

Министерство науки и высшего образования Российской Федерации

Томский государственный университет  
систем управления и радиоэлектроники

А.А. Бомбизов

## **ЧАСЫ**

Методические указания к выполнению  
лабораторной и самостоятельной работы  
по дисциплине «Проектирование систем на кристалле»

Томск  
2020

УДК 681.3 (075.32)

ББК 32.973стд1-02

**Б 803**

**Рецензент:**

**Тренкаль Е.И.**, доцент кафедры конструирования узлов и деталей радиоэлектронной аппаратуры ТУСУР, канд. техн. наук

**Бомбизов, Александр Александрович**

Б 803 Знакомство со средой. Базовая логика: методические указания к выполнению лабораторной и самостоятельной работы по дисциплине «Проектирование систем на кристалле» / А.А. Бомбизов. – Томск. гос. ун-т систем упр. и радиоэлектроники, 2020. – 15 с.

Настоящее методическое указание по выполнению лабораторной и самостоятельной работы по дисциплине «Проектирование систем на кристалле».

Методическое пособие содержит краткое описание порядка визуального проектирования RTL-описания ПЛИС.

Одобрено на заседании каф. КУДР, протокол № 234 от 5 марта 2022 г.

УДК 681.3 (075.32)

ББК 32.973стд1-02

© Бомбизов А.А., 2020

© Томск. гос. ун-т систем упр. и радиоэлектроники, 2020

## 1 Введение

Разработка решений для FPGA не ограничивается созданием HDL-описания в текстовом редакторе. Существуют альтернативные и, возможно, более удобные инструменты для построения сложных систем.

Целью настоящей работы является освоение визуального проектирования системы в среде Vivado 2019.

## 2 Краткая теория

Для облегчения разработок каждый поставщик ПЛИС предлагает собственный набор аппаратных и программных блоков интеллектуальной собственности (IP – intellectual property). Аппаратные блоки интеллектуальной собственности (аппаратные IP) представляют собой уже реализованные блоки, такие как микропроцессорные ядра, гигабитные интерфейсы, умножители, сумматоры, функции умножения с накоплением и им подобные. Эти блоки разрабатываются так, чтобы они были максимально эффективны и с точки зрения потребляемой мощности, и с точки зрения производительности, и с точки зрения площади, занимаемой на кристалле. Для каждого семейства ПЛИС характерны различные комбинации и различное количество таких блоков. Программные IP-блоки представляют собой библиотеки исходных кодов высокоуровневых функций, которые могут использовать разработчики конечных устройств. Эти функции выражаются языком описания аппаратных средств HDL, таким как Verilog или VHDL на уровне регистровых передач.

В течение последних лет производители ПЛИС уделяют много внимания повышению эффективности инструментов проектирования при работе с FPGA большой логической емкости. В САПР Vivado (предыдущая САПР от Xilinx именовалась ISE Design Suite) компания Xilinx отказалась от графического ввода схемы. Взамен этого предложен инструмент под названием IP Integrator, который призван обеспечить графическое представление проекта, но уже не для низкоуровневых компонентов, а для IP-ядер (как и следует из его названия). Актуальность такого способа ввода становится понятной, если учесть эффективность повторного использования кода (англ. термин code reuse), что при программировании выступает в виде использования готовых библиотек, а при разработке проектов на базе ПЛИС – в виде использования готовых функциональных компонентов – IP-ядер.

Инструмент IP Integrator был добавлен в САПР Vivado 2013.3. С ним связано понятие блочного проекта (block design), который может выступать

в качестве элемента иерархии, наряду с модулями на языках VHDL и Verilog. На рисунке 1 показано, что на панели Flow Navigator есть группа, имеющая название IP Integrator. При выборе в этой группе пункта Create Block Design начинается работа «мастера» создания новой подсистемы на основе IP-ядер. Помимо IP-ядер в блочном проекте допускается использование созданных ранее модулей.



Рисунок 1 – Группа IP-интегратора

Для освоения работы с IP-интегратором выполним разработку тестового проекта, схема которого изображена на рисунке 2. Схема состоит из двух примитивов И, ИЛИ и разработанного во второй лабораторной работе модуля дешифратора.

