ФГБОУ ВПО “Воронежский государственный
технический университет”

Кафедра полупроводниковой электроники
и наноэлектроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 1, 2
по дисциплине
«Проектирование цифровых устройств в базисе ПЛИС»
для студентов направления
210100.62 «Электроника и наноэлектроника»
(профиль Микроэлектроника
и твердотельная электроника»)
очной формы обучения

Воронеж 2014

Составители: д-р техн. наук А.В. Строгонов, аспирант П.С. Городков

УДК 621.382

Методические указания к выполнению лабораторных работ № 1, 2 по дисциплине «Проектирование цифровых устройств в базисе ПЛИС» для студентов направления 210100.62 «Электроника и наноэлектроника» (профиль «Микроэлектроника и твердотельная электроника») очной формы обучения / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. А.В. Строгонов, П.С. Городков. Воронеж, 2014. 18 с.

В методических указаниях приведены примеры по проектированию и отладке цифровых устройств в базисе ПЛИС серии Cyclone с использованием учебного лабораторного стенда LESO2.1 (Лаборатории электронных средств обучения, ЛЭСО ГОУ ВПО «СибГУТИ»).

Методические указания подготовлены в электронном виде в текстовом редакторе Word 2003 и содержатся в файле MYLeso.new.doc

Табл. 1. Ил. 10. Библиогр.: 5 назв.

Рецензент д-р техн. наук, проф. М.И. Горлов

Ответственный за выпуск зав. кафедрой
д-р физ.-мат. наук, проф. С.И. Рембеза

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ЛАБОРАТОРНАЯ РАБОТА № 1

Разработка проекта умножителя целых чисел без знака размерностью 4х4 в базисе ПЛИС типа ППВМ серии Cyclone фирмы Altera в САПР Quartus II

Цель работы: получение навыков разработки цифровых автоматов с использованием высокоуровневого языка описания аппаратных средств VHDL и редактора состояний цифрового автомата САПР ПЛИС Altera Quartus II.

Задание:

Разработайте функциональные блоки с использованием языка VHDL:

1) сдвиговый 4-разрядный регистр с асинхронным сигналом сброса (CLR) и синхронными сигналами загрузки (LD), сдвига (SH), выбора направления сдвига (DIR);

2) управляющий автомат на пять состояний на основе однопроцессного шаблона;

Разработайте функциональные блоки с использованием мегафункций САПР Altera Quartus II:

1) 8- разрядный сумматор;

2) шинный мультиплексор двух 8-разрядных сигналов в один;

3) 8-разрядный регистр-аккумулятор результата;

4) детектор нуля для случая, если множитель окажется равным нулю;

5) блоки памяти для хранения умножаемых чисел.

Разработайте с использованием редактора состояний (State Machine Viewer) цифровой автомат умножителя и извлеките в автоматическом режиме код языка VHDL. Переработайте проект умножителя с использованием полученных результатов.

Объедините разработанные функциональные блоки в иерархический проект.

Теоретические сведения

На рис. 1 и рис. 2 показаны верхний и нижний уровни иерархии проекта умножителя целых десятичных чисел (P=B(множимое)*A(множитель)=константа*A) размерностью 4x4 разряда. Сигнал A (множитель) следует рассматривать как число а сигнал B как константу (множимое). Умножитель на рис. 1 настроен на умножение двух чисел 10x10. Умножитель состоит из двух однотипных регистров ShiftN, сдвигающих влево или вправо в зависимости от сигнала DIR, задающего направление сдвига (пример 1), детектора нуля AllZero, управляющего автомата avt на пять состояний (пример 2), 8-разрядного сумматора на мегафункции lpm_add_sub, шинного мультиплексора на мегафункции lpm_mux и 8-разрядного регистра на мегафункции lpm_dff, выполняющего роль аккумулятора. Сдвиговый регистр ShiftN (DIR=0) на вход, которого подается число A, выполняющий функцию множителя, работает как преобразователь параллельного кода в последовательный, параллельный выход которого SRA[7..0] используется для детектирования нуля.

