Московский государственный технический университет
имени Н.Э. Баумана

Методические указания

А.Ю. Попов
С.В. Ибрагимов

Лабораторная работа №2

Проектирование цифровых устройств на основе ПЛИС

Москва 2022

Оглавление

Цель работы: закрепление на практике теоретических сведений, полученных при изучении методики проектирования цифровых устройств на основе программируемых логических интегральных схем (ПЛИС), получение необходимых навыков работы с системой автоматизированного проектирования ISE WebPack устройств на основе ПЛИС фирмы Xilinx, изучение аппаратных и программных средств моделирования, макетирования и отладки устройств на основе ПЛИС.

Для выполнения работы студенту необходимо ознакомиться с архитектурой ПЛИС FPGA Spartan 3 производства фирмы Xilinx, изучить методику проектирования устройств на основе ПЛИС с использованием САПР ISE WebPack (версии 8 или старше), спроектировать и реализовать с помощью набора XC3S200 (или набора Nexys2 на основе ПЛИС XC3S500E) устройство управления счетом и индикацией состояния 16-разрядного счетчика.

Описание проектируемого устройства

В данной лабораторной работе осваивается методика проектирования цифровых устройств на примере разработки и реализации на ПЛИС схемы управления счетом и индикацией состояния 16-разрядного счетчика. Отладка устройства производится с помощью набора XC3S200 (или набора Nexys2 на основе ПЛИС XC3S500E с аналогичными функциями), который содержит матрицу 7-сегментных индикаторов и кнопки, необходимые для управления разрабатываемым устройством. Выбор набора (XC3S200 или Nexys2) определяется в соответствии с вариантом в таблице 1 (см. далее). Состав устройства и назначение используемых ресурсов отладочного набора XC3S200 показаны на рисунке 1 (набор Nexys2 содержит аналогичные ресурсы).

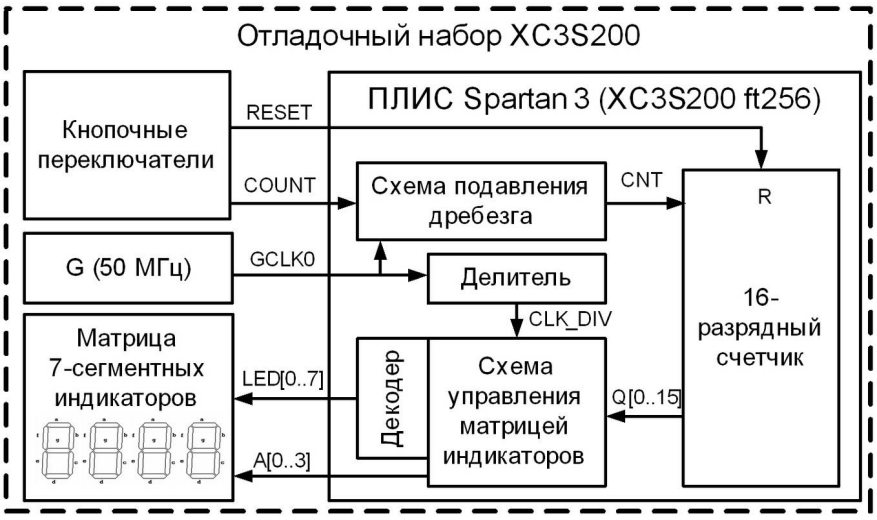

Рисунок 1 – Использование отладочного набора XC3S200 для реализации схемы управления и индикацией 16-ти разрядного счетчика

В устройстве используется синхронный 16-разрядный счетчик с асинхронным сбросом. Для управления счетом и сбросом используются две кнопки, входящие в состав отладочного набора (линии RESET и COUNT). При этом учитывается, что подача внешнего сигнала управления счетом COUNT непосредственно на вход синхронизации счетчика привела бы к многочисленным ложным срабатываниям из-за наличия дребезга при замыкании и размыкании кнопки. Для устранения этого в работе используется схема подавления дребезга, выдающая на счетчик сигнал CNT.

Состояние 16-разрядного счетчика передается на схему управления матрицей индикаторов, которая обеспечивает мультиплексированную передачу тетрад данных на декодер 7-сегментого кода, а также сопровождает выдачу данных сигналами управления анодами (A[0..3]). На выходе декодера формируется код активизации сегментов (LED[0..7]),

передаваемый непосредственно на 4 индикатора отладочного набора. Делитель частоты должен выдавать сигнал разрешения (CLK_DIV) таким образом, чтобы каждая цифра активировалась с частотой (100-200 Гц).

Схема подавления дребезга

Схема подавления дребезга представляет собой автомат, воспринимающий входной сигнал COUNT от кнопки и выдающий выходной сигнал CNT в соответствии с приведенной на рисунке 2 диаграммой.