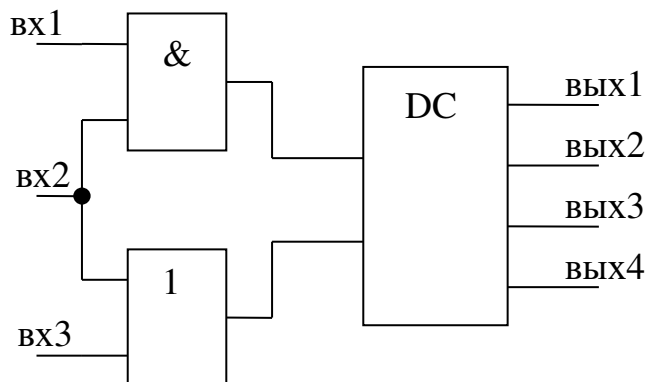


Рисунок 2 – Тестовый проект

Для запуска IP Integrator нужно нажать кнопку для создания нового блочного проекта Create Block Design, которая находится в панели Flow Navigator – группа IP Integrator, пункт Create Block Design (рисунок 1). В появившемся диалоге (рисунок 3) необходимо ввести имя, для тестового проекта – testBlockDesign, и нажать на кнопку ОК.

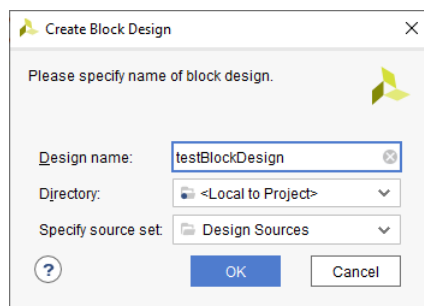


Рисунок 3 – Диалог ввода названия блочного проекта

По завершению работы мастера будет отображена рабочая область (рисунок 4).

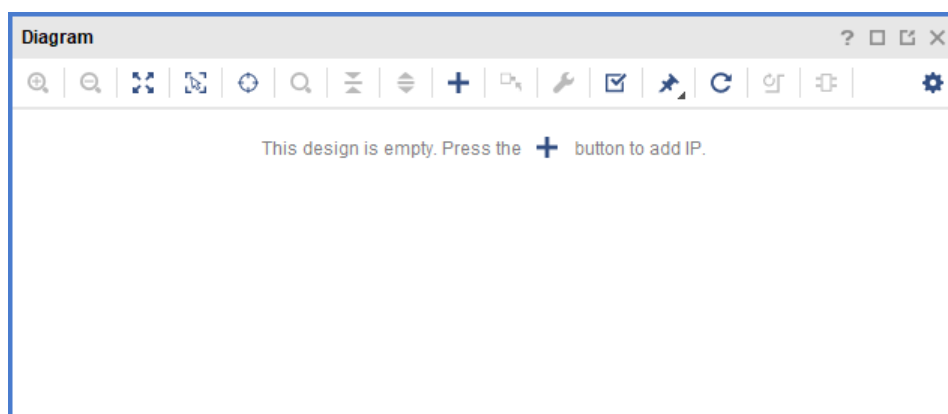


Рисунок 4 – Рабочая область блочного проекта

Для добавления на рабочую область ранее созданного модуля необходимо нажать на свободную область правой кнопкой мыши и выбрать пункт Add Module (рисунок 5).

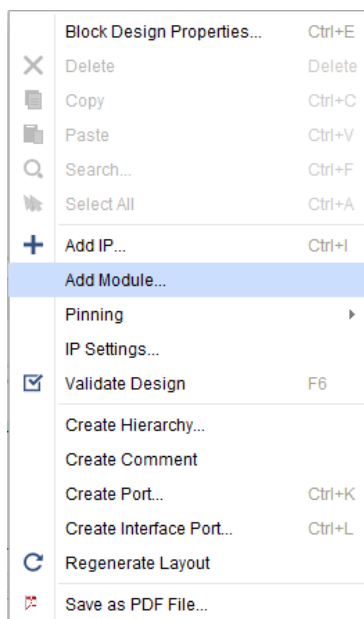


Рисунок 5 – Контекстное меню для добавления модуля

В появившемся диалоге необходимо выбрать требуемый модуль (Decoder – дешифратор) как показано на рисунке 6.