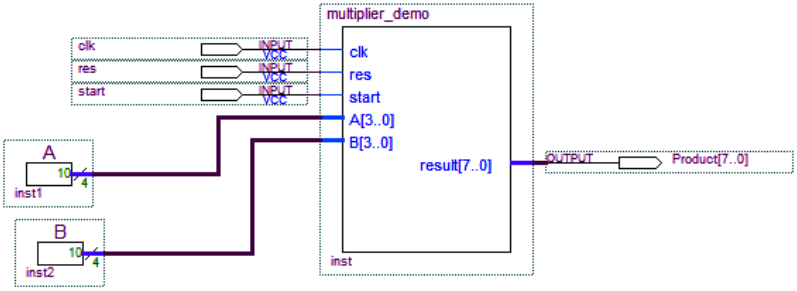


Рис. 1. Умножитель размерностью 4x4. Верхний уровень иерархии

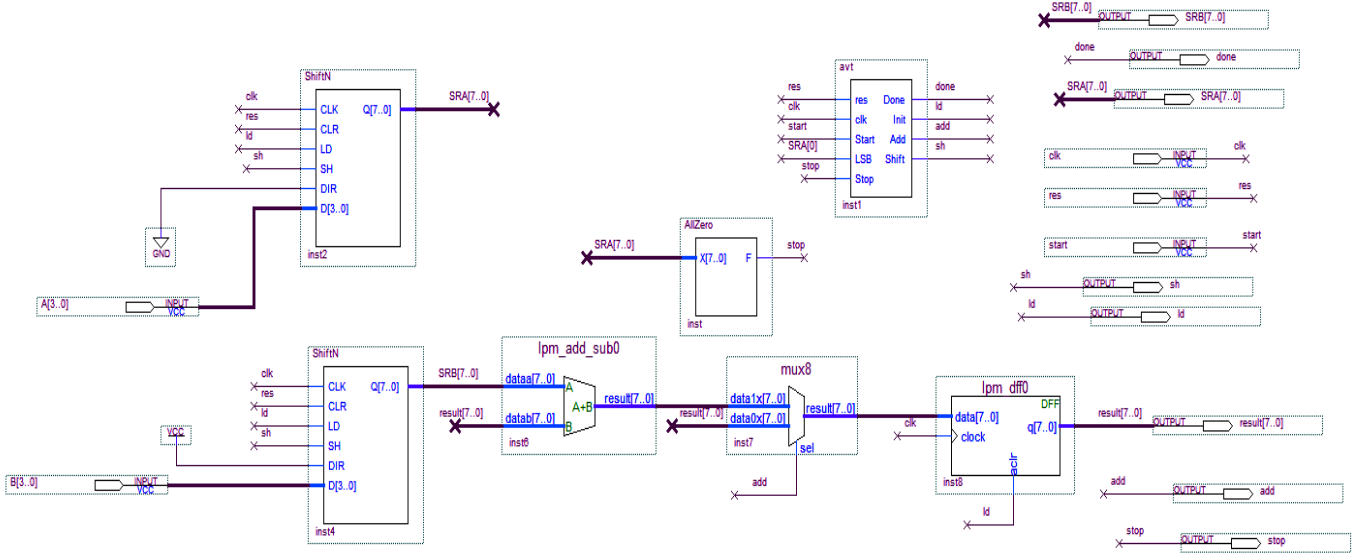


Рис. 2. Умножитель размерностью 4x4. Нижний уровень иерархии

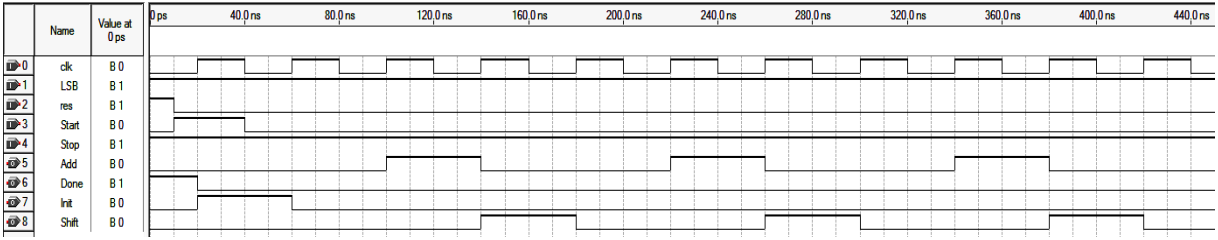


Рис. 3. Тест цифрового автомата

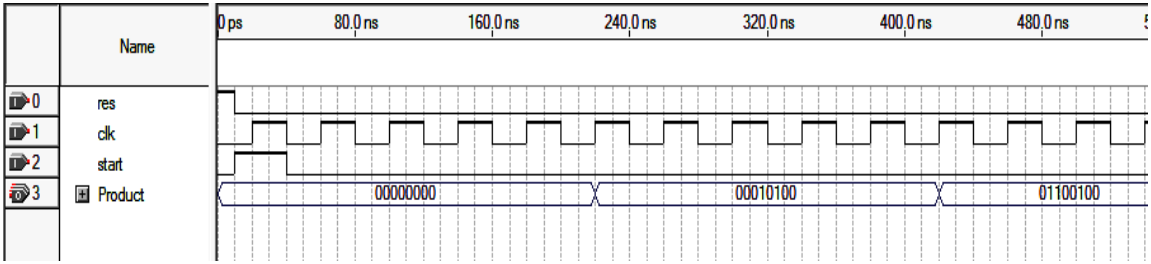
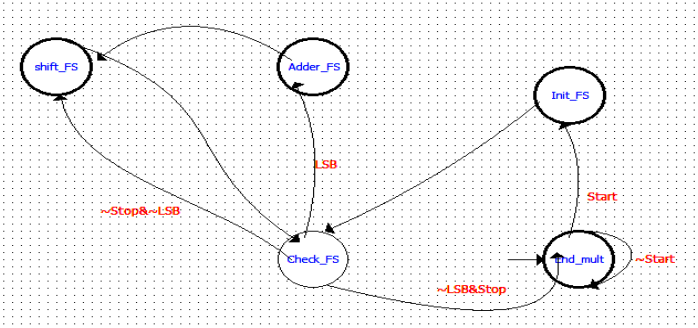
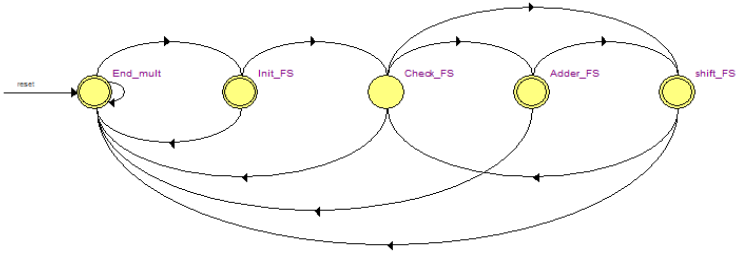


Рис. 4. Тестирование умножителя на примере умножения 10x10. Результат 100



а)



б)

Рис. 5. Граф-автомат, разработанный с помощью редактора состояний (а) и синтезированный граф-автомат (меню Netlist Viewers/State Machine Viewer)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity ShiftN is
port(CLK, CLR, LD, SH, DIR: in STD_LOGIC;
D: in std_logic_vector(3 downto 0);
Q: out std_logic_vector(7 downto 0));
end ShiftN;
architecture a of ShiftN is
  begin
  process (CLR, CLK)
  variable St: std_logic_vector(7 downto 0);
  subtype InB is natural range 3 downto 0;
       begin
            if CLR = '1' then
                 St := (others => '0'); Q <= St;
            elsif CLK'EVENT and CLK='1' then
                 if LD = '1' then
                       St:=(others=>'0');
                       St(InB) := D;
                       Q <= St;
                 elsif SH = '1' then
                       case DIR is
                      when '0' => St := '0' & St(St'LEFT downto 1);
