ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»»

Московский институт электроники и математики им. А.Н.Тихонова

**Курс «Системное проектирование цифровых устройств»**

**ОТЧЕТ**

**по практической работе №5**

тема работы: «Обработка звука на ПЛИС»

Выполнили:

Сиротинский Н.В.,

Якубов В.Ю.

Группа: БИВ185

Вариант 2

Принял:

Американов А.А.

Москва 2021

1. Задание
2. В соответствии с вариантом выбрать плату (DE1-SoC)
3. Создать проект, в котором сигнал с линейного входа от микрофона (или плеера) подается на линейный выход (наушники, динамики). Добавить управление от кнопки, убирающее один из каналов или меняющее каналы местами. Разработать генератор шума и добавить его к выходному звуку.
4. Реализовать индикацию на светодиодной ленте и 7-сегментном индикаторе. Реализовать регулировку громкости с использованием переключателей.
5. Выполнение работы
   1. Основная часть

Создаем проект в Quartus и добавляем в него все необходимые файлы из дополнительных материалов. В файле part1.v находится модуль верхнего уровня, там необходимо отредактировать 4 строчки так, чтобы звук с линейного входа передавался на линейный выход. Значения считываемых каналов необходимо без изменений передавать на запись. Сигналы готовности чтения и записи передаем в сигналы разрешения. В итоге получается следующее

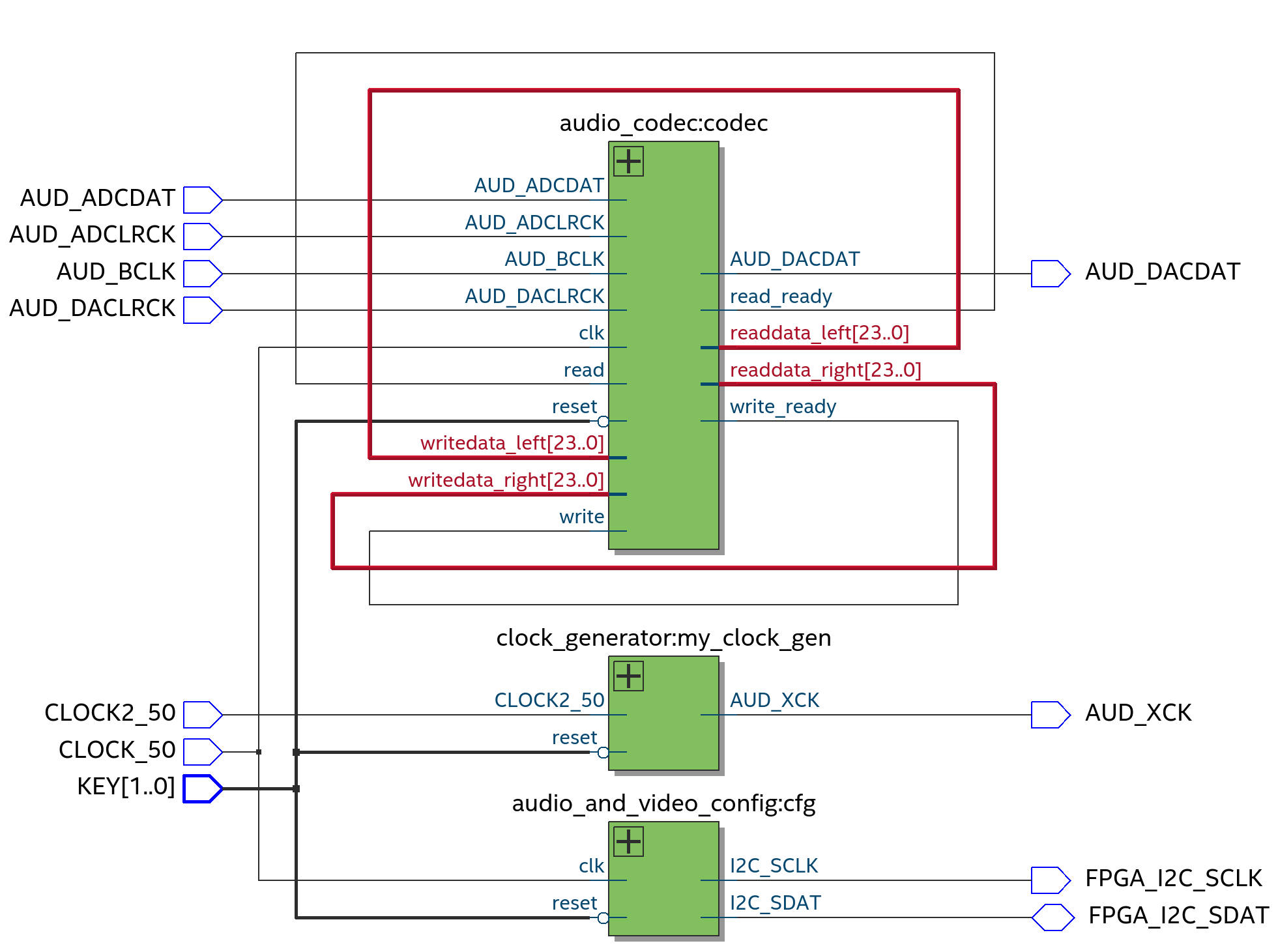
assign writedata\_left = readdata\_left;