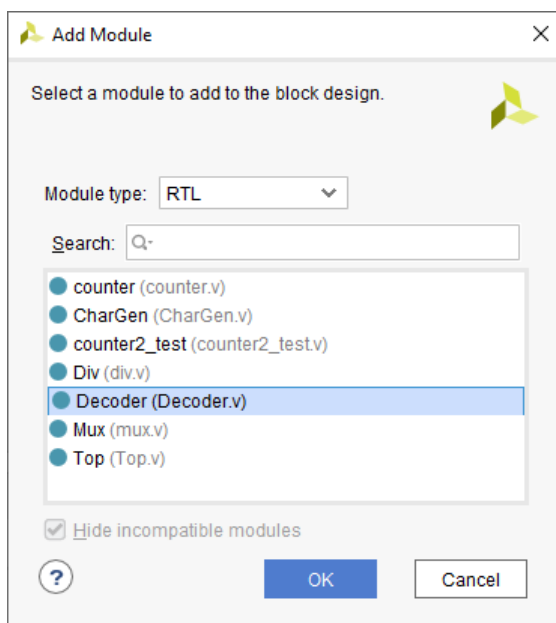


Рисунок 6 – Диалог выбора модуля, включенного в текущий проект

После выбора модуля на рабочей панели появится его блок, как показано на рисунке 7.

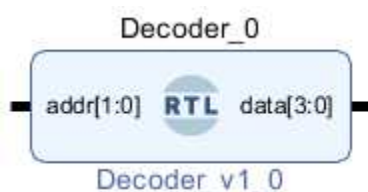


Рисунок 7 – Модуль на рабочей панели

Нужно отметить, что модуль проекта изображен с надписью RTL, все входы изображены слева, а выходы справа.

Далее необходимо добавить IP-блоки для элементов И и ИЛИ путем нажатия на кнопку «+» как показано на рисунке 8.



Рисунок 8 – Добавление IP-блоков

В результате появится окно со всем разнообразием IP-блоков, представленных в текущей версии среды разработки, как показано на рисунке 9.

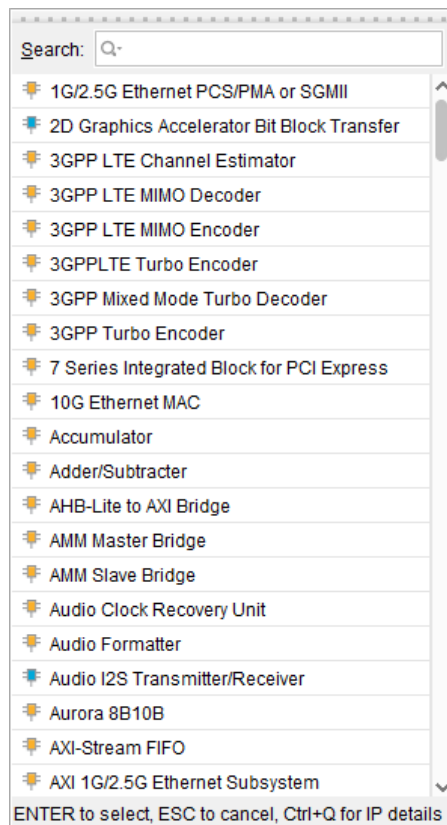


Рисунок 9 – Окно выбора IP-блоков

IP-блоками логических элементов являются Utility Reduced Logic и Utility Vector Logic (рисунок 10). Первый может использоваться в качестве одного из трех элементов И, ИЛИ, исключающее ИЛИ. Второй – И, ИЛИ, исключающее ИЛИ, инвертор. У первого шинный вход и один выход, у второго два шинных входа и шинный выход.

