Лабораторна робота № *2*

|  |  |
| --- | --- |
|  | “Разработка общей структуры пользовательского компонента для системы класса «system-on-chip»” |
|  | (назва лабораторної роботи) |
| з дисципліни | Компонентно-орієнтоване проектування |
|  | (шифр)  ХАІ**.**503**.**535**.**20О**.**123-Комп'ютерна інженерія**,** 1805023, 1805025 |

|  |  |  |
| --- | --- | --- |
| Виконали студенти гр. | *535* | Єлюхін Р. В.  Резніков А.О. |
|  | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | К.т.н., доцент каф. 503 | |
|  |  | *Перепелицин А. Є.* |
| (підпис, дата) |  | (П.І.Б.) |

**Цель работы** – ознакомиться с принципами построения пользовательских компонентов для систем класса «System-on-Chip», разработать общую структуру пользовательского компонента.

**Постановка задачи**

Согласно заданию разработать общую структуру пользовательского компонента. Обосновать наличие каждого из модулей в составе компонента, дать их короткую характеристику и назначение. Обозначить связи между модулями компонента и тип передаваемой информации (управляющие сигналы, сигналы тактирования, данные и т.д.). На структурной схеме отобразить входные и выходные порты – описать их назначение и общий формат передаваемых данных (в том числе направление передачи).

Система опроса клавиатуры с возможностью индикации нажатых символов

1. Общее описание системы компонента, его назначение и предполагаемая (предварительная) характеристика

Данная система представляет собой систему считывания и индикации, при помощи семисегментных индикаторов, нажатых клавиш клавиатуры. Каждый символ клавиатуры, или комбинацию символов, можно представить в виде 16-ричных символов, для индикации этих 16-ричных символов мы используем семисегментные индикаторы.

1. Общая структура системы компонента (схема /рисунок) со всеми необходимыми обозначениями (характер связей и т.д.).



1. Описание модулей системы, портов ввода /вывода и др
   1. Преобразователь частоты – этот компонент представляет собой полный контроль над частотой.
   2. Контроллер интерфейса PS/2 – этот компонент служит для обработки и передачи нажатых клавиш клавиатуры. Данный блок определяет нажатий символ, или комбинацию символов и на выход передает массив битов, который представляет собой нажатый символ, или комбинацию символов в виде 16-ричных чисел.
   3. Основной блок – блок управления входящими сигналами.
   4. Контроллер семисегментного индикатора – блок индикации нажатого символа. Данный компонент служит для преобразования и индикации 16-ричных чисел на соответствующих им семисегментных индикаторах(для каждого 16-ричного числа свой индикатор).
   5. Контроллер кнопки – блок управления кнопкой. Осуществляет борьбу с дребезгом контактов.