assign writedata\_right = writedata\_right;

assign read = read\_ready;

assign write = write\_ready;

Скомпилируем проект и проведем моделирование. На рисунке 1 приведено RTL-представление модуля.

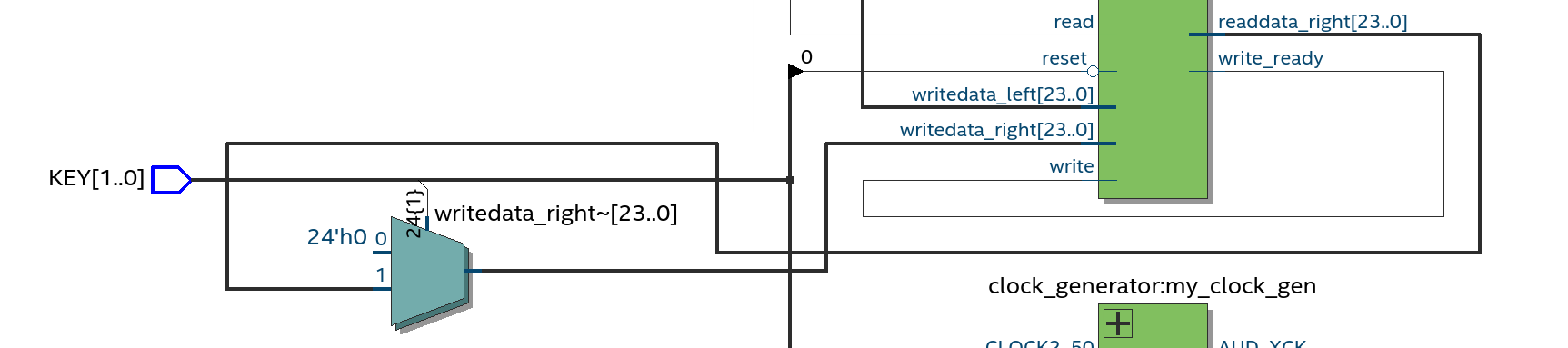
Рисунок 1. RTL-представление модуля верхнего уровня

Видно, что данные, получаемые из кодека, подаются ему же на вход.

Теперь модифицируем код, добавив управление каналами кнопкой. Пусть по нажатию кнопки выключается правый канал. В коде это будет выглядеть следующим образом

assign writedata\_right = KEY[1] ? readdata\_right : 24'b0;

На рисунке 2 изображено изменение в RTL-представлении.

Рисунок 2. Управление кнопкой в RTL-представлении

Видно, что на вход кодеку приходят данные с мультиплексора, который по значению с кнопки выбирает либо 0, либо выходные данные с кодека.

Теперь с помощью примеров реализуем генератор шума. Код модуля генератора шума приведен в листинге 1. Подключение шума выполним следующим образом:

noise\_generator noise\_gen(

.clk (CLOCK2\_50),

.rst\_n (reset),

.enable (SW[9]),

.Q(noise)

);