                      when '1' => St := St(St'LEFT-1 downto 0) & '0';
                       end case;
                       Q <= St;
                 end if;
            end if;
       end process;
end a;
```

Пример 1. Сдвиговый регистр на языке VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY avt IS
   PORT (
      res,clk, Start, LSB, Stop : IN STD_LOGIC;
      Done : OUT STD_LOGIC;
      Init : OUT STD_LOGIC;
      Add : OUT STD_LOGIC;
      Shift : OUT STD_LOGIC);
END avt;
ARCHITECTURE BEHAVIOR OF avt IS
TYPE type_fstate IS (Check_FS,Init_FS,Adder_FS,shift_FS,End_mult);
   SIGNAL fstate : type_fstate;
   SIGNAL reg_fstate : type_fstate;
BEGIN
Init <='1' when reg_fstate = Init_FS else '0';
Add <='1' when reg_fstate = Adder_FS else '0';
Shift <='1' when reg_fstate = shift_FS else '0';
Done <='1' when reg_fstate = End_mult else '0';
process (clk, res) begin
if res = '1' then reg_fstate <= End_mult;
elsif clk'event and clk = '1' then
case reg_fstate is
when  Init_FS => reg_fstate <= Check_FS;
when Check_FS =>
                    if LSB = '1' then reg_fstate <= Adder_FS;
                    elsif Stop ='0' then reg_fstate <= shift_FS;
                    else reg_fstate <= End_mult;
                    end if;
when Adder_FS => reg_fstate <= shift_FS;
when shift_FS => reg_fstate <= Check_FS;
when End_mult => if Start = '1' then reg_fstate <= Init_FS; end if;
end case;
end if;
end process;
END BEHAVIOR;
```