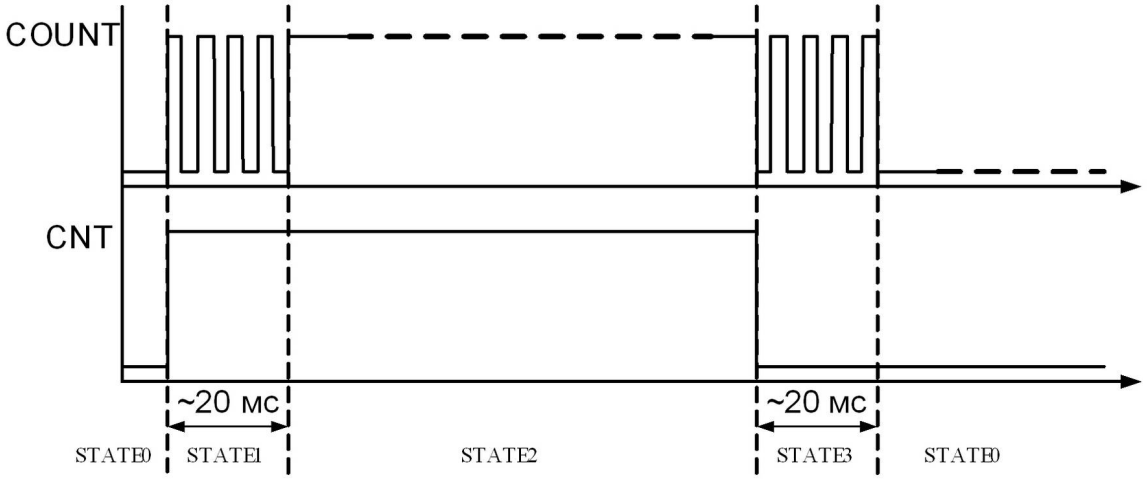

Рисунок 2 – Диаграмма работы схемы подавления дребезга

Кнопки, имеющиеся в наличие на плате XC3S200 или Nexys2, обладают дребезгом и не снабжены схемами их подавления (триггерами Шмидта и т.д.). Как при нажатии, так и при отпускании кнопки происходит многократное изменение уровня напряжения на линии COUNT, вызванное упругими соударениями. Для предотвращения нежелательных многократных срабатываний устройств, следует построить схему, исключающую возможность прохождения сигналов в момент дребезга. Это можно осуществить с помощью дополнительного счетчика, исполняющего роль схемы задержки на длительность переходных процессов. При подаче на вход данного счетчика сигнала отладочного набора GCLK0, имеющего частоту 50 МГц, в качестве сигнала, информирующего об истечении времени задержки, может быть использован сигнал DLY_EN равный лог. «1» только при достижении счетчиком значения 2**20-1. После окончания счета необходимо выполнить сброс счетчика в исходное нулевое состояние.

Автомат, реализующий указанную логику работы, может находиться в одном из четырех состояний: ожидания нажатия (STATE0), счет времени после нажатии (STATE1),

ожидание отпускания (STATE2), счет времени после отпускания (STATE3). Диаграмма переходов состояния показана на рисунке 3.

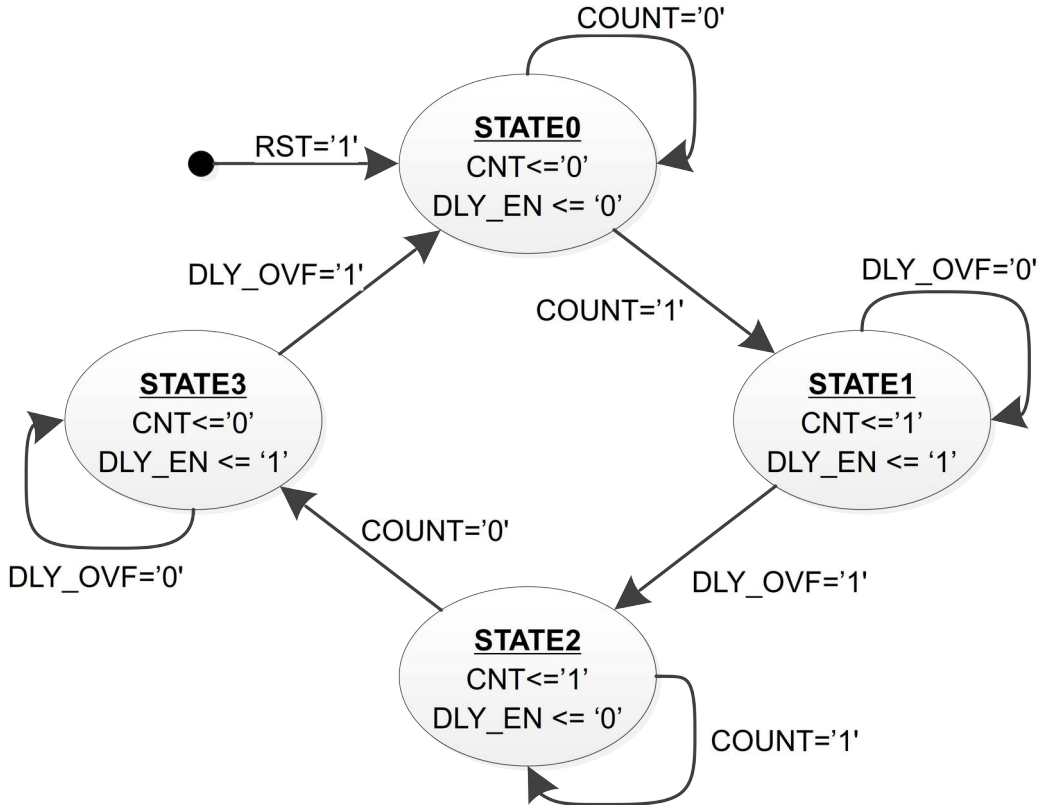

Рисунок 3 – Диаграмма состояния автомата подавления дребезга

Схема подавления дребезга представляет собой сочетание автомата, реализующего диаграмму, показанную на рисунке 3, и исполнительное устройство, представляющие собой счетчик и схему сравнения. Входными сигналами автомата являются сигнал COUNT (сигнал от кнопки с дребезгом) и DLY_OVF, равный лог. «1» при достижении счетчиком значения 2**20-1, и сигнализирующий об истечении времени задержки. Выходными сигналами автомата являются сигнал CNT (сигнал кнопки, очищенный от дребезга) и сигнал управления счетчиком DLY_EN.