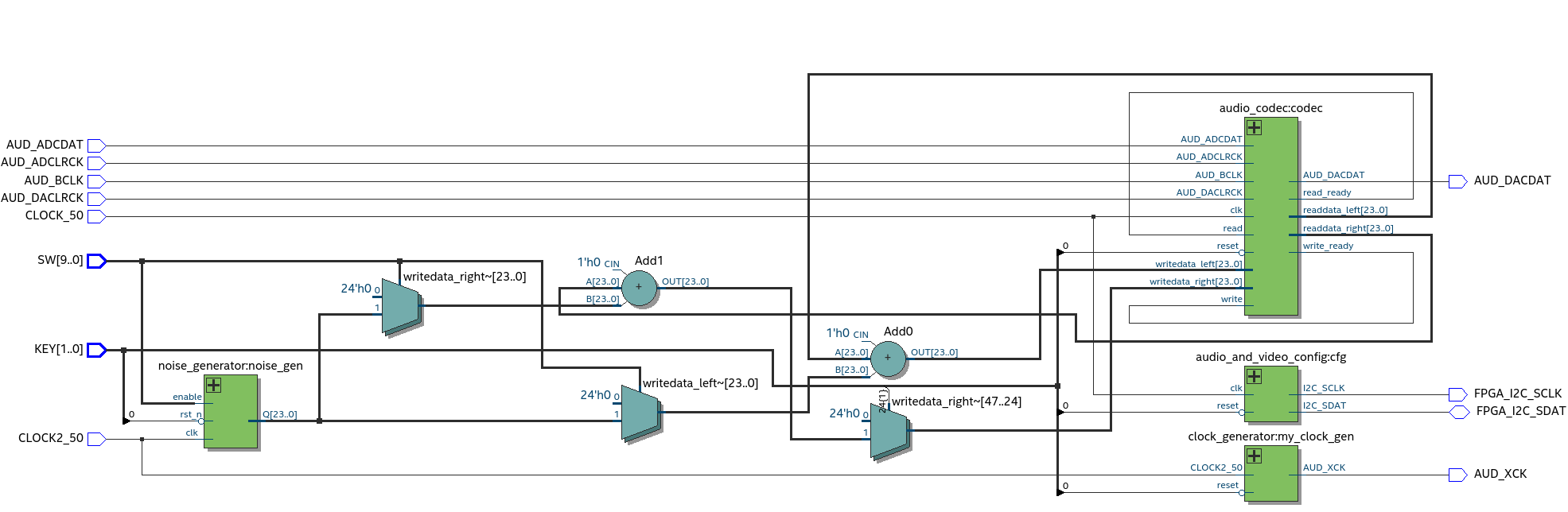
И прибавим шум к звуковым данным:

assign writedata\_left = readdata\_left + (SW[9] ? noise : 24'b0);

assign writedata\_right = KEY[1] ? readdata\_right + (SW[9] ? noise : 24'b0)

: 24'b0;

На рисунке 3 изображено RTL-представление, полученное после компиляции проекта.

Рисунок 3. RTL-представление модуля с генератором шума

Видно, что теперь оба звуковых канала из кодека сначала складываются с шумом, а потом подаются в кодек.

* 1. Реализация регулировки громкости

Код модуля регулировки громкости представлен в листинге 1 приложения. Его RTL представление можно увидеть на рисунке 4.