Пример 2. Код языка VHDL управляющего автомата

Рис. 3 демонстрирует принцип работы управляющего автомата. Автомат принимает пять состояний с именами Check_FS, Init_FS, Adder_FS, shift_FS, End_mult. В каждом из состояний активным является один из сигналов Init, Add, Shift и Done. Автомат (пример 2) разработан по “классической” схеме, с использованием одного оператора Process (однопроцессный шаблон) для описания памяти состояний и логики переходов.

Автомат инициализируется высоким уровнем сигнала Start синхронизируемого синхросигналом clk переводящим выход Init в активное состояние. При высоком уровне сигнала Init происходит загрузка обоих сдвиговых регистров параллельным кодом.

Если на вход автомата LSB все время будет поступать логическая 1 (младший разряд SRA[0] 8-разрядного сигнала SRA[7..0]) с выхода сдвигового регистра ShiftN при DIR=0, то управляющий автомат будет вырабатывать сигналы “сложить” (Add) и “сдвинуть” (Shift). Это возможно, например, при загрузке в регистр числа 15D (1111BIN). На рис. 4 показан пример умножения чисел 10x10. Результат 100. По окончании процесса умножения вырабатывается сигнал готовности Done.

Рассмотрим разработку цифрового автомата умножителя (рис. 5) с помощью редактора состояний САПР Quartus II (State Machine Viewer). Для разработки проекта будем использовать версию Quartus II Web Edition version 9.1.

Далее извлечем код языка VHDL в автоматическом режиме. В синтезированном коде используется двухпроцессный шаблон. Первый оператор Process описывает блок регистров (память состояний) для хранения состояний автомата. Второй оператор Process используется для описания логики переходов и логики формирования выхода (пример 3). Тестирование умножителя на примере умножения 5x5 показано на рис. 6. Общие сведения по числу задействованных ресурсов в проекте показаны в таблице.

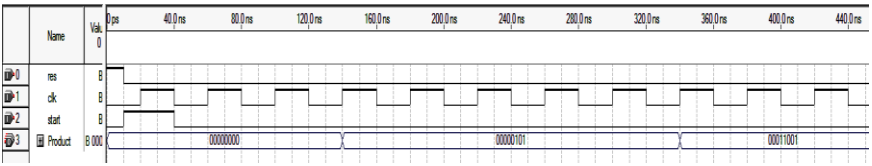


Рис. 6. Тестирование умножителя на примере умножения 5x5. Результат 25

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY avt_flow IS
PORT (
reset : IN STD_LOGIC := '0';
clock : IN STD_LOGIC;
Start : IN STD_LOGIC := '0';
LSB : IN STD_LOGIC := '0';
Stop : IN STD_LOGIC := '0';
Done : OUT STD_LOGIC;
Init : OUT STD_LOGIC;
Add : OUT STD_LOGIC;
Shift : OUT STD_LOGIC);
END avt_flow;
ARCHITECTURE BEHAVIOR OF avt_flow IS
TYPE type_fstate IS (Check_FS,Init_FS,Adder_FS,shift_FS,End_mult);
   SIGNAL fstate : type_fstate;
   SIGNAL reg_fstate : type_fstate;
BEGIN
   PROCESS (clock,reg_fstate)
   BEGIN
      IF (clock='1' AND clock'event) THEN
         fstate <= reg_fstate;
      END IF;
   END PROCESS;
   PROCESS (fstate,reset,Start,LSB,Stop)
   BEGIN
      IF (reset='1') THEN
         reg_fstate <= End_mult;
         Done <= '0'; Init <= '0';
         Add <= '0'; Shift <= '0';
      ELSE
         Done <= '0'; Init <= '0';
         Add <= '0'; Shift <= '0';
```