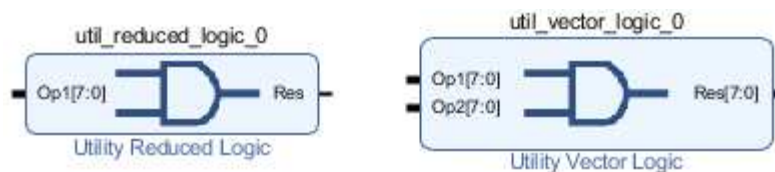


Рисунок 10 – IP-блоки логических элементов

Для тестового проекта будет использоваться вариант Utility Reduced Logic. На рабочую область необходимо разместить два логических блока. Затем путем клика правой кнопкой мыши по каждому нужно выбрать пункт «Customize Block...» и в открывшемся диалоге установить тип логического элемента и разрядность (2) входной шины.

Далее необходимо объединить входы (см. рисунок 2) в шине. Для этого должен быть использован IP-блок Concat (для разъединения шины служит IP-блок Slice). Добавьте на рабочую область три таких IP-блока и

сконфигурируйте их как показано на рисунке 11, то есть объедините две шины с одной цепью в одну общую шину с двумя цепями.

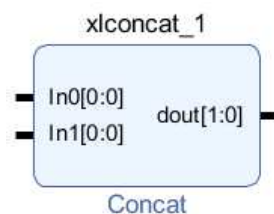



Рисунок 11 – IP-блоки логических элементов

Соедините все блоки в соответствии с рисунком 2. Для оптимизации размещения нажмите на кнопку Regenerate Layout .

В созданном блочном проекте определите внешние порты путем нажатия правой кнопкой мыши по порту IP-блока и выбора пункта Make External. Нужно учесть, что один из внешних входов будет для двух модулей общим.

После нажатия на кнопку  должна получиться схема, как показано на рисунке 12.

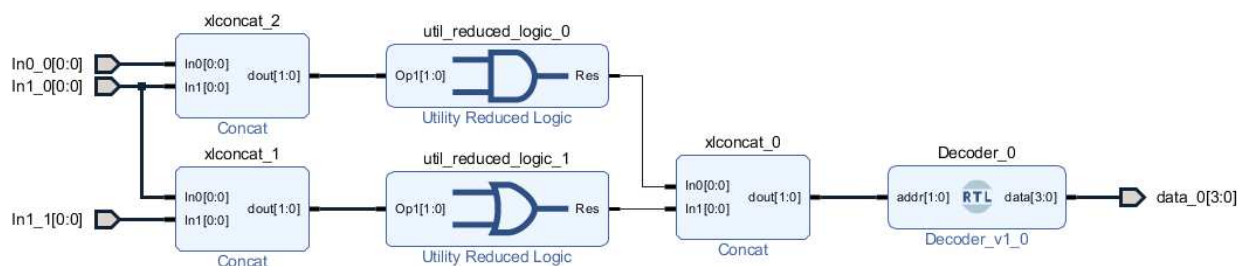


Рисунок 12 – Блочный проект логических элементов

Красивая картинка это еще не конец. Далее следует преобразовать блочный проект в модуль HDL-описания. Для этого необходимо переключиться во вкладку Sources и в контекстном меню блочного проекта выбрать пункт Create HDL Wrapper... как показано на рисунке 13.