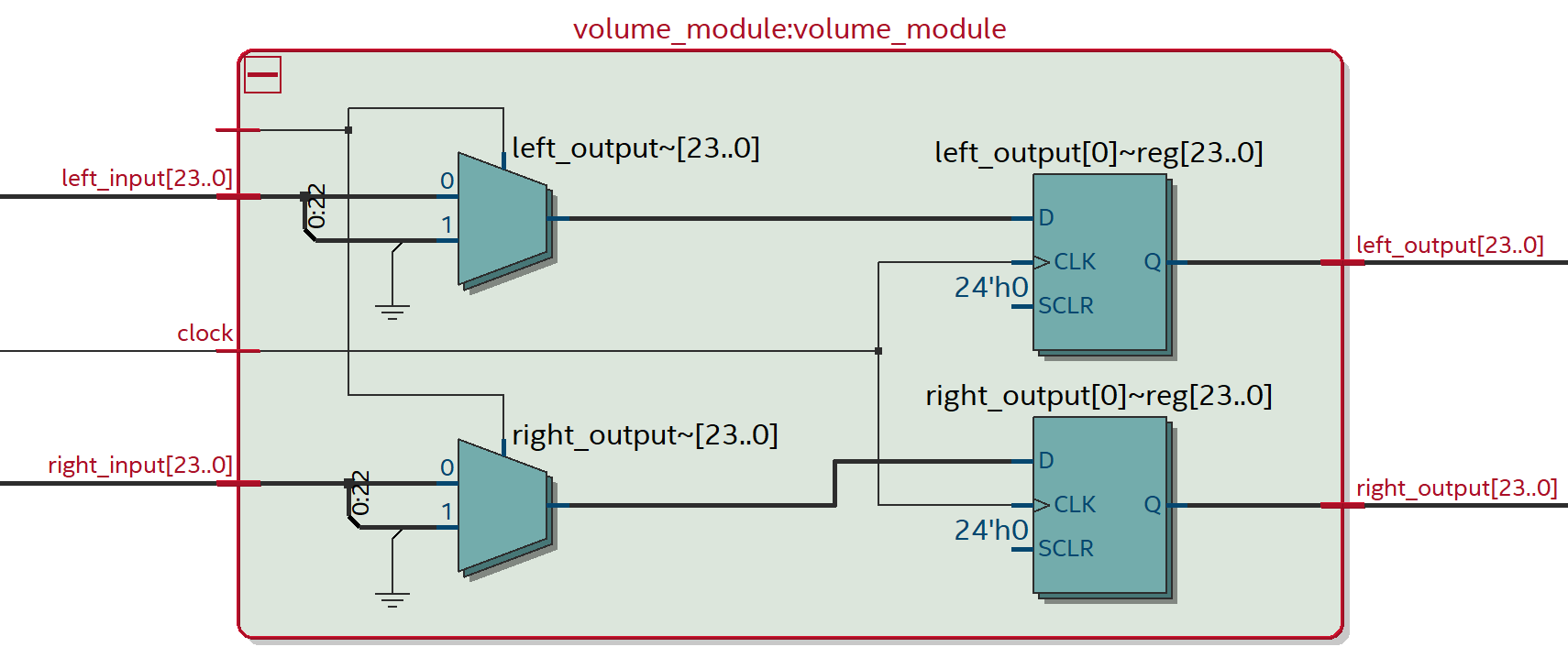


Рис. 4. RTL-представление модуля громкости

* 1. Отображение громкости на LED индикаторах

Код модуля вывода громкости на led индикатор представлен в листинге 3 приложения. Его RTL можно увидеть на рисунке 5.

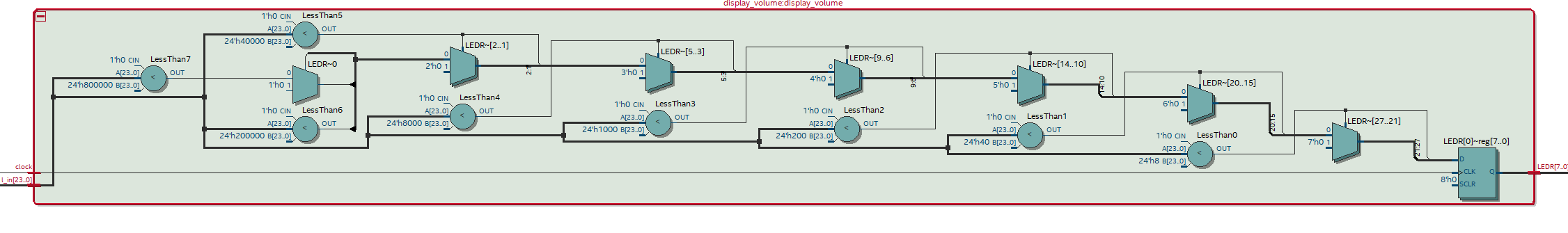


Рис. 5. RTL-представление модуля вывода громкости

1. Выводы

В ходе выполнения данной практической работы была изучена цифровая обработка сигналов на ПЛИС. В частности, научились работать со звуком, получать и выдавать, а также модифицировать звуковой сигнал. Был реализован модуль наложения шума, управление каналами. Реализовали регулировку громкости звукового сигнала и ее индикация на светодиодах.

Приложение

Листинг 1

Модуль noise\_generator

module noise\_generator (clk, enable, Q);

input clk, enable;

output [23:0] Q;

reg [2:0] counter;

always@(posedge clk)

if (enable)

counter = counter + 1’b1;

assign Q = {{10{counter[2]}}, counter, 11’d0};

endmodule

Листинг 2

Модуль регулировки громкости

module volume\_module(

input clock,

input volume\_flg,

input [23:0] left\_input, right\_input,

output reg [23:0] left\_output, right\_output

);

always@(posedge clock)

begin

left\_output = volume\_flg? left\_input\*2 : left\_input;

right\_output = volume\_flg? right\_input\*2: right\_input;

end

endmodule

Листинг 3

Модуль вывода громкости на индикаторы

module display\_volume(

input clock,

input [23:0] l\_in,

output reg [7:0] LEDR

);

always@(posedge clock)

begin

LEDR = 8'b0;

if ( l\_in< 1<<3) LEDR[0]<=1;

else if ( l\_in< 1<<6) LEDR[1]=1;

else if ( l\_in< 1<<9) LEDR[2]=1;

else if ( l\_in< 1<<12) LEDR[3]=1;

else if ( l\_in< 1<<15) LEDR[4]=1;

else if ( l\_in< 1<<18) LEDR[5]=1;

else if ( l\_in< 1<<21) LEDR[6]=1;

else if ( l\_in< 1<<23) LEDR[7]=1;

end

endmodule