Счетчик, входящий в состав исполнительного устройства, используется для реализации задержки на время дребезга. В том случае, когда разрешающий сигнал DLY_EN, формируемый автоматом, равен лог. «1», выполняется счет, в противном случае — счетчик сбрасывается в нулевое состояние. После достижения счетчиком значения 2**20-1, в схеме сравнения формируется сигнал DLY_OVF=1, приводящий к переключению автомата в состояние STATE0 или STATE2.

Текущее состояние счетчика хранится в регистре, состоящем из 2-х D-триггеров. Выходы D-триггеров объединены в вектор S, значение которого соответсвует текущему

состоянию автомата. Информационные входы D-триггеров объединены в вектор SN. Таким образом, значение сигнала SN в текущем такте соответсвует состоянию, в которое автомат перейдет в следующем такте. Сигнал SN вырабатывается комбинационными схемами в зависимости от текущего состояния и входных сигналов автомата. Значения выходных сигнналов автомата вырабатываются комбинационными схемами в зависимости от текущего состояния автомата. Детальные временные диаграммы, иллюстрирующие переключение сигнала CNT из лог. «0» в лог. «1» и наоборот приведены соответственно на рисунках 4 и 5.

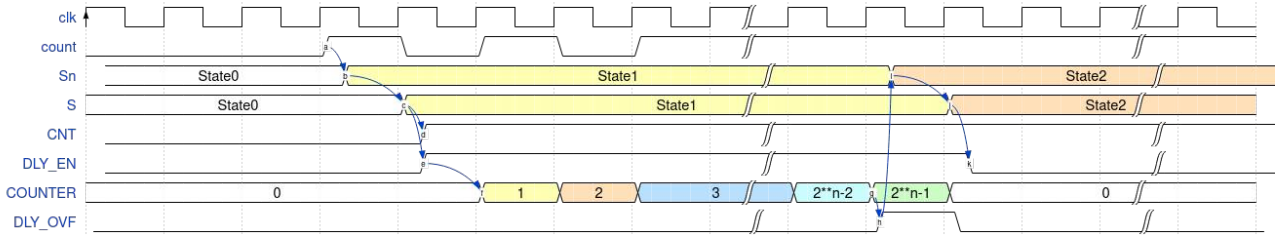

Рисунок 4 – Диаграмма работы схемы подавления дребезга (переключение из «0» в «1»)

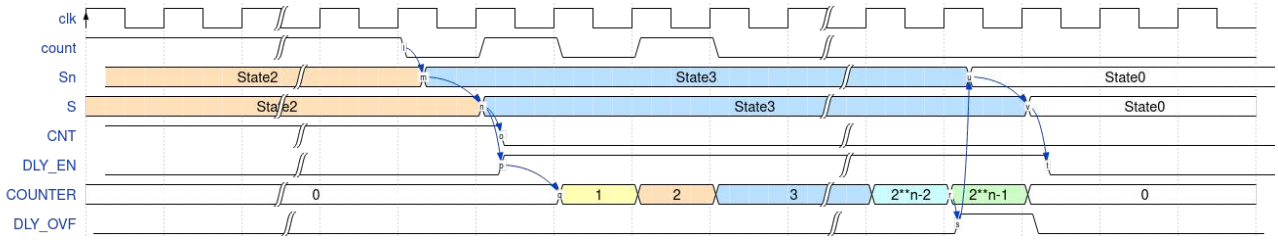

Рисунок 5 – Диаграмма работы схемы подавления дребезга (переключение из «1» в «0»)

Структура лабораторного проекта в Xilinx ISE

Проект представляет собой дерево описаний модулей и файлов с директивами САПР, представленных в различных форматах:

- Исходные описания устройств *.vhd (для языка VHDL), *.v (для языка Verilog), *.sv (для языка System Verilog) и т.д.

- Файлы пользовательских ограничений *.ucf, в которых описываются требования к назначению портам контактов микросхемы ПЛИС, специальные требования по размещению компонентов проекта, временные ограничения для сигналов синхронизации и сигналов данных, и другие.

- Исходные описания модулей, созданные в редакторе схем *.sch, которые целесообразно использовать лишь в ограниченных случаях (например, для модуля верхнего уровня).

Проект в данной лабораторной работе будет состоять из следующей иерархии описаний:

- Модуль верхнего уровня, представляющий структурное описание всего проекта (необходимо разработать самостоятельно). Функциональная схема модуля представлена на рисунке 1.
  - Модуль подавления дребезга на языке VHDL (необходимо разработать по индивидуальному заданию).
  - Модуль управления матрицей индикаторов (предоставлен готовый модуль, файл sevenseg_mux.vhd)
  - Модуль преобразования двоичного кода в код активации 7-сегментов (предоставлен готовый модуль, файл sevenseg_decoder.vhd)
  - Файл пользовательских ограничений *.ucf. Для этого необходимо модифицировать готовый файл main_xc3s200.ucf по таблице 5 (см. далее)

Задание 1. Выполнить кодирование состояний автомата, представленного на рисунке 3, в соответствии с индивидуальным вариантом из таблицы 1. Для этого заполнить таблицу 2 (состояние автомата представляется с помощью пары компонентов вектора S – S(1) и S(0)).

Таблица 1 – Индивидуальные варианты кодирования состояний автомата

Вариант:	Набор	Двоичный код состояния S(1),S(0)			
		State0	State1	State2	State3
1	XC3S200	00	01	10	11
2	XC3S200	00	01	11	10
3	Nexys 2-1200E	00	10	01	11
4	XC3S200	00	10	11	01
5	XC3S200	00	11	10	01
6	Nexys 2-500E	00	11	01	10
7	XC3S200	01	00	10	11
8	XC3S200	01	00	11	10
9	Nexys 2-1200E	01	10	00	11
10	XC3S200	01	10	11	00
11	XC3S200	01	11	00	10
12	Nexys 2-500E	01	11	10	00
13	XC3S200	10	00	01	11
14	XC3S200	10	00	11	01
15	Nexys 2-1200E	10	01	00	11
16	XC3S200	10	01	11	00
17	XC3S200	10	11	00	01
18	Nexys 2-500E	10	11	01	00

Вариант:	Набор	Двоичный код состояния S(1),S(0)			
		State0	State1	State2	State3
19	XC3S200	11	00	01	10
20	XC3S200	11	00	10	01
21	Nexys 2-1200E	11	01	00	10
22	XC3S200	11	01	10	00
23	XC3S200	11	10	00	01
24	Nexys 2-500E	11	10	01	00

По таблице выходов 2 определить функции сигналов управления: CNT = f(S(1),S(0)), DLY_EN = f(S(1),S(0)). Результаты занести в отчет.

Таблица 2 – Таблица выходов

Состояние	State0	State1	State2	State3
Двоичный код состояния S(1),S(0)				
CNT	0	1	1	0
DLY_EN	0	1	0	1

Для определения состояния автомата в следующем такте (сигналы SN(*)) необходимо заполнить таблицу 3, после чего - определить и провести минимизацию с помощью карт Карно функций SN(0)=f(DLY_OVF,COUNT,S1,S0) и SN(1)=f(DLY_OVF,COUNT,S1,S0). Результаты занести в отчет. При заполнении таблицы 3 следует иметь в виду, что символ ‘x’ обозначает одновременно и 0 и 1, то есть, каждая строка таблицы соответствует паре клеток карты Карно.

Таблица 3 – Сигналы SN(*) и D*

COUNT	DLY_OVF	S1(t)	S0(t)	S1(t+1)	S0(t+1)	SN(1)	SN(0)	Описание события
0	x							Ожидание нажатия кнопки
1	x							Нажатие кнопки
x	0							Ожидание окончания счета
x	1							Конец счета
1	x							Ожидание отпускания
0	x							Отпускание кнопки
x	0							Ожидание окончания счета
x	1							Конец счета

Задание 2. Разработайте текстовое описание модуля в соответствии с полученными функциями DLY_EN, CNT, SN(0), SN(1) на основе следующего шаблона:

```
--  Пример модуля подавления дребезга 10 мс.
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity lab2_example IS
    PORT (
                RST:    IN              STD_LOGIC; --Системный сигнал сброса
                CLK:    IN              STD_LOGIC; --Сигнал синхронизации
                COUNT: IN               STD_LOGIC; --Сигнал кнопки с дребезгом
                CNT:    OUT             STD_LOGIC  --Сигнал кнопки, очищенный от дребезга
        );
END lab2_example;
ARCHITECTURE behavioral OF lab2_example IS
        -- Кодируем состояния в соответствии с вариантом
        CONSTANT STATE0: STD_LOGIC_VECTOR (1 downto 0) := "00";
        CONSTANT STATE1: STD_LOGIC_VECTOR (1 downto 0) := "01";
        CONSTANT STATE2: STD_LOGIC_VECTOR (1 downto 0) := "10";
        CONSTANT STATE3: STD_LOGIC_VECTOR (1 downto 0) := "11";
        -- Состояние автомата в момент времени t
        SIGNAL S:               STD_LOGIC_VECTOR (1 downto 0);
        -- Состояние автомата в момент времени t+1
        SIGNAL SN:              STD_LOGIC_VECTOR (1 downto 0);
        SIGNAL COUNTER: integer; -- Счетчик 2^20
        SIGNAL DLY_OVF: STD_LOGIC; -- Сигнал "Завершение счета"
        SIGNAL DLY_EN: STD_LOGIC; -- Сигнал разрешения работы счетчика
BEGIN
        -- Память состояний
        FSM_STATE_inst: PROCESS (CLK)
        BEGIN
                IF (CLK='1' and CLK'event) THEN
                        IF (RST='1') THEN
                                S <= STATE0;
                        ELSE
                                S <= SN;
                        END IF;
                END IF;
        END PROCESS;
        -- Комбинационная схема для выработки сигналов CNT и DLY_EN (по индивидуальному варианту)
        CNT <= S(1) xor S(0);
        DLY_EN <= S(0);
        --Комбинационные схемы для определения следующего состояния (по индивидуальному варианту)
        SN(0) <= (DLY_OVF and S(0)) or S(1);--пример 1
        SN(1) <= '1' WHEN ((DLY_OVF='0' and  S(0) = '1') or S(1) = '0') ELSE '0';--пример 2


        -- Описание счетчика
        COUNTER_inst: PROCESS (CLK)
        BEGIN
                IF (CLK='1' and CLK'event) THEN
                        IF (RST='1' or DLY_EN = '0') THEN
                                COUNTER <= 0;
                        ELSE
                                COUNTER <= COUNTER + 1;
                        END IF;
                END IF;
        END PROCESS;
        DLY_OVF <= '1' WHEN COUNTER = 2**20-1 ELSE '0'; --Длительность задержки


END Behavioral;
```

Собрать на основе полученного описания проект в САПР ISE. Для этого выполнить следующие действия:

1. Запустить САПР ISE.
2. В меню File выбрать пункт New Project.
3. Указать название и путь к файлу создаваемого проекта.
4. Указать тип модуля верхнего уровня: Schematic. Нажать кнопку Next.
5. В поле Product Category указать: All.
6. В поле Family указать: Spartan3 для набора XC3S200 или Spartan3E для набора Nexys2.
7. В поле Device указать: XC3S200 (для набора XC3S200), XC3S1200E (для набора Nexys 2-1200E) или XC3S500E (для набора Nexys 2-500E).
8. В поле Package указать: FT256 для набора XC3S200 или FG320 для набора Nexys2-1200E и FG320 для набора Nexys2-500E.
9. В поле Speed указать: -5.
10. В поле Synthesis Tool указать: XST(VHDL/Verilog).
11. В поле Simulator указать: Modelsim-SE VHDL.
12. В поле Preferred Language: VHDL. Трижды нажать кнопку Next, Нажать кнопку Finish. Необходимо указать путь к симулятору Mentor Modelsim в меню Edit → Preferences → ISE General → Integrated Tools → Model Tech Simulator: «C:\Altera\ 13.0sp1\modelsim_ase\win32aloem\modelsim.exe».
13. Создать новое описание модуля VHDL (VHDL Module), выбрав в меню Project пункт New Source. Далее выбрать тип описания (VHDL Module) и указать имя и путь к создаваемому файлу описания. Нажать кнопки Next и Finish. В итоге будет создан и открыт для редактирования файл описания, в котором необходимо создать описание модуля.

Задание 3. В интегрированном редакторе тестов САПР Xilinx ISE разработать тест для полученного устройства и выполнить моделирование его работы в симуляторе Modelsim. Для этого необходимо создать новое описание теста проекта, выбрав в меню Project пункт New Source. Далее выбрать тип описания (Test Bench WaveForm) и указать имя и путь к создаваемому файлу теста. Далее нажать на кнопку Next и в открывшемся диалоге выбрать тестируемое описание, после чего нажать на кнопки Next и Finish. В результате будет создан файл графического тестового воздействия для выбранного описания и будет вызван диалог

его настройки. В этом диалоге необходимо указать сигнал синхронизации (например, CLK) и время тестирования в поле Initial Length of Test Bench (должно составить 100000нс). Остальные параметры можно оставить неизменными.

Изменяя значения входных сигналов проверяемого модуля, необходимо создать тестовый сигнал подобный тому, что приведен на рисунке 2.

После создания теста и его сохранения можно приступать к моделированию, для чего активизировать вкладку Sources в окне Sources. В выпадающем списке Sources For выбрать строку Behavioral Simulation и выбрать запускаемый тест в дереве описаний проекта.

Необходимо также указать длительность симуляции (по умолчанию установлена длительность: 1000 наносекунд). Для этого во вкладке Processes окна Processes выбрать пункт ModelSim Simulator и нажмите правой кнопкой на пункт Simulate behavioral model. Далее выберите пункт Properties и в поле Simulation Run Time укажите значение 100000 наносекунд.

После этого можно начать моделирование. Для этого во вкладке Processes окна Processes выбрать пункт ModelSim Simulator и пункт Simulate behavioral model.

Заметим, что реальные переходные процессы в кнопках имеют большую длительность (порядка 10мс). Такие длительные процессы не рационально моделировать в их непосредственном виде в связи с большим расходом оперативной памяти, которая требуется для такого моделирования. Однако, мы можем уменьшить время ожидания окончания переходного процесса, уменьшив константу, по достижении которой формируется сигнал DLY_OVF, с 2**20-1 до 2**7-1. Исправление необходимо внести в строку, помеченную комментарием «Длительность задержки». Делать данную правку необходимо ТОЛЬКО ВО ВРЕМЯ МОДЕЛИРОВАНИЯ, после окончания которого необходимо вернуть все в начальное состояние.

Описание устройства и результаты моделирования занести в отчет.

Разработка устройств управления матрицей четырех 7-сегментных индикаторов

Для индикации информационных сигналов большой разрядности при отладке устройств с помощью набора XC3S200 или наборов Nexys2 целесообразно использовать имеющиеся четыре 7-сегментных индикатора. Управление их работой осуществляется благодаря подаче восьми общих сигналов управления сегментами LED[0..7] одновременно с установкой в активный низкий уровень сигнала выборки анода A[0..3].

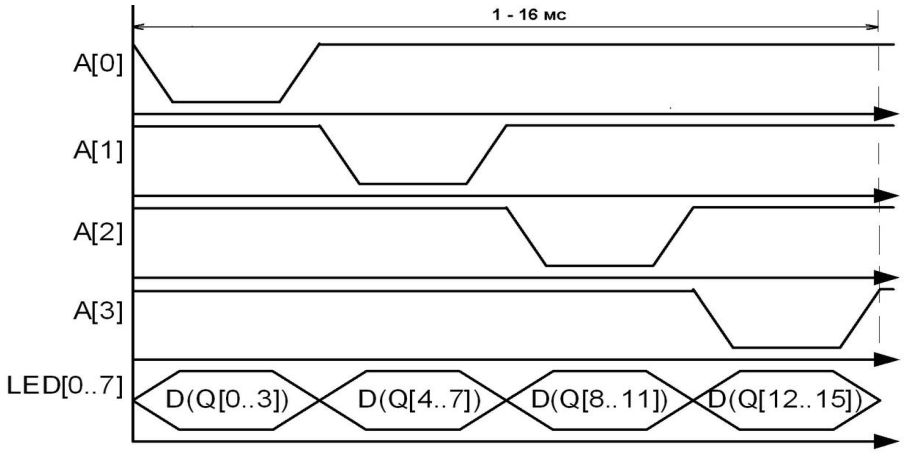

Рисунок 6 – Диаграмма работы устройства управления четырьмя 7-сегментными индикаторами

Таким образом, состояние линий LED должно устанавливаться схемой управления в соответствии с активным в данный момент 7-сегментным индикатором. Диаграмма работы устройства управления четырьмя 7-сегментными индикаторами показана на рисунке 6. Четыре тетрады сигналов данных (Q[0..3], Q[4..7], Q[8..11], Q[12..15]) должны быть преобразованы в код активизации индикаторов с помощью декодера. Темп активизации индикаторов должен обеспечивать отсутствие видимого мерцания сегментов (100-200 Гц).