```
        CASE fstate IS
          WHEN Check_FS =>
            IF ((NOT((LSB = '1')) AND (Stop = '1'))) THEN
              reg_fstate <= End_mult;
            ELSIF ((LSB = '1')) THEN
              reg_fstate <= Adder_FS;
            ELSIF ((NOT((Stop = '1')) AND NOT((LSB = '1')))) THEN
              reg_fstate <= shift_FS;
            -- Inserting 'else' block to prevent latch inference
            ELSE
              reg_fstate <= Check_FS;
            END IF;
          WHEN Init_FS =>
            reg_fstate <= Check_FS;
            Init <= '1';
          WHEN Adder_FS =>
            reg_fstate <= shift_FS;
            Add <= '1';
          WHEN shift_FS =>
            reg_fstate <= Check_FS;
            Shift <= '1';
          WHEN End_mult =>
            IF ((Start = '1')) THEN
              reg_fstate <= Init_FS;
            ELSIF (NOT((Start = '1'))) THEN
              reg_fstate <= End_mult;
            -- Inserting 'else' block to prevent latch inference
            ELSE
              reg_fstate <= End_mult;
            END IF;
            Done <= '1';
WHEN OTHERS =>
            Done <= 'X';
            Init <= 'X';
            Add <= 'X';
            Shift <= 'X';
            report "Reach undefined state";
        END CASE;
      END IF;
    END PROCESS;
        END BEHAVIOR;
```

Пример 3. VHDL-код извлеченный в автоматическом режиме из граф-автомата созданного с помощью редактора состояний

в САПР Quartus II

Общие сведения по числу задействованных ресурсов ПЛИС типа ППВМ серии Cyclone EP1C3T144C8N

Логических элементов (Logic Cells, LC)	Триггеров логических элементов LC Registers	Таблиц перекодировок LUT-only LC	Рабочая частота в наихудшем случае Fmax, МГц
47	41	6	275

Вопросы для отчета

1. Основные режимы загрузки регистра?
2. Какие типы сумматоров вы знаете?
3. Какие функции выполняет цифровой автомат?
4. Поясните принцип работы мультиплексора?
5. Поясните принцип умножения целых положительных чисел методом сдвига и умножения

