РАЗРАБОТКА ГЕНЕРАТОРА СИГНАЛОВ ПРОИЗВОЛЬНОЙ ФОРМЫ ТАБЛИЧНЫМ МЕТОДОМ В ПРОГРАММНОЙ СРЕДЕ QUARTUS

Кусаинов Бахтияр Азаматович

студент, кафедра средства связи и информационная безопасность, Омский Государственный Технический университет, РФ, г. Омск

Грачев Георгий Игоревич

студент, кафедра средства связи и информационная безопасность, Омский Государственный Технический университет, РФ, г. Омск

DEVELOPMENT OF AN ARBITRARY-FORM SIGNAL GENERATOR BY A TABULAR

Bakhtiyar Kusainov

Student, Department of Communications and Information Security, Omsk State Technical University, Russia, Omsk

Grachev George

Student, Department of Communications and Information Security, Omsk State Technical University, Russia, Omsk

Аннотация. В данном курсовом проекте произведено разработка и проектирование устройства генерации сигнала табличным методом на ПЛИС при помощи программной среды Quartus на языке описания аппаратуры Verilog.

Abstract. In this course project, the development and design of a signal generation device by a tabular method on an FPGA using the Quartus software environment in the Verilog hardware description language was carried out.

Ключевые слова: Генератор, аналоговые сигналы, цифровые сигналы, ПЛИС, Quartus, Программируемая логическая интегральная схема, проектирование, Verilog.

Keywords: Generator, analog signals, digital signals, FPGA, Quartus, Programmable logic integrated circuit, design, Verilog.

Инновационные методы решения экологических проблем существуют.

В Verilog память – это массив регистров. Объявляется следующим образом: reg [7:0] mem_name[1023:0]; «mem_name» память, которая содержит 1024 ячеек, каждая ячейка состоит из 8-битного регистра.

Доступ к отдельной ячейке осуществляется так: dac1_d <= mem_name[124] – 8-ми битное значение из 124-ой ячейки записывается в 8- битный регистр dac1_d.

Доступ к отдельному биту ячейки осуществляется так: assign led[0] <= mem_name[124][7] – Значение старшего (7-го) бита 124-ой ячейки назначается 0-му биту шины led.

Доступ к группе бит ячейки осуществляется так: assign led <= mem_name[124][7:4] Значение четырех старших бит 124-ой назначается шине led.

Память может быть инициализирована начальными значениями с

помощью системных функций readmemh и readmemb. При компиляции проекта функции считывают значение из текстового файла и помещают их в указанную память. Файл для функции readmemh должен содержать данные в шестнадцатеричном формате, для функции readmemb – в двоичном.

Вызов функций должен происходить в блоке инициализации Inicial.

Общий синтаксис вызова следующий: $readmemh (" file_name " , memory_name [, start_addr [, finish_addr]]); где file_name – относительный путь к файлу и имя, memory_name – имя инициализируемой памяти, start_addr – начальный адрес для загрузки, finish_addr – конечный адрес. Последние два параметра опциональны.

Программный код генератора, а также табличные значения в текстовом файле показаны на рисунке 1 и 2.

```
1  module course(clk_50MHz, dac1, dac2);
2
3      input clk_50MHz;            // Вход с тактового генератора
4      reg [13:0] dac1_d,          // Шина данных ЦАП1
5                 dac2_d;          // Шина данных ЦАП2
6
7      output [13:0] dac1, dac2;
8
9      reg [5:0] rom_1[63:0];                  // Объявляем ROM. Объем: 64 ячеек,
10                                             // разрядность ячеек: 6бит.
11     initial                                 // инициируем память значениями из файла
12     begin                                   // Требуется указать путь к файлу с таблицей
13         $readmemh("C:/users/Dmitry/Downloads/sin_full_64_6bit.txt", rom_1);
14     end
15
16     reg [7:0] phase_acc_1;                  // Регистр аккумулятора фазы
17
18     // Фазовый аккумулятор
19     always @(posedge clk_50MHz)
20     begin
21         phase_acc_1 <= phase_acc_1 + 1;     // Прибавляем код частоты
22     end
23
24     always @(posedge clk_50MHz)
25     begin
26         dac1_d <= {phase_acc_1[7:0], 6'h0};         // На ЦАП1 выводим значение фазы
27         dac2_d <= {rom_1[phase_acc_1[7:0]], 6'h0};// На ЦАП2 выводим сигнал
28     end
29     assign dac1 = dac1_d;
30     assign dac2 = dac2_d;
31 endmodule
```

Рисунок 1. Программный код генератора

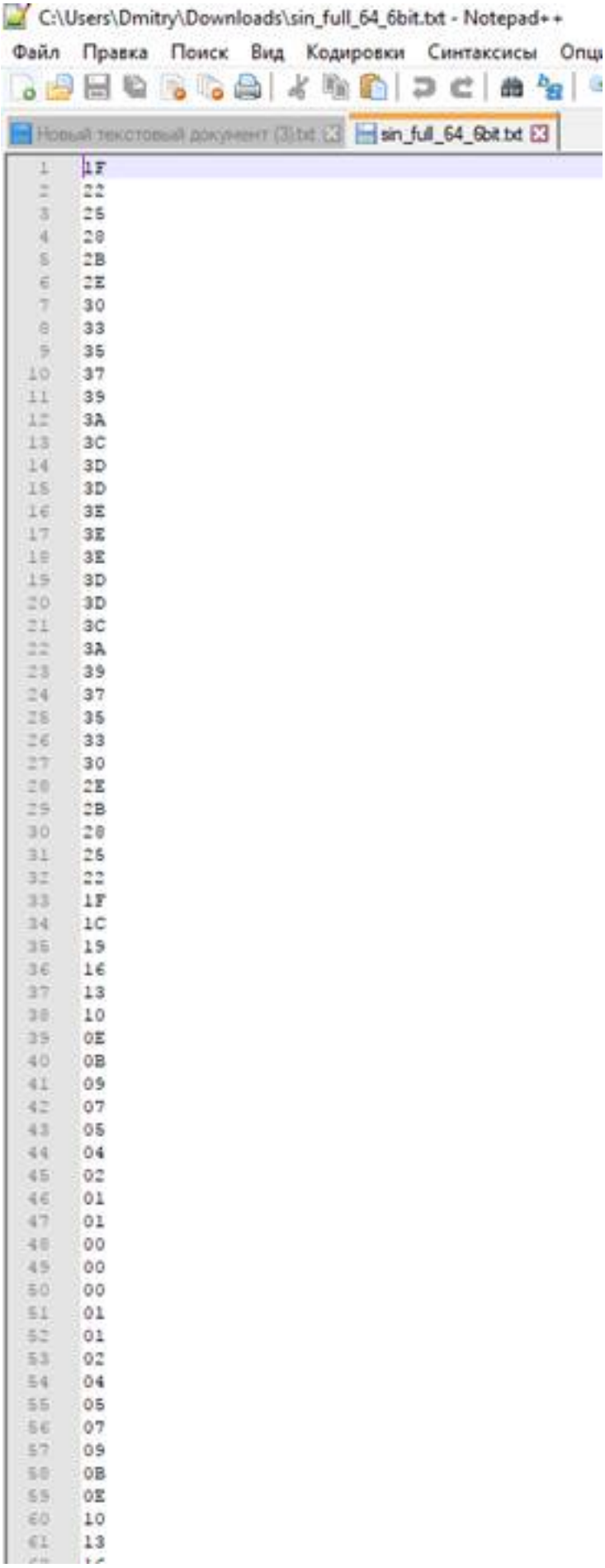

Рисунок 2. Текстовый файл с значениями

На 11 строчке кода происходит чтение текстового файла, в котором записаны 64 отсчета по 6 бит. Данные значения заносятся 6 битный массив регистров rom_1 на 64 ячейки.

По каждому положительному перепаду генератора CLK происходит инкрементация регистра фазового аккумулятора, а также внесение в регистры dac1_d и dac2_d значений. Регистр dac1_d формирует синусоидальный сигнал, dac2_d – пилообразный.

С помощью встроенных инструментов произведена симуляция проекта (рисунок 3).

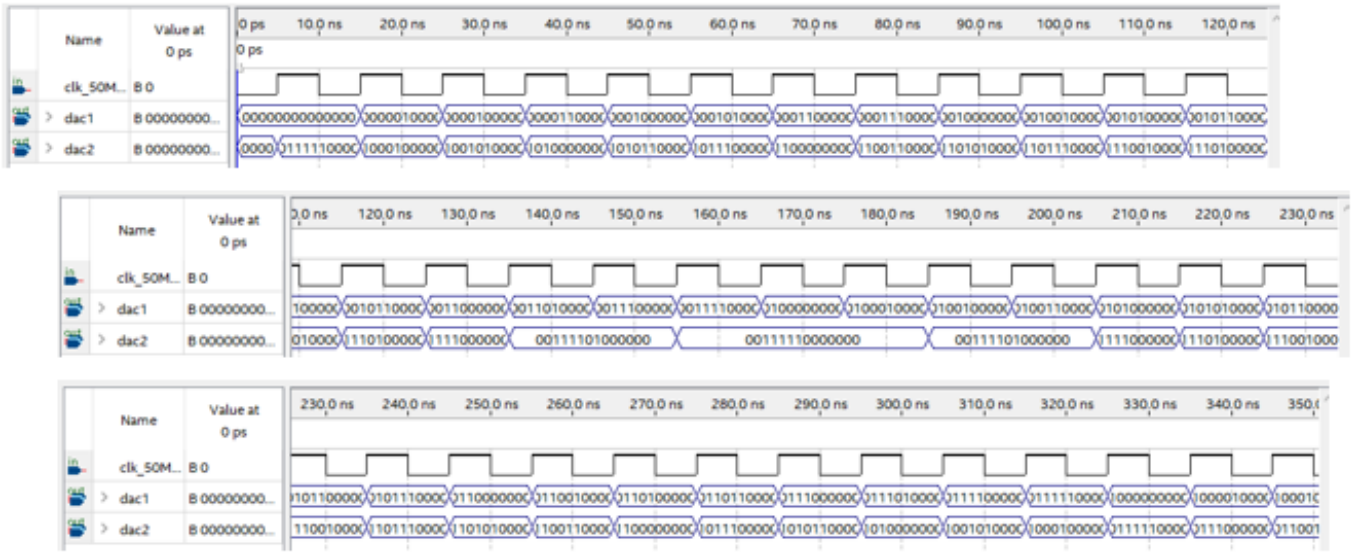

Рисунок 3. Симуляция работы программы

Заключение

Генераторы сигнала являются неотъемлемой частью современной цифровой индустрии. В ходе выполнения курсового проекта были изучены различные способы генерации сигналов, а также виды генераторов.

В программной среде Quartus разработан работоспособный образец генератора сигнала табличным методом, задаваемый в отдельном текстовом файле. Устройство способно генерировать сигнал с 64 отсчетами на период с 6 битной точностью.

Список литературы:

1. Берикашвили В.Ш. Электроника и микроэлектроника: импульсная и цифровая электроника, 2018 г. – 188 с.

2. Википедия. Свободная энциклопедия [Электронный ресурс]. Режим доступа: URL

https://ru.wikipedia.org/wiki/Шифратор

3. Миловзоров О. В. Электроника, 2015 г.

4. Андык В.С. Автоматизированные системы управления технологическими процессами на ТЭС, 2018 г. – 248 с.

5. Кузовкин В.А. Электротехника и электроника, 2018 г. – 226 с.