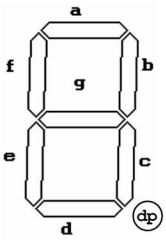

Рисунок 7 - Назначение сегментов индикатора

Для задания частоты активизации CLK_DIV целесообразно использовать дополнительный 16-разрядный счетчик – делитель частоты (период активизации четырех индикаторов должен составлять от 1 до 16 мс, частота от 1KHz до 60 Hz).

Для преобразования четырех информационных сигналов в код активизации восьми светодиодов используется табличное перекодирование, реализуемое программой синтеза при помощи LUT-таблиц. Расположение сегментов индикатора показано на рисунке 7, а коды их активизации для различных значений входного четырехразрядного слова D[0..3] указаны в таблице 4 (нулевой разряд означает активизацию сегмента; сегмент DP не активизируется).

Таблица 4 – Значения информационных входов активизации сегментов 7-сегментного индикатора для возможных значений информационного слова D[0..3]

D[0..3]	DP	A	B	C	D	E	F	G
0000	1	0	0	0	0	0	0	1
0001	1	1	0	0	1	1	1	1

D[0..3]	DP	A	B	C	D	E	F	G
0010	1	0	0	1	0	0	1	0
0011	1	0	0	0	0	1	1	0
0100	1	1	0	0	1	1	0	0
0101	1	0	1	0	0	1	0	0
0110	1	0	1	0	0	0	0	0
0111	1	0	0	0	1	1	1	1
1000	1	0	0	0	0	0	0	0
1001	1	0	0	0	0	1	0	0
1010	1	0	0	0	1	0	0	0
1011	1	1	1	0	0	0	0	0
1100	1	0	1	1	0	0	0	1
1101	1	1	0	0	0	0	1	0
1110	1	0	1	1	0	0	0	0
1111	1	0	1	1	1	0	0	0

Задание 4. Разработать устройство управления, принимающее 16-разрядное информационное слово Q[0..15] и управляющее их последовательной выдачей по шине D[0..3] на декодер 7-сегментных индикаторов в соответствии с показанной на рисунке 6 диаграммой. Для этого создать файл VHDL, содержащий следующий текст описания:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY Seven_Segment_Driver IS
  PORT(
       CLK            : IN          std_logic;
       CLK_DIV        : IN          std_logic;
       Q              : IN          std_logic_vector(15 DOWNTO 0);
       RST            : IN          std_logic;
       D              : OUT         std_logic_vector(3 DOWNTO 0);
       A              : OUT         std_logic_vector(3 DOWNTO 0));
END ENTITY Seven_Segment_Driver;

ARCHITECTURE Struct OF Seven_Segment_Driver IS
       --Internal Anode
   SIGNAL      A_int : std_logic_vector(3 DOWNTO 0);
BEGIN

--Output Anode
A <= A_int;

A_drive: PROCESS (CLK,RST)
BEGIN
  IF (RST = '1') THEN
     A_int<="1110";
  ELSIF (CLK'EVENT AND CLK='1') THEN
       IF (CLK_DIV='1') THEN
```

```
               A_int(3)<=A_int(2);
               A_int(2)<=A_int(1);
               A_int(1)<=A_int(0);
               A_int(0)<=A_int(3);
        END IF;
  END IF;
END PROCESS A_drive;

D(0)  <= (Q(0)  AND NOT(A_int(0)))
            OR (Q(4)  AND NOT(A_int(1)))
            OR (Q(8) AND NOT(A_int(2)))
            OR (Q(12) AND NOT(A_int(3)));

D(1)  <= (Q(1)  AND NOT(A_int(0)))
            OR (Q(5)  AND NOT(A_int(1)))
            OR (Q(9) AND NOT(A_int(2)))
            OR (Q(13) AND NOT(A_int(3)));

D(2)  <= (Q(2)  AND NOT(A_int(0)))
            OR (Q(6)  AND NOT(A_int(1)))
            OR (Q(10) AND NOT(A_int(2)))
            OR (Q(14) AND NOT(A_int(3)));

D(3)  <= (Q(3)  AND NOT(A_int(0)))
            OR (Q(7)  AND NOT(A_int(1)))
            OR (Q(11) AND NOT(A_int(2)))
            OR (Q(15) AND NOT(A_int(3)));

END ARCHITECTURE Struct;
```

В интегрированном редакторе тестов САПР ISE разработать тест для полученного устройства и выполнить моделирование его работы в ISE Simulator. Описание устройства и результаты моделирования занести в отчет. Подключить устройство к текущему проекту, выбрав в меню Project пункт Add Source.

Задание 5. Разработать поведенческое VHDL описание схемы преобразования четырехразрядного информационного кода D[0..3] в код активизации 7-сегментного индикатора LED[0..7] в соответствии с таблицей 4. Подключить устройство к библиотеке компонентов текущего проекта. Текст описания занести в отчет.

Разработка и отладка основного модуля проекта

Основной модуль (модуль верхнего уровня, файл main_ex.vhd) объединяет описания модулей: подавления дребезга (файл lab2_ex.vhd - см. задание 2), модуль управления индикацией (файл sevenseg_mux.vhd – см. задание 4), модуль перекодирования тетрады в код семисегментного индикатора (sevenseg_decoder.vhd – см. задание 5). Модуль должен содержать основной 16-разрядный счетчик (MAIN_COUNTER), а также устройство деления частоты входного синхросигнала (50 MHz) на делитель 2**16 для получения частоты активации 7-сегментов (~800 Гц). Модуль управления индикацией использует сигнал переполнения этого счетчика (CLK_DIV) для циклического сдвига четырехразрядного регистра A_int. Таким образом, каждая цифра активируется каждый четвертый импульс CLK_DIV, в результате чего частота активации составляет ~200 Гц.

Задание 6. В редакторе схем САПР ISE добавить исходное описание, указав путь к файлу main_ex.vhd.

Код файла представлен ниже. Необходимо заменить пропущенные сигналы, отмеченные символом «...».

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY main IS
   PORT ( CLK   : IN    std_logic;
          COUNT : IN    std_logic;
          RESET : IN    std_logic;
          A     : OUT   std_logic_vector (3 DOWNTO 0);
          LED   : OUT   std_logic_vector (7 DOWNTO 0));
END main;

ARCHITECTURE Behavioral OF main IS

   COMPONENT Seven_Segment_Driver
      PORT (   CLK            : IN    std_logic;
               CLK_DIV        : IN    std_logic;
               RST            : IN    std_logic;
               Q              : IN    std_logic_vector (15 DOWNTO 0);
               D              : OUT   std_logic_vector (3 DOWNTO 0);
               A              : OUT   std_logic_vector (3 DOWNTO 0));
   END COMPONENT;

   COMPONENT led_decode
      PORT ( DH       : IN    std_logic_vector (3 DOWNTO 0);
             SEG_DATA : OUT   std_logic_vector (7 DOWNTO 0));
   END COMPONENT;

   COMPONENT lab2_example
      PORT ( RST   : IN    std_logic;
             CLK   : IN    std_logic;
             COUNT : IN    std_logic;
             CNT   : OUT   std_logic);
   END COMPONENT;

   SIGNAL CNT_int,CNT_ff,CNT_RISE:std_logic;
   SIGNAL COUNTER: integer;
   SIGNAL COUNTER_OVF: std_logic;

   -- Main counter
   SIGNAL MAIN_COUNTER: std_logic_vector(15 DOWNTO 0);
   ...
   ...
 BEGIN

   ssd_inst : Seven_Segment_Driver
      PORT MAP (CLK=>CLK,
                CLK_DIV=> ...,
                Q(15 DOWNTO 0)=>...,
                RST=>...,
                D(3 DOWNTO 0)=>...,
                A(3 DOWNTO 0)=>...
                );

   led_decode_inst : led_decode
      PORT MAP (DH(3 DOWNTO 0)=>...,
                SEG_DATA(7 DOWNTO 0)=>...
                );
```

```
   lab2_example_inst : lab2_example
      PORT MAP (CLK=>CLK,
                COUNT=>...,
                RST=>RESET,
                CNT=>CNT_int);

  -- Описание делителя частоты
  COUNTER_inst: PROCESS (CLK)
  BEGIN
    IF (CLK='1' and CLK'event) THEN
      IF (RESET='1' or COUNTER_OVF='1') THEN
        COUNTER <= ...;
      ELSE
        COUNTER <= COUNTER + 1;
      END IF;
    END IF;
  END PROCESS;
  COUNTER_OVF <= '1' WHEN COUNTER = 2**16 ELSE '0';

  --Детектор фронта сигнала CNT
  CNT_RISE <= '1' WHEN CNT_int='1' and ...='0' ELSE '0';
  CNT_ff_inst: PROCESS (CLK)
  BEGIN
    IF (CLK='1' and CLK'event) THEN
      IF (RESET='1') THEN
        CNT_ff <= '0';
      ELSE
        CNT_ff <= ...;
      END IF;
    END IF;
  END PROCESS;


  --Основной счетчик
  MAIN_COUNTER_inst: PROCESS (CLK)
  BEGIN
    IF (CLK='1' and CLK'event) THEN
      IF (RESET='1') THEN
        MAIN_COUNTER <= (others => '0');
      ELSIF (CNT_RISE = '1') THEN
        MAIN_COUNTER <= MAIN_COUNTER + 1;
      END IF;
    END IF;
  END PROCESS;


END BEHAVIORAL;
```

Созданное описание следует указать в качестве модуля верхнего уровня, для чего необходимо выбрать его в дереве описаний проекта во вкладке Sources окна Sources, после чего в меню Sources выбрать пункт Set as Top Module. Схему основного модуля занести в отчет.

Модификация файла ограничений проекта

Для реализации частей разработанного устройства и последующей их отладки необходимо создать файл ограничений, содержащий назначение контактов целевой

микросхемы. Вариант назначения сигналам контактов микросхемы, расположенной на плате набора XC3S200, представлены в таблице 5 и в файле main_xc3s200.ucf.

Таблица 5 – Вариант назначения контактов микросхемы сигналам схемы

Сигнал	Номер контакта для набора XC3S200	Номер контакта для набора Nexys2 500E XC3S500E	Номер контакта для набора Nexys2 1200E XC3S1200E	Назначение
CLK	T9	B8	B8	Глобальный сигнал GCLK0(50 МГц)
COUNT	M13	H13	H13	Сигнал от кнопки 3
RESET	L14	B18	B18	Сигнал от кнопки 0
LED[0]	N16	H14	H14	Сигнал управления сегментом G
LED[1]	F13	J17	J17	Сигнал управления сегментом F
LED[2]	R16	G14	G14	Сигнал управления сегментом E
LED[3]	P15	D16	D16	Сигнал управления сегментом D
LED[4]	N15	D17	D17	Сигнал управления сегментом C
LED[5]	G13	F18	F18	Сигнал управления сегментом B
LED[6]	E14	L18	L18	Сигнал управления сегментом A
LED[7]	P16	C17	C17	Сигнал управления сегментом DP
A[0]	D14	F17	F17	Сигнал управления анодом 0
A[1]	G14	H17	H17	Сигнал управления анодом 1
A[2]	F14	C18	C18	Сигнал управления анодом 2
A[3]	E13	F15	F15	Сигнал управления анодом 3

Задание 7. В программе Xilinx PACE создать файл ограничений *.ucf или добавьте в проект имеющийся main_xc3s200.ucf. В редакторе необходимо назначить внешние выводы сигналам разрабатываемого устройства в соответствии с таблицей 5. Для этого выбрать модуль верхнего уровня в дереве описаний проекта во вкладке Sources окна Sources, после чего во вкладке Processes окна Processes выбрать ветвь User Constraints и пункт Assign Package Pins. В окне редактора PACE назначение контактов выполняется в поле LOC.

Задание 8. В САПР ISE выполнить автоматический синтез технологической схемы, размещение и трассировку полученного устройства на кристалле Spartan3 XC3S200 ft256 (или для Spartan XC3S 500E/1200E, FG320), генерировать файл конфигурации ПЛИС (*.bin). Для этого в окне Sources выбрать вкладку Sources и в списке Sources For выбрать строку

Synthesis/Implementation. После этого в дереве описаний проекта выбрать описание верхнего уровня, а во вкладке Processes окна Processes выбрать ветвь Generate Programming File и пункт Programming File Generation Report.

Занести в отчет общие сведения о результатах проектирования устройства с вкладки Design Summary. Сделать выводы о быстродействии полученного устройства, используя отчет Static Timing Report.

Задание 8. Выполнить программирование макетной ПЛИС Spartan3 отладочного набора XC3S200 или Nexys2. Для этого в окне Sources выбрать вкладку Sources и в списке Sources For выбрать строку Synthesis/Implementation. После этого в дереве описаний проекта выбрать описание верхнего уровня, а во вкладке Processes окна Processes выбрать ветвь Generate Programming File и пункт Configure Device. В результате будет запущен модуль iMPACT. В открывшемся диалоге выбора способа программирования отметить пункт Configure devices using Boundary-Scan и выбрать автоматический способ идентификации. В результате будет определена цепочка, состоящая из ПЛИС и Flash ПЗУ.

После автоматического определения цепочки граничного сканирования по JTAG интерфейсу необходимо запрограммировать ПЛИС XC3S200 (или Spartan XC3S500E/1200E FG320). Для этого необходимо указать файл конфигурации ПЛИС (*.bit), полученный в задании 7, после чего в окне iMPACT Processes меню выбрать пункт Program.

Провести тестирование разработанного устройства. Результаты тестирование представить в отчете.

Контрольные вопросы

1. Назовите основные этапы проектирования цифровых устройств на основе ПЛИС.
2. Какой тип автомата (Мили или Мура) реализован в устройстве подавления дребезга.
3. Перечислите устройства, входящие в состав отладочного набора XC3S200 или Nexys2.
4. Какую информацию содержит файл ограничений *.ucf.
5. Какой стиль описания на VHDL использован в примере описания драйвера 7-сегментных индикаторов.

Требования к отчету

Отчет должен содержать:

6. ФИО студента, номер группы, номер варианта, номер и название лабораторной работы.

7. Функциональную схему разрабатываемого устройства
8. Диаграмма состояния автомата подавления дребезга
9. Заполненные в соответствии с индивидуальным заданием таблицы 2 и 3.
10. Карты Карно и результаты минимизации функций сигналов CNT, DLY_EN, SN(0),SN(1).
11. VHDL код модуля подавления дребезга.
12. Результаты верификации схемы в ПО Modelsim.
13. VHDL код модуля управления 7-сегментным индикатором и декодера 7-сегментного кода.
14. VHDL код модуля верхнего уровня проекта.
15. Результаты верификации проекта на отладочной плате в виде протокола испытаний: таблицы со столбцами «Номер эксперимента», «Ожидаемый результат», «Полученный результат».
16. Выводы по проделанной работе.

Список литературы

1. Попов А.Ю. Проектирование цифровых устройств с использованием ПЛИС: Учеб. пособие. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2009.
2. Угрюмов Е. П. Цифровая схемотехника: Учеб. Пособие для вузов. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 800 с.: ил.
3. Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах с программируемой структурой, БХВ-Петербург, 2006, 708 с.
4. Spartan-3 FPGA Family: Complete Data Sheet. Xilinx Inc.
5. Xilinx ISE Guide (HTML Book). Xilinx Inc.
6. Xilinx ISE 9 Software Manuals
7. Spartan-3 Starter Kit Board User Guide (www.digilentinc.com)
8. Nexys 2 Board User Guide (www.digilentinc.com)
9. В. Зотов Инструментальный комплект Spartan3 Starter Kit