ЛАБОРАТОРНАЯ РАБОТА № 2

Аппаратная верификация проекта умножителя размерностью 4x4 в базисе ПЛИС с помощью учебного лабораторного стенда LESO2.1

Цель работы: получение навыков отладки цифровых устройств с помощью учебного лабораторного стенда LESO2.1.

Задание:

1) разработайте схему подавителя дребезга контактов с использованием суммирующего счетчика-делителя частоты;
2) подключите входные и выходные сигналы проекта к внешним выводам ПЛИС;

3) осуществите загрузку конфигурационного файла в ПЛИС с помощью программы-загрузчика;
4) проведите верификацию проекта на примере умножения как четных, так и нечетных чисел или их перестановки.

Теоретические сведения

Реализуем умножитель целых положительных чисел размерностью 4x4 разряда в базисе ПЛИС типа ППВМ серии Cyclone EP1C3T144C8N фирмы Altera с помощью учебного лабораторного стенда LESO2.1 (Лаборатории электронных средств обучения, ЛЭСО ГОУ ВПО «СибГУТИ») отечественной разработки. Учебный лабораторный стенд предназначен для обучения основам проектирования цифровой техники на основе ПЛИС.

Стенд подключается к персональному компьютеру через порт USB. Для записи файла конфигурации в память ПЛИС через порт USB персонального компьютера требуется преобразовать *.sof- файл в формат с расширением *.rbf. Загрузка конфигурационного файла в ПЛИС производится с помощью отдельной программы – загрузчика (l2flash.exe).

Входные и выходные контакты (сигналы) к внешним выводам ПЛИС подключены с помощью меню Assignments/Pins (рис. 7). Из-за того что стенд имеет 8 переключателей S1-S8 пришлось отказаться от загрузки чисел с внешних портов (4-х разрядные сигналы A и B) и от сигнала Done так как доступно всего лишь 8 светодиодов. Умножаемые числа предварительно сохраняются в константах (мегафукция LPM_constant). Светодиоды LED1-LED8 отображают результат умножения (8-разрядный сигнал Product[7..0]). В проекте принято, что LED8 (pin 121) это младший значащий разряд.