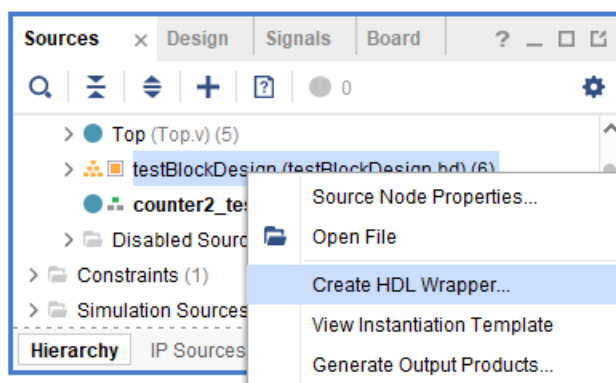


Рисунок 13 – Создание HDL-описания блочного проекта



В появившемся диалоге необходимо выбрать пункт «Let Vivado manage wrapper and auto-update» (автоматическое обновление при изменении дочерних модулей и связей) и нажать на кнопку ОК. В результате будет сгенерирован код модуля со структурным HDL-описанием разработанной ранее схемы.

```

module testBlockDesign
  (In0_0,
   In1_0,
   In1_1,
   data_0);
input [0:0]In0_0;
input [0:0]In1_0;
input [0:0]In1_1;
output [3:0]data_0;

wire [3:0]Decoder_0_data;
wire [0:0]In0_0_1;
wire [0:0]In1_0_1;
wire [0:0]In1_1_1;
wire util_reduced_logic_0_Res;
wire util_reduced_logic_1_Res;
wire [1:0]xlconcat_0_dout;
wire [1:0]xlconcat_1_dout;
wire [1:0]xlconcat_2_dout;

assign In0_0_1 = In0_0[0];
assign In1_0_1 = In1_0[0];
assign In1_1_1 = In1_1[0];
assign data_0[3:0] = Decoder_0_data;
testBlockDesign_Decoder_0_0 Decoder_0
  (.addr(xlconcat_0_dout),
   .data(Decoder_0_data));
testBlockDesign_util_reduced_logic_0_1 util_reduced_logic_0
  (.Op1(xlconcat_2_dout),
   .Res(util_reduced_logic_0_Res));
testBlockDesign_util_reduced_logic_0_2 util_reduced_logic_1
  (.Op1(xlconcat_1_dout),
   .Res(util_reduced_logic_1_Res));
testBlockDesign_xlconcat_0_0 xlconcat_0
  (.In0(util_reduced_logic_0_Res),
   .In1(util_reduced_logic_1_Res),
   .dout(xlconcat_0_dout));
testBlockDesign_xlconcat_0_1 xlconcat_1

```

```

(.In0(In1_0_1),
 .In1(In1_1_1),
 .dout(xlconcat_1_dout));
testBlockDesign_xlconcat_0_2 xlconcat_2
(.In0(In0_0_1),
 .In1(In1_0_1),
 .dout(xlconcat_2_dout));
endmodule

```

Перед последующей компиляцией необходимо подготовленный модуль сделать модулем верхнего уровня. Если потребуется вернуться к структурной схеме блочного проекта, то для этого необходимо нажать Open block design (см. рисунок 1).

В практической части работы необходимо разработать часы реального времени. Для упрощения должен быть реализован алгоритм работы с минутами и секундами (при необходимости переделать на часы и минуты, думаю, не составит труда). На рисунке 14 изображен фрагмент схемы электрической структурной будущего модуля.

