

Использование GPIO выводов платы De1-SoC

Плата De1-SoC в своем составе имеет 80 пользовательских GPIO выводов. Для удобства подключения периферийных устройств GPIO порты разделены на 2 независимые группы (банки) по 40 выводов. В качестве выводов для обмена данными доступны 36 выводов, остальные имеют фиксированные назначение – 2 вывода питания (5 V, 3.3 V), а также 2 вывода «земля». Структура распределения GPIO выводов представлена на рисунке 1.

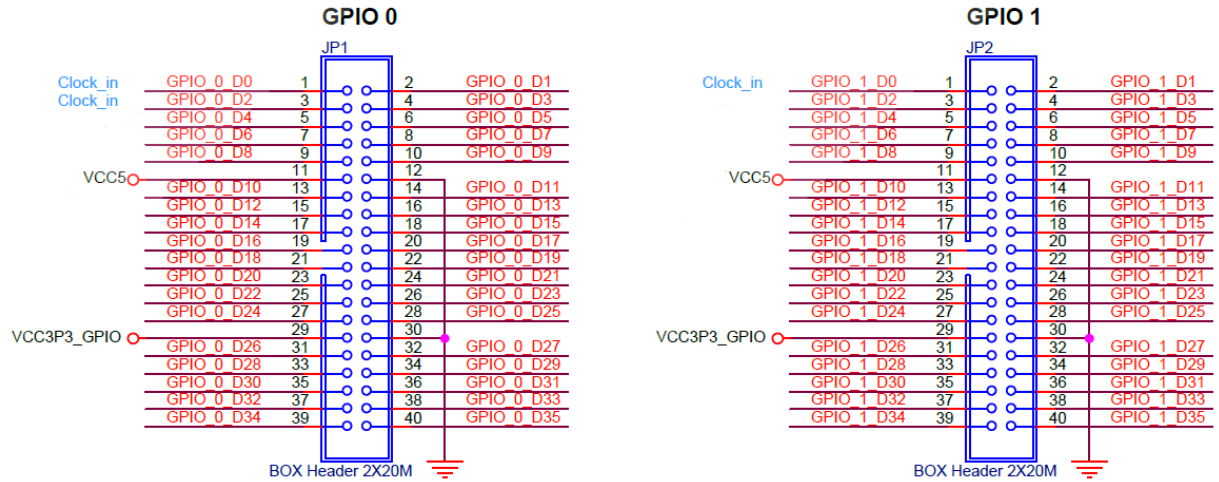


Рис. 1. Распределение GPIO выводов по группам (банкам) на плате De1-SoC.

Для примера использования GPIO выводов, подключим к одному из выводов светодиод и напомним модуль, который с заданной частотой будет мигать светодиодом. Схема подключения светодиода показана на рис. 2.

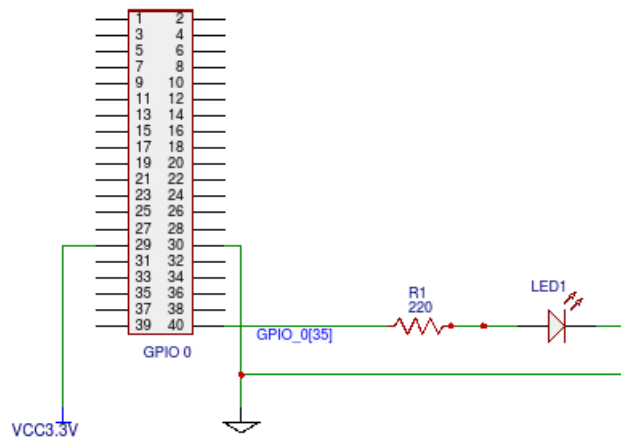


Рис. 2. Схема подключения светодиода.

В данной схеме используется светодиод LED1, а также резистор, номиналом 220 Ом, для ограничения прохождения тока через светодиод.

Создаем новый проект для DE1-SOC в САПР Quartus II и добавляем новый Verilog HDL файл. Пример кода модуля приведен ниже.

```

1 module blink_led_gpio(
2     clk,
3     gpio_0_35
4 );
5
6     input clk;
7     output reg gpio_0_35;
8     reg turn;
9
10    initial
11    begin
12        turn = 1'b0;
13    end
14
15    always @(posedge clk)
16    begin
17        turn = turn + 1'b1;
18        gpio_0_35 = turn;
19    end
20
21 endmodule

```

Рис. 3. Код модуля, отвечающего за мигание светодиода.

Чтобы можно было увидеть мигание светодиода, создадим вспомогательный модуль, который будет понижать частоту тактового сигнала с 50 MHz до 1 Hz, на которой будет работать светодиод.

```

1 module Clock_1hz(
2     input_clk,
3     output_clk
4 );
5
6     input input_clk;
7     output reg output_clk;
8
9     reg signal;
10    reg[24:0] counter;
11    reg[24:0] const_data = 25'b101111101011100001000000;
12
13    initial
14    begin
15        signal = 1'b0;
16        counter = 25'b0;
17    end
18
19    always@(posedge input_clk)
20    begin
21        counter = counter + 1'b1;
22        if(counter == const_data)
23        begin
24            signal = ~signal;
25            counter = 25'b0;
26        end
27        output_clk = signal;
28    end
29
30 endmodule

```

Рис. 4. Код модуля делителя частоты.

Добавляем новый BDF файл, в котором зададим схему подключения модулей.

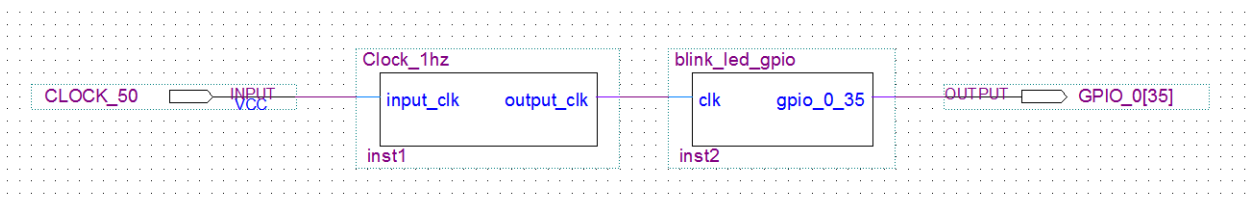


Рис. 5. BDF схема для примера подключения светодиода.

По умолчанию, GPIO выводы в De1-SoC используют I/O стандарт логического уровня 2.5V. Для удобства использования выводов (т.к. в банке GPIO присутствует вывод питания 3.3V), необходимо задать стандарт логического уровня 3.3V. Для этого, вызываем редактор Pin Planner (Assignments -> Pin Planner) и у используемого GPIO порта меняем свойство I/O Standart с 2.5 V (default) на 3.3-V LVCMOS. Если для питания подключенных внешних устройств не планируется использовать выводы на плате, то можно использовать иной стандарт логических уровней, наиболее подходящий для решаемой задачи.

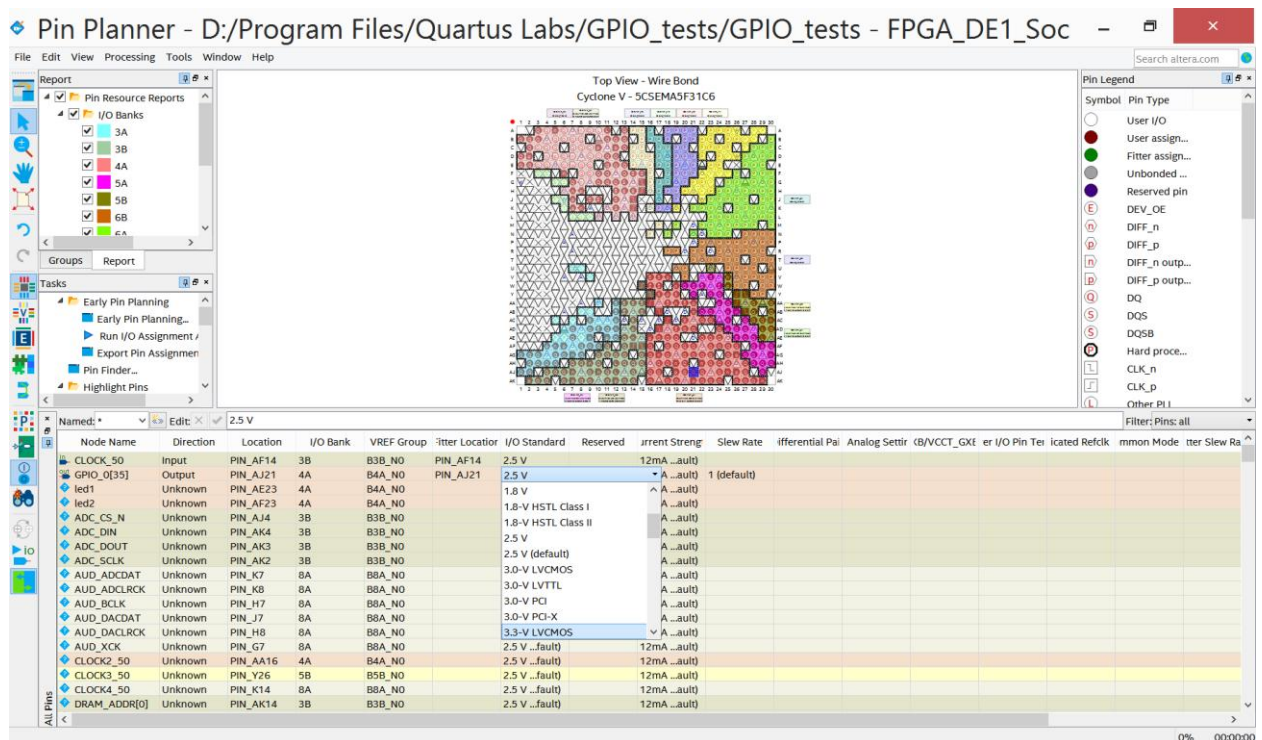


Рис. 6. Изменение I/O стандарта используемого GPIO вывода.

Затем компилируем проект и, если во время компиляции не появилось ошибок, загружаем скомпилированный *.SOF файл на плату. В результате светодиод должен мигать 1 раз в секунду.

Взаимодействие с внешними устройствами

В ранее приведенном примере GPIO выводы использовались для управления внешними подключенными устройствами. С помощью этих же выводов можно также принимать данные от подключенных устройств. GPIO входы могут одновременно использоваться и как input выходы, также как и output выходы.

Для демонстрации примера их использования подсоединим плату Arduino Mega к De1-SoC. Arduino будет передавать данные о включении определенных выводов на De1-SoC, которые, в свою очередь, подключены к светодиодам. Подсоединим выводы 10, 11, 12 Arduino к выводам 30, 32, 34 соответственно на De1-SoC, а выводы 31, 33, 35 De1-SoC подключим к светодиодам. Подключенные к Arduino выводы De1-SoC будут передавать их состояние на светодиоды.

Непосредственно подключить Arduino к De1-SoC нельзя, поскольку используются разные стандарты логических уровней. Arduino поддерживает стандарт 5 V, в то время, как максимально возможный стандарт для De1-SoC – 3.3 V. Поэтому, в разрыв соединения необходимо подключить преобразователь логических уровней.

Задача преобразователя логических уровней – переводить (согласовывать) сигнал из одного стандарта в другой. Без такого согласования, при передаче сигнала от устройства с более высоким стандартом логического уровня к устройству с более низким стандартом логического уровня, велика вероятность вывести из строя устройство с более низким стандартом логического уровня.

Существует множество способов согласовывать логические уровни; одним из самых распространенных является использование транзисторов. Ниже приведена схема используемого в проекте преобразователя на базе bss138. Основным достоинством данного преобразователя является двунаправленность. Также он подходит для использования в таких интерфейсах передачи данных, как I2C и SPI.

Общая схема подключения платы Arduino, преобразователя логических уровней с GPIO De1-SoC и светодиодами показана на рис. 7.

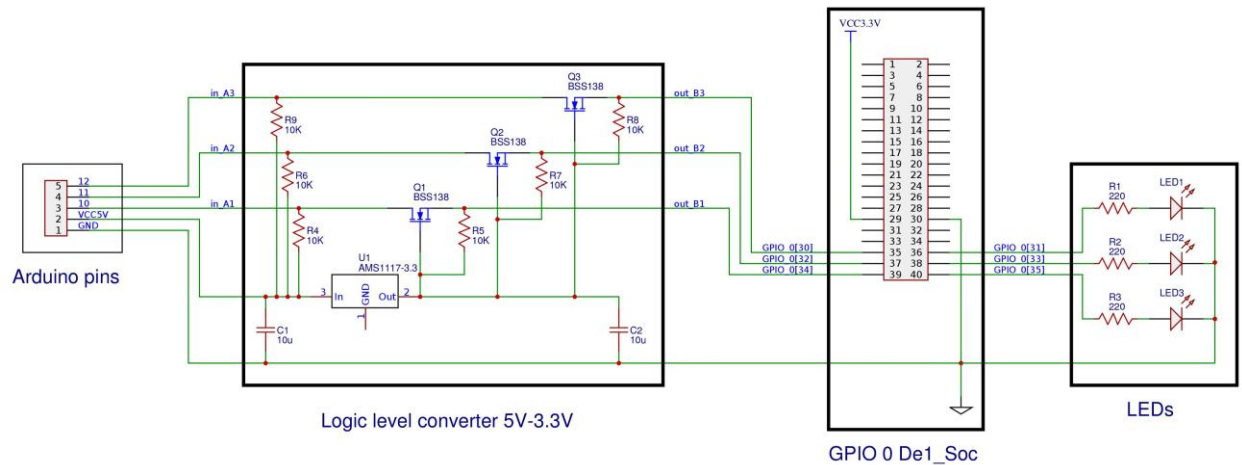


Рис. 7. Схема подключения элементов микросхемы.

Создадим два проекта. Первый проект предназначен для платы Arduino. В нем будет осуществляться управление состоянием выводов на Arduino.

Если вы не работали с платами Arduino, для начала необходимо скачать Arduino IDE (<https://www.arduino.cc/en/Main/Software>) и, следуя указаниям установщика, выполнить инсталляцию IDE, а также необходимых драйверов. Далее необходимо скачать архив **arduino_to_de1.zip** с проектом для платы Arduino, распаковать архив и открыть файл проекта в Arduino IDE.

```

arduino_to | Arduino 1.6.5
Файл Правка Элементы Инструменты Помощь

//arduino_to
1 int pin_a1 = 10;
2 int pin_a2 = 11;
3 int pin_a3 = 12;
4
5 int data1 = 0;
6 int data2 = 0;
7 int data3 = 0;
8 int incomingByte = 0;
9 int counter = 1;
10
11 void setup() {
12   pinMode(pin_a1, OUTPUT);
13   pinMode(pin_a2, OUTPUT);
14   pinMode(pin_a3, OUTPUT);
15   Serial.begin(9600);
16   Serial.println("number of set");
17 }
18
19 void loop() {
20   if (Serial.available() > 0) {
21     incomingByte = Serial.read();
22     if (incomingByte == 48) {
23       switch (counter) {
24         case 1:
25           data1 = 0;
26           counter++;
27           Serial.print("set data1 "); Serial.println(data1);
28           break;
29         case 2:
30           data2 = 0;
31           Serial.print("set data2 "); Serial.println(data2);
32           counter++;
33           break;
34         case 3:
35           data3 = 0;
36           Serial.print("set data3 "); Serial.println(data3);
37           digitalWrite(pin_a1, data1);
38           digitalWrite(pin_a2, data2);
39           digitalWrite(pin_a3, data3);
40           Serial.print("set "); Serial.print(data1); Serial.print(" "); Serial.print(data2); Serial.print(" "); Serial.print(data3);
41           Serial.println("");
42           counter=1;
43           break;
44       }
45     }
46     if (incomingByte == 49) {
47       switch (counter) {
48         case 1:
49           data1 = 1;
50           Serial.print("set data1 "); Serial.println(data1);
51           counter++;
52           break;
53         case 2:
54           data2 = 1;
55           Serial.print("set data2 "); Serial.println(data2);
56           counter++;
57           break;
58         case 3:
59           data3 = 1;
60           Serial.print("set data3 "); Serial.println(data3);
61           digitalWrite(pin_a1, data1);
62           digitalWrite(pin_a2, data2);
63           digitalWrite(pin_a3, data3);
64           Serial.print("set "); Serial.print(data1); Serial.print(" "); Serial.print(data2); Serial.print(" "); Serial.print(data3);
65           Serial.println("");
66           counter=1;
67           break;
68       }
69     }
70   }
71 }

```

Рис. 8. Код программы для Arduino.

На следующем шаге необходимо подключить плату Arduino через USB провод к компьютеру. Далее, во вкладке «Инструменты -> Плата» необходимо выбрать тип подключенной платы (Arduino / Genuino Mega or Mega 2560), во вкладке «Инструменты -> Процессор» выбрать тип процессора (ATmega2560 (Mega 2560)). Во вкладке «Инструменты -> Порт» необходимо выбрать COM порт, к которому подключена плата. У COM порта, к которому подключена плата, будет указан тип платы, на пример COM12 (Arduino/Genuino Mega or Mega 2560).

Второй проект создается в Quartus II для платы De1-SoC. В нем необходимо создать Verilog файл, в котором необходимо описать устройство передачи полученных данных о состоянии подключенных к Arduino выводов, на выводы, подключенные к светодиодам.

```

1 module arduino_to_de1(
2     clk,
3     b1,
4     b2,
5     b3,
6     output_led1,
7     output_led2,
8     output_led3
9 );
10 input clk;
11 input b1;
12 input b2;
13 input b3;
14 output reg output_led1;
15 output reg output_led2;
16 output reg output_led3;
17
18 initial
19 begin
20     output_led1 = 1'b0;
21     output_led2 = 1'b0;
22     output_led3 = 1'b0;
23 end
24
25 always@(posedge clk)
26 begin
27     output_led1 = b1;
28     output_led2 = b2;
29     output_led3 = b3;
30 end
31 endmodule

```

Рис. 9. Код модуля, который управляет светодиодами.

Создадим BDF файл, в котором зададим подключение модулей:

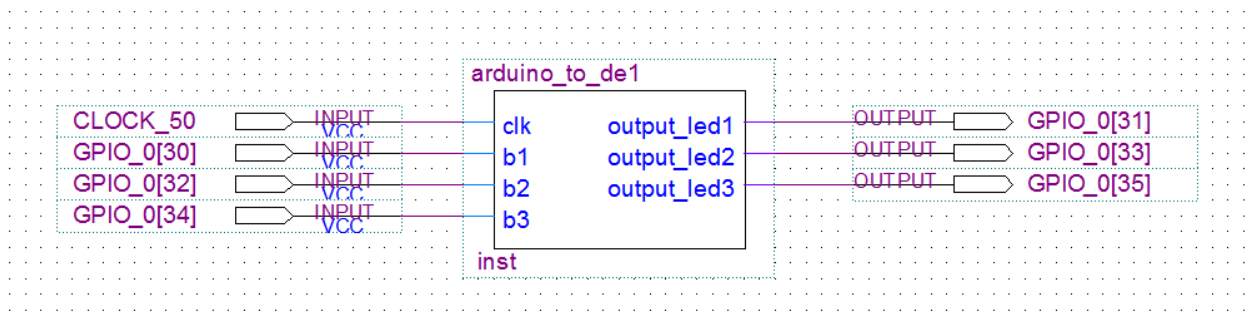


Рис. 10. BDF схема примера подключения Arduino к De1-SoC.

Скомпилируем проект. Во время компиляции должно появиться сообщение об ошибке «Error 275088: Inconsistent I / O type for element GPIO_0».

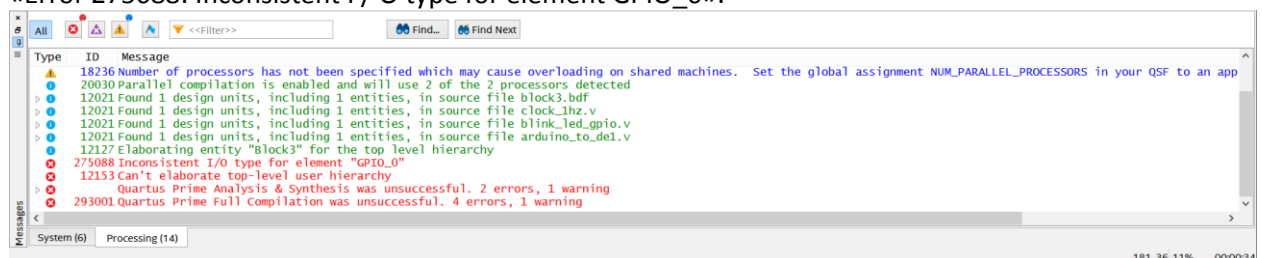


Рис. 11. Ошибка при компиляции проекта с разным определением выводов.

Это связано с тем, что GPIO выводы в рамках банка организованы в виде шины. Поэтому при назначении части выводов как input выводов, а части – как output, компилятор не может понять, какой тип имеет банк (шина) – input или output. Чтобы избежать данной ошибки при

необходимости использовать выводы в банке в разных направлениях, необходимо всем выводам из банка задавать один и тот же тип – bidir. Таким образом, определяют выводы как двунаправленные. В модуле для пинов можно использовать либо тип inout, либо более привычные объявления – input / output.

Правильное определение GPIO выводов, не приводящее к возникновению ошибок показано ниже.

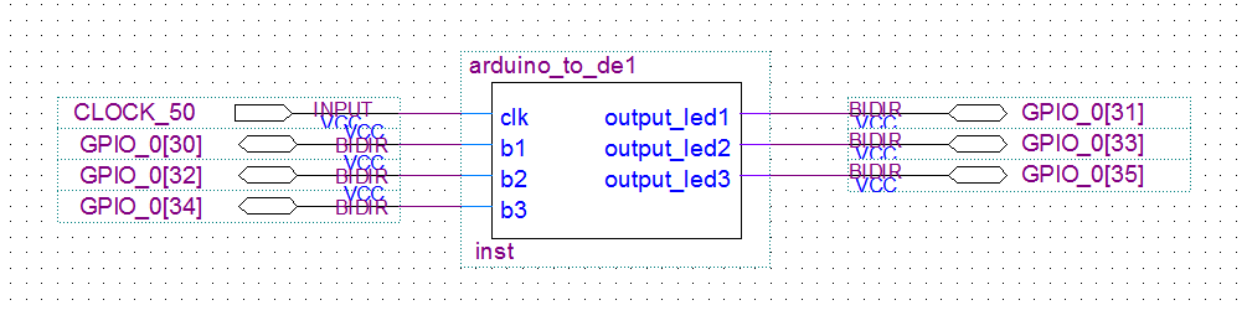


Рис. 11. Исправленная BDF схема для примера подключения Arduino к De1-SoC.

В результате получили схему управления светодиодами с компьютера посредством Arduino и платы De1-SoC; ее внешний вид приведен ниже:

