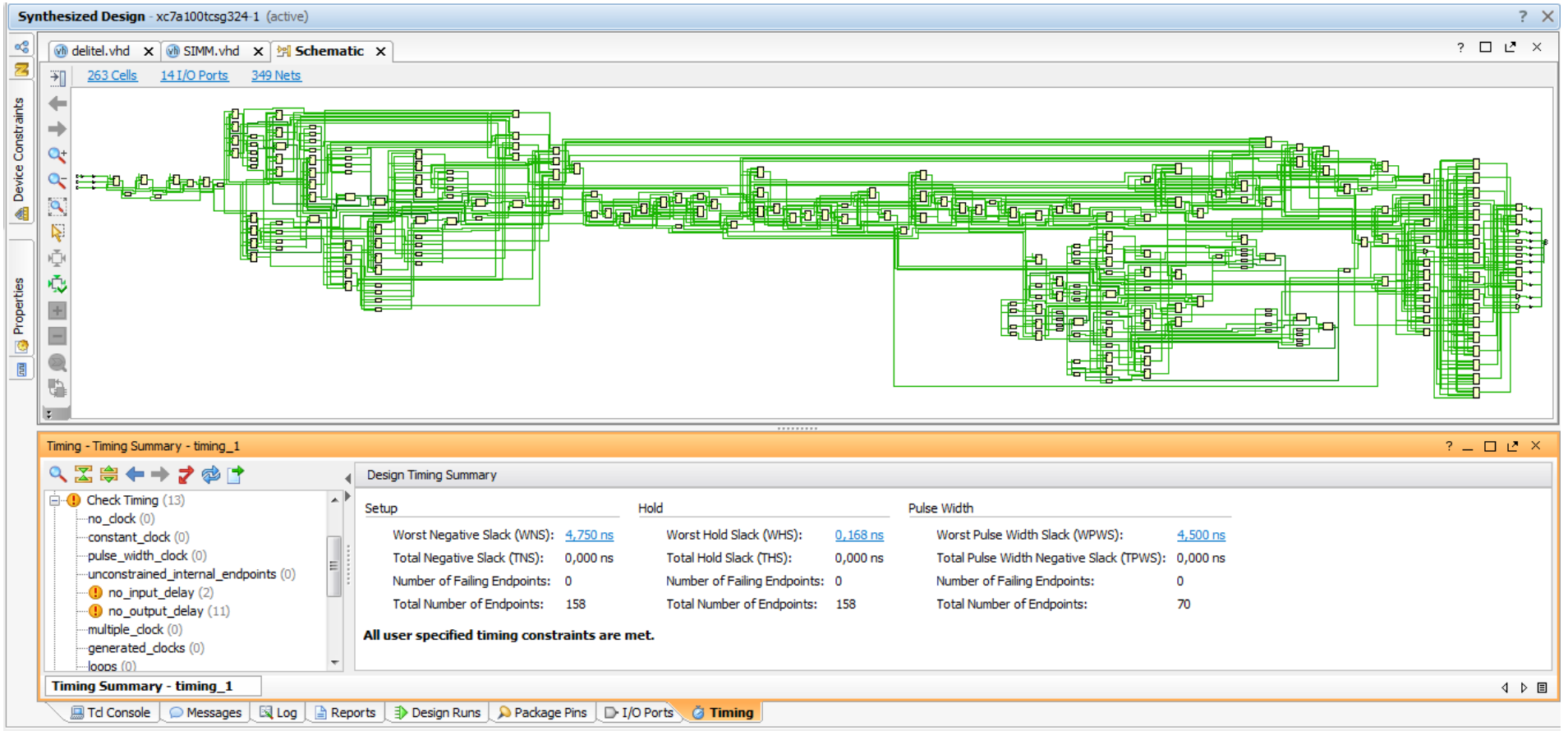

Рисунок 9 – Синтезированная схема и сводка по времени

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, в соответствии со схемой, указанной в Board Reference Manual.

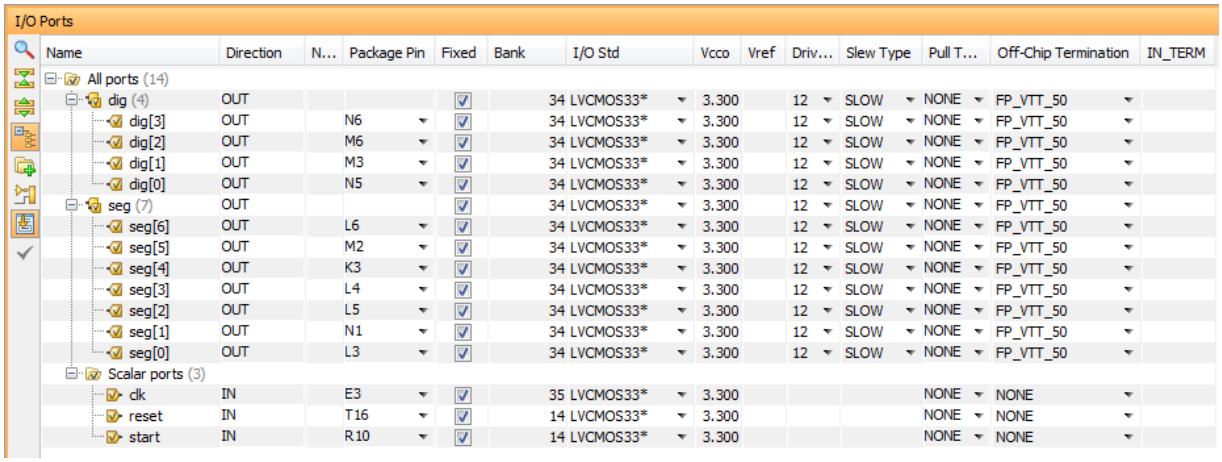

Рисунок 10 – Расположение портов ввода-вывода

5. Реализация

Следующий шаг разработки устройства – это реализация. После завершения процесса реализации нужно проверить выполнение временных ограничений (Report Timing Summary)(рисунок 11)

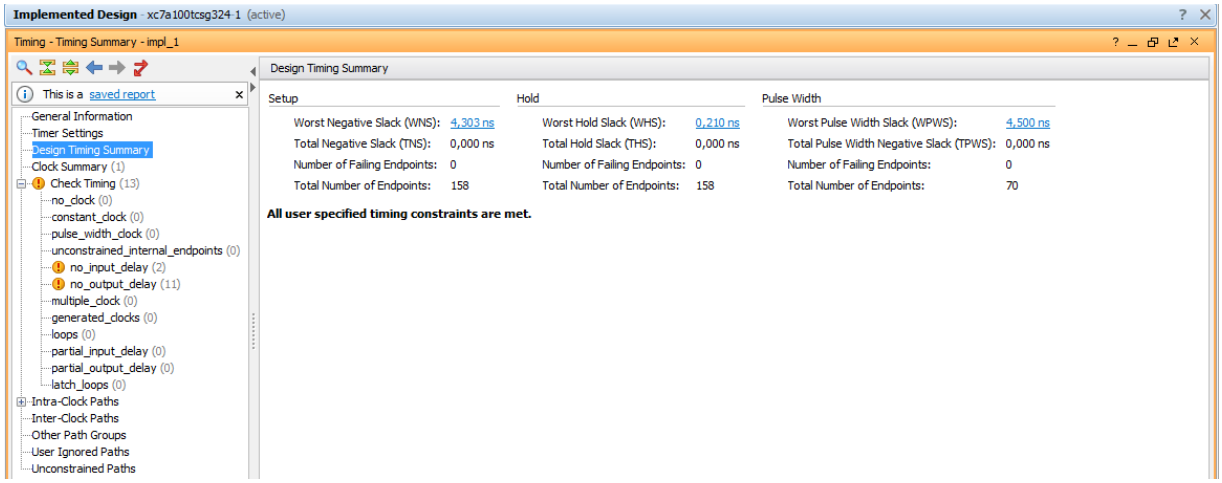

Рисунок 11 – временная сводка после реализации

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.

В окне Hardware Manager необходимо выбрать ”Open Target”, а затем ”Auto Connect” Если соединение будет успешным, надпись ”unconnected” сменится на имя платы, появится кнопка ”Program Device”. При нажатиии на нее нужно выбрать отладочную плату, а затем файл с битовым потоком (с расширением .bit) – ”Bitstream file” . Нажать ”Program”.

7. Проверка функционирования устройства на отладочной плате

После завершения программирования устройства была произведена проверка его работоспособности, в результате которой было выяснено, что устройство работает корректно. Функции, возложенные на кнопки reset, start-stop правильно работают, все описанные и требуемые заданием функции устройства работают.