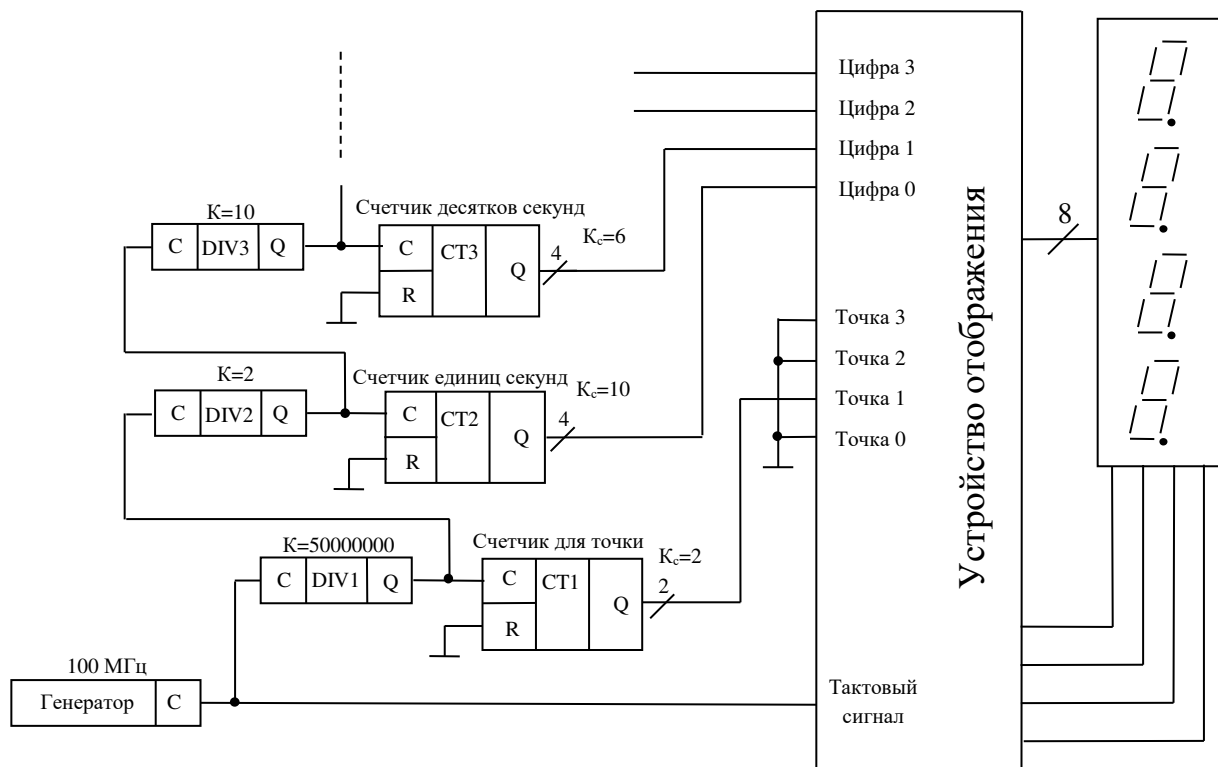


Рисунок 14 – Структурная схема устройства отображения с динамической индикацией

В схеме источником сигнала является тактовый генератор с частотой 100 МГц. Далее формируется частота для мигания точкой с равная 1 Гц, которая достигается путем деление исходной в 50000000 раз (2 Гц) и выравнивания скважности к значению два с использованием счетчика с

коэффициентом счета  $K_c=2$ . Работа часов разделяется на работу счетчиков единиц секунд (0–9), десятков секунд (0–5), единиц минут (0–9), десятков минут (0–5). Для каждого счетчика должна быть сформирована своя тактовая частота: для единиц секунд – 1 Гц (достигается путем деления частоты 100 МГц на 50000000 и на 2), для десятков секунд – 0,1 Гц (достигается путем деления частоты тактирования единиц секунд на десять) и т.д.

### **3 Порядок выполнения работы**

В ходе данной работы необходимо создать, верифицировать на симуляторе и испытать на отладочной плате модуль часов.

Для этого потребуется выполнить следующие действия:

3.1 Изучите предложенный в п. 2 теоретический материал.

3.2 Для выполнения текущей работы используйте проект, созданный в третьей лабораторной работе.

3.3 Для освоения порядка работы с блочным проектированием реализуйте тестовый модуль, описанный в теоретической части.

Реализация модуля часов с использованием блочного дизайна.

3.4 Создайте новую подсистему на основе IP-ядер. Название целесообразно указать ClockSystem (непринципиально).

3.5 Добавьте модуль устройства отображения, созданный в третьей лабораторной работе. Предусправляя этап симуляции, стоит сократить делитель частоты, встроенный в модуль до двух, иначе придется ожидать большое количество тактов до появления изменений на временной диаграмме. Если вы поменяли что-нибудь во встроенном модуле, то после этого необходимо открыть блочный проект и нажать на Refresh Changes Modules по аналогии с Reload для обновления RTL-схемы.

3.6 Добавьте модули делителя DIV1 и счетчика CT1 ( $K_c=2$ ). Для этапа симуляции установите значение делителя частоты равным 10.

3.7 Выполните соединение выхода делителя частоты со входом счетчика.

3.8 Выполните соединение выхода счетчика со входом «точка 1» устройства отображения. Если в устройстве отображения для подключения сигналов точке используется шинный вход, то используйте для преобразования цепей IP-блок Concat.

3.9 Выполните подключение остальных неиспользуемых входов для точек к логическому нулю, используя IP-блок Constant.

3.10 Создайте внешние порты для тактового сигнала, для анодов и катодов. Тактовый сигнал должен поступать на делитель DIV1 и на устройство отображения.