Для подачи тактовых импульсов с помощью кнопки Bottom необходимо использовать фильтр (блок Antitinkling).

Данный блок предназначен для подавления дребезга контактов. Из-за этого явления непосредственное подключение кнопки с механическим замыканием контактов к цифровой схеме не всегда допустимо. Суть дребезга заключается в многократном неконтролируемом замыкании и размыкании контактов в момент коммутации, в результате чего на цифровую схему подается множество импульсов вместо одного.

Частота тактового генератора в учебных стендах LESO2 равна 6 МГц, в стендах LESO2.1 и LESO2.3 – 50 МГц. Делитель частоты должен обеспечить интервал между импульсами больше, чем длительность дребезга и менее чем длительность нажатия кнопки. На рис. 8 показана схема подавителя дребезга с использованием суммирующего счетчика-делителя частоты. В нашем случае 19-разрядный счетчик обеспечивает коэффициент счета 524287 и выходной сигнал cout с пониженной частотой 95,37 Гц (100 Гц – период 10 мс). Время дребезга кнопки примерно составляет 2 мс.

На рис. 9 и 10 показано тестирование умножителя на примере умножения 5x5. Тестирование осуществляется следующим образом. Согласно рис.6 щелкаем переключателем S2 (pin 50) выполняющим роль асинхронного сигнала res. Переводим в верхнее положение переключатель S3 (pin 51) – сигнал start, далее нажимаем на кнопку Button (pin 37) один раз, происходит загрузка чисел в умножитель. Переводим переключатель S3 в нижнее положение. Щелкаем три (рис.9) и пять раз (рис. 10) кнопкой Button для имитации подачи синхросигнала. Итоговый результат умножения десятичное число 25, а процесс умножения осуществляется за 9 тактов синхрочастоты.

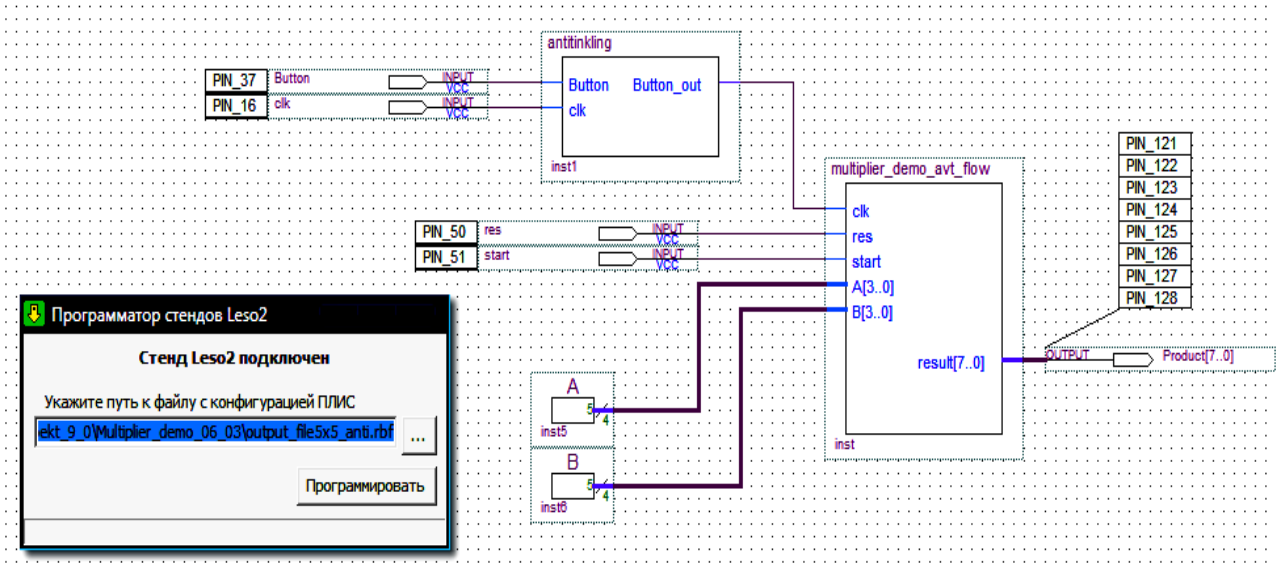
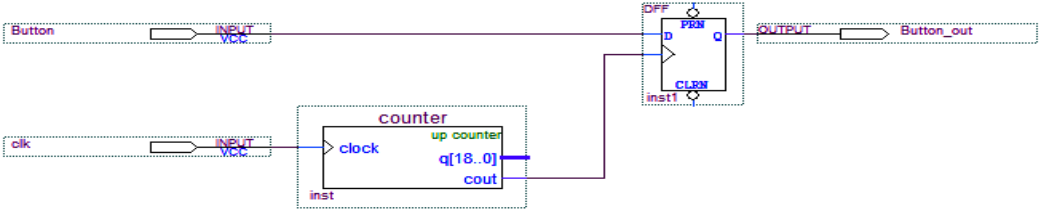
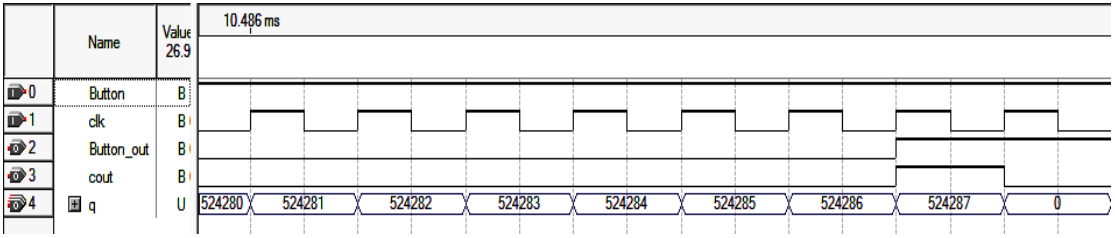


Рис. 7. Схема умножителя с подключенными внешними выводами



а)



б)