**VHDL код описания сущности (ENTITY) каждого модуля:**

**button\_controller**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity button\_controller is**

**port**

**(**

**clk : in std\_logic;**

**button\_in : in std\_logic;**

**button\_out : out std\_logic**

**);**

**end button\_controller;**

**architecture rtl of button\_controller is**

**begin**

**end rtl;**

**seven\_segment\_indicator**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity seven\_segment\_indicator is**

**port**

**(**

**clk : in std\_logic;**

**indicator\_in : in std\_logic\_vector(0 to 7);**

**data\_out : out std\_logic;**

**sck : out std\_logic**

**);**

**end seven\_segment\_indicator;**

**architecture rtl of seven\_segment\_indicator is**

**begin**

**end rtl;**

**main\_block**

**-- Quartus II VHDL Template**

**-- Simple Dual-Port RAM with different read/write addresses and**

**-- different read/write clock**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity main\_block is**

**generic**

**(**

**DATA\_WIDTH : natural := 8;**

**ADDR\_WIDTH : natural := 6**

**);**

**port**

**(**

**clk : in std\_logic;**

**button\_in : in std\_logic;**

**ps2\_code\_in : in std\_logic\_vector(0 to 7);**

**ps2\_code\_new\_in : in std\_logic;**

**indicator\_out : out std\_logic\_vector(0 to 7);**

**nios : in std\_logic**

**);**

**end main\_block;**

**architecture rtl of main\_block is**

**begin**

**end rtl;**

**PS2\_controller**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity PS2\_controller is**

**port**

**(**

**clk : in std\_logic;**

**ps2\_clk : in std\_logic;**

**ps2\_data : in std\_logic;**

**ps2\_code\_new : out std\_logic;**

**ps2\_code : out std\_logic\_vector(0 to 7)**

**);**

**end PS2\_controller;**

**architecture rtl of PS2\_controller is**

**begin**

**end rtl;**

**top\_level\_model**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity top\_level\_model is**

**generic**

**(**

**DATA\_WIDTH : natural := 8;**

**ADDR\_WIDTH : natural := 6**

**);**

**port**

**(**

**clk : in std\_logic;**

**button\_in : in std\_logic;**

**indicator\_out\_display : out std\_logic;**

**clk\_display : out std\_logic;**

**ps2\_clk : in std\_logic;**

**ps2\_data : in std\_logic**

**);**

**end top\_level\_model;**

**architecture rtl of top\_level\_model is**

**component main\_block is**

**generic**

**(**

**DATA\_WIDTH : natural := 8;**

**ADDR\_WIDTH : natural := 6**

**);**

**port**

**(**

**clk : in std\_logic;**

**button\_in : in std\_logic;**

**ps2\_code\_in : in std\_logic\_vector(0 to 7);**

**ps2\_code\_new\_in : in std\_logic;**

**indicator\_out : out std\_logic\_vector(0 to 7);**

**nios : in std\_logic**

**);**

**end component main\_block;**

**component button\_controller is**

**port**

**(**

**clk : in std\_logic;**

**button\_in : in std\_logic;**

**button\_out : out std\_logic**

**);**

**end component button\_controller;**

**component frequency\_controller is**

**port**

**(**

**clk : in std\_logic;**

**clk\_button : out std\_logic;**

**clk\_main : out std\_logic;**

**clk\_ps2 : out std\_logic;**

**clk\_indicator : out std\_logic**

**);**

**end component frequency\_controller;**

**component PS2\_controller is**

**port**

**(**

**clk : in std\_logic;**

**ps2\_clk : in std\_logic;**

**ps2\_data : in std\_logic;**

**ps2\_code\_new : out std\_logic;**

**ps2\_code : out std\_logic\_vector(0 to 7)**

**);**

**end component PS2\_controller;**

**component seven\_segment\_indicator is**

**port**

**(**

**clk : in std\_logic;**

**indicator\_in : in std\_logic\_vector(0 to 7);**

**data\_out : out std\_logic;**

**sck : out std\_logic**

**);**

**end component seven\_segment\_indicator;**

**--Скопировать entity и переименовать в component для всех блоков**

**--Создаем сигналы для контроллеров**

**signal clk\_button : std\_logic;**

**signal button\_out : std\_logic;**

**signal clk\_main : std\_logic;**

**signal clk\_ps2 : std\_logic;**

**signal clk\_indicator : std\_logic;**

**signal ps2\_code : std\_logic\_vector(0 to 7);**

**signal ps2\_code\_new : std\_logic;**

**signal nios : std\_logic;**

**signal indicator\_out\_main : std\_logic\_vector(0 to 7);**

**begin**

**l\_bc : button\_controller port map(**

**clk => clk\_button,**

**button\_in => button\_in,**

**button\_out => button\_out**

**); --оператор создания компонентов, добавлять port map и/или generic map**

**l\_fc : frequency\_controller port map(**

**clk => clk,**

**clk\_button => clk\_button,**

**clk\_main => clk\_main,**

**clk\_ps2 => clk\_ps2,**

**clk\_indicator => clk\_indicator**

**);**

**l\_ps2 : PS2\_controller port map(**

**clk => clk\_ps2,**

**ps2\_clk => ps2\_clk,**

**ps2\_data => ps2\_data,**

**ps2\_code\_new => ps2\_code\_new,**

**ps2\_code => ps2\_code**

**);**

**l\_main : main\_block port map(**

**clk => clk\_main,**

**button\_in => button\_out,**

**ps2\_code\_in => ps2\_code,**

**ps2\_code\_new\_in => ps2\_code\_new,**

**nios => nios,**

**indicator\_out => indicator\_out\_main**

**);**

**l\_seven\_segm : seven\_segment\_indicator port map(**

**clk => clk\_indicator,**

**indicator\_in => indicator\_out\_main,**

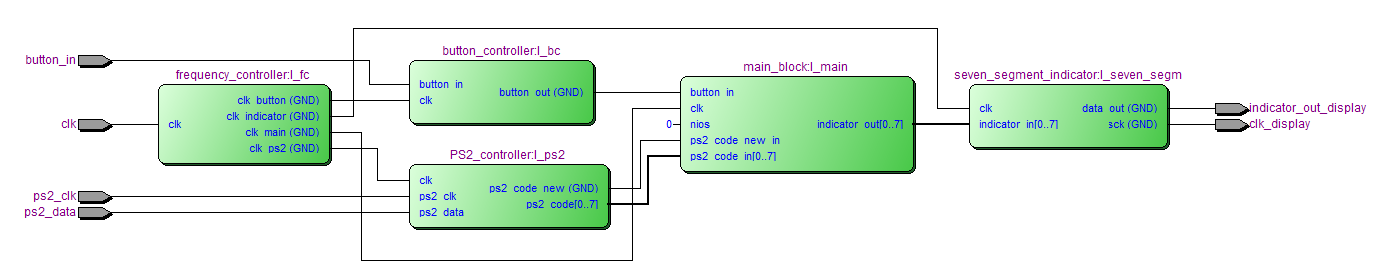
**data\_out => indicator\_out\_display,**

**sck => clk\_display**

**);**

**end rtl;**

**RTL представление проекта:**



**Выводы**

На этой лабораторной работе мыразработали общую структуру нашего компонента. Мы написали код описания сущности каждого модуля, далее мы связали все модули между собой и построили RTL представление проекта.