3.11 Преобразуйте проект в модуль HDL-описания. В результате должен получиться:

```
module ClockSystem_wrapper(anode_0,cathod_0,iclk_0);
```

где `anode_0`, `cathod_0` – выходы для подключения к счетверенному семисегментному индикатору; `iclk_0` – вход тактового сигнала.

3.12 Создайте тестовый стенд для верификации работы и выполните симуляцию работы модуля `ClockSystem_wrapper`. В результате симуляции должна получиться временная диаграмма, как показано на рисунке 15.



Рисунок 15 – Временная диаграмма симуляции работы точки в часах

В данную диаграмму добавлен сигнал с выхода счетчика СТ1. В результате, когда счетчик выдает логическую единицу, то при активации анода для цифры 1 выполняется включение точки в седьмом бите катодной шины.

3.13 Добавьте в блочный проект делитель DIV2 с коэффициентом деления 2 и счетчик СТ2 с коэффициентом счета 10 как показано на рисунке 14.

3.14 Подсоедините вход делителя DIV2 к выходу делителя DIV1.

3.15 Подсоедините выход делителя DIV2 ко входу счетчика СТ2.

3.16 Подсоедините выход счетчика СТ2 ко входу «цифра 0» устройства отображения.

3.17 Выполните симуляцию работы модуля ClockSystem\_wrapper. В результате должна получиться диаграмма, как показано на рисунке 16.

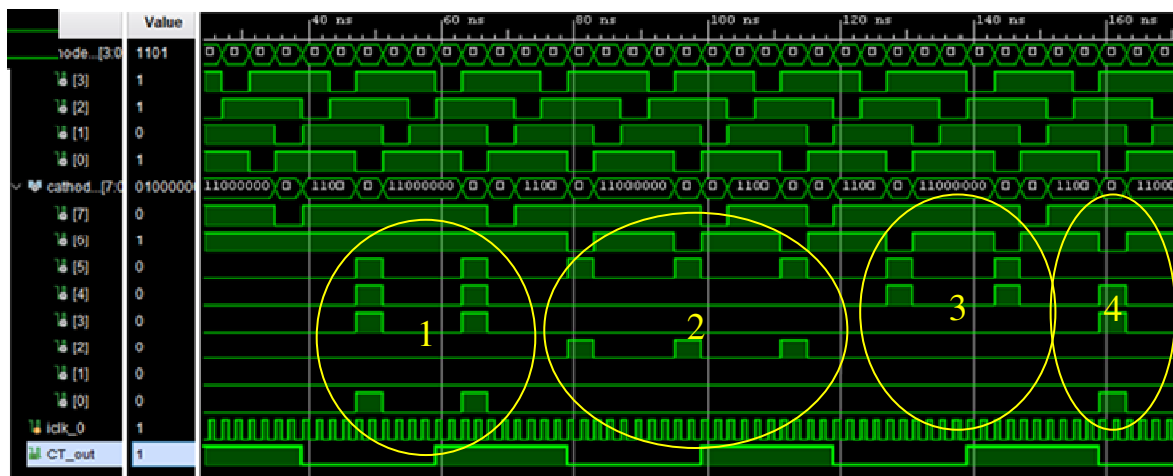


Рисунок 16 – Временная диаграмма симуляции единиц секунд  
(начало отсчета 40 нс)

3.18 Добавьте и соедините модули для отображения десятков секунд согласно рисунку 14. Выполните симуляцию и верификацию работы модуля часов. В результате должна получиться диаграмма, как изображено на рисунке 17.