Рис. 8. Подавитель дребезга с использованием суммирующего счетчика-делителя частоты (а) и временные диаграммы его работы (б)

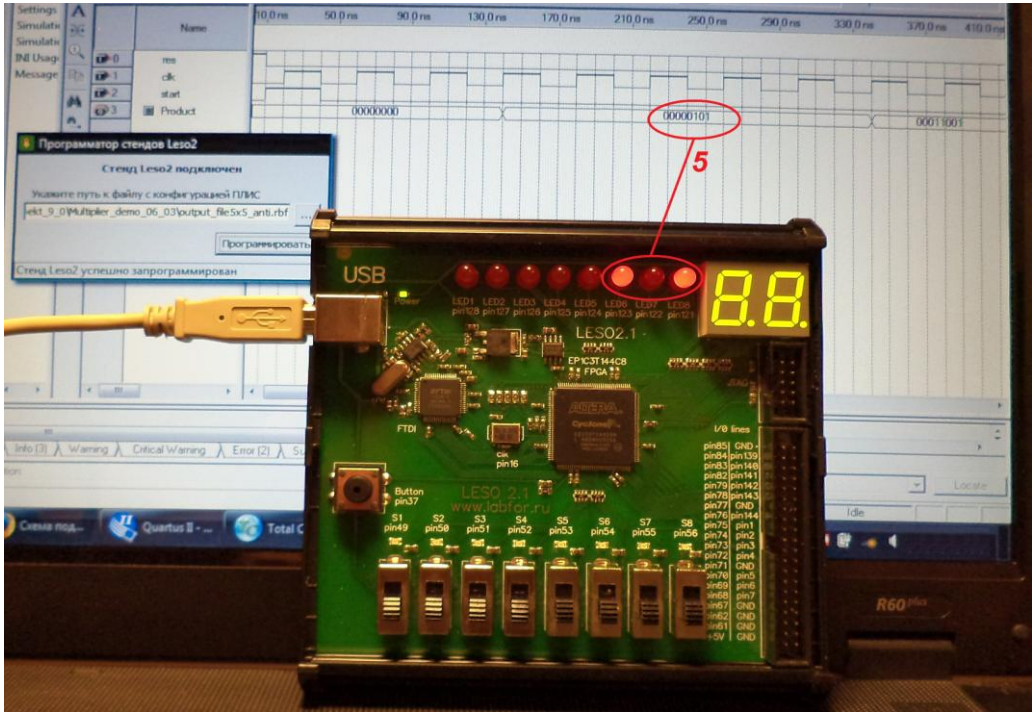


Рис. 9. Тестирование умножителя на примере умножения 5x5. Промежуточный результат 5

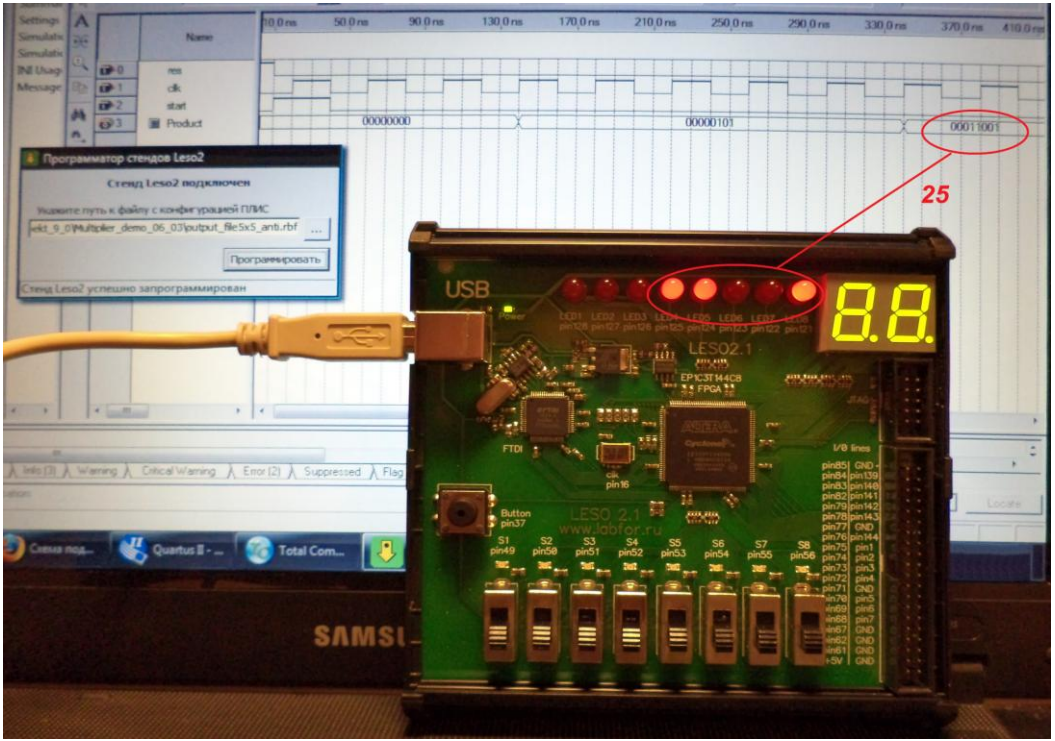


Рис. 10. Тестирование умножителя на примере умножения 5x5. Итоговый результат 25

Вопросы для отчета

1. Как осуществляется программирование ПЛИС?
2. Что такое дребезг контактов и как его подавить?
3. Перечислите основные функциональные блоки учебного лабораторного стенда LESO2.1
4. Как провести аппаратную верификацию проекта?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. www.labfor.ru. Учебный лабораторный стенд LESO 2.1. Паспорт и Инструкция по эксплуатации. Новосибирск. 2009.
2. Строгонов, А.В. Проектирование умножителя методом правого сдвига и сложения с управляющим автоматом в базисе ПЛИС [Текст] / А.В. Строгонов, А.В. Быстрицкий // Компоненты и технологии. - 2013. - N12. - С.6-10.
3. Строгонов А.В. Проектирование умножителя целых чисел со знаком методом правого сдвига и сложения в базисе ПЛИС [Текст] / А.В. Строгонов, А.В. Быстрицкий // Компоненты и технологии. - 2014. - N1. - С.94-100.
4. Строгонов А.В. Цифровая обработка сигналов в базисе программируемых логических интегральных схем: учеб. пособие [Электронный ресурс]. - Электрон. текстовые и граф. данные (39 Мб). - Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2014.

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 1, 2
по дисциплине «Проектирование цифровых устройств в базисе ПЛИС» для студентов направления
210100.62 «Электроника и наноэлектроника»
(профиль «Микроэлектроника и твердотельная электроника»)
очной формы обучения

Составители:
Строгонов Андрей Владимирович
Городков Павел Сергеевич

В авторской редакции

Компьютерный набор А.В. Строгонова

Подписано к изданию 15.09.2014.
Уч.-изд. л. 1,1.

ФГБОУ ВПО “Воронежский государственный технический университет”
394026 Воронеж, Московский просп., 14