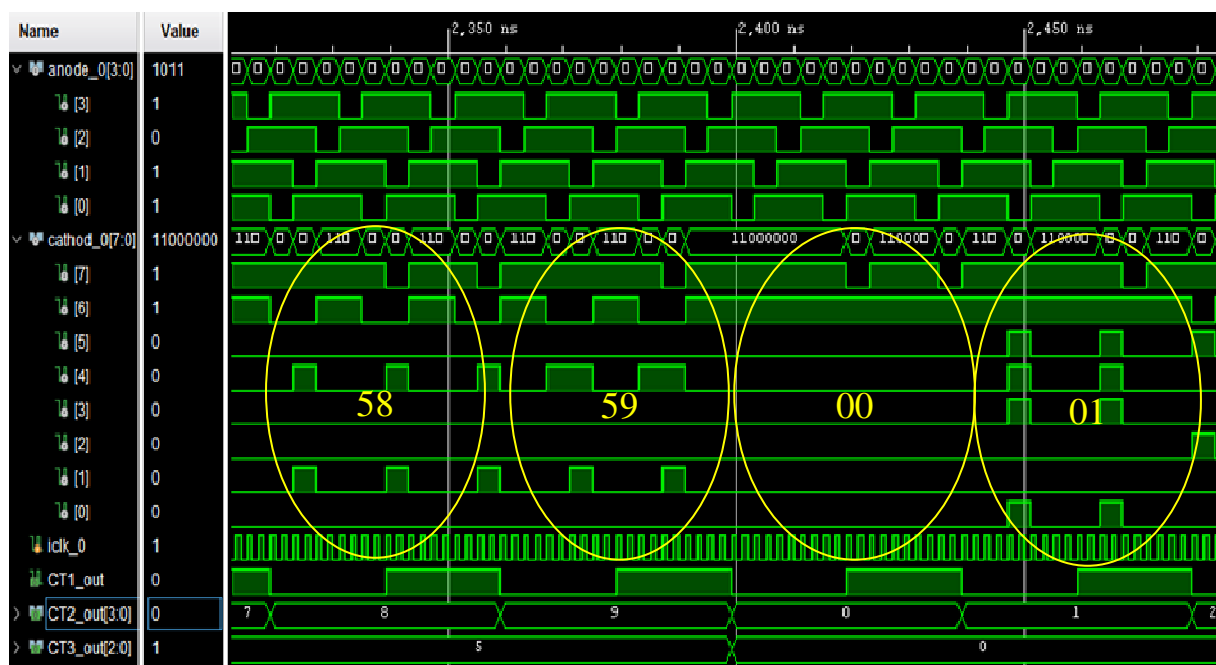


Рисунок 17 – Временная диаграмма симуляции секунд  
(начало отсчета 2300 нс)

3.19 Добавьте и соедините модули для отображения единиц и десятков минут.

3.20 Выполните симуляцию и верификацию.

3.21 Добавьте в схему механизмы перевода (в большую сторону) секунд и минут. В блочном дизайне у вас должно быть добавлено два внешних порта.

3.22 Предусмотрите в тестовом стенде эмуляцию сигналов (см. аналогию с reset во второй лабораторной работе).

3.23 Выполните симуляцию и верификацию.

3.24 Установите коэффициент деления в делителе модуля отображения  $1 \cdot 10^5$ – $5 \cdot 10^5$ .

3.25 Установите коэффициент деления делителя DIV1 в  $50 \cdot 10^6$ .

3.26 Выполните синтез (не забудьте Refresh Changes Modules), настройку портов (для перевода используйте кнопки BTNL – для секунд и BTNR – для минут), размещение и трассировку и генерацию прошивки.

#### **4 Контрольные вопросы**

4.1 Что такое IP-блок?

4.2 Принцип построения часов на ПЛИС?

4.3 Порядок создания блочного проекта?

4.4 Разница между аппаратными и программными IP-блоками?

4.5 Каким образом осуществляется перевод секунд и минут?

#### **Список литературы**

1 Basys3™ FPGA Board Reference Manual. URL: [https://reference.digilentinc.com/\\_media/reference/programmable-logic/basys-3/basys3\\_rm.pdf](https://reference.digilentinc.com/_media/reference/programmable-logic/basys-3/basys3_rm.pdf) (дата обращения: 31.01.2020);

2 Basys3™ Schematic. URL: [https://reference.digilentinc.com/\\_media/reference/programmable-logic/basys-3/basys-3\\_sch.pdf](https://reference.digilentinc.com/_media/reference/programmable-logic/basys-3/basys-3_sch.pdf) (дата обращения: 31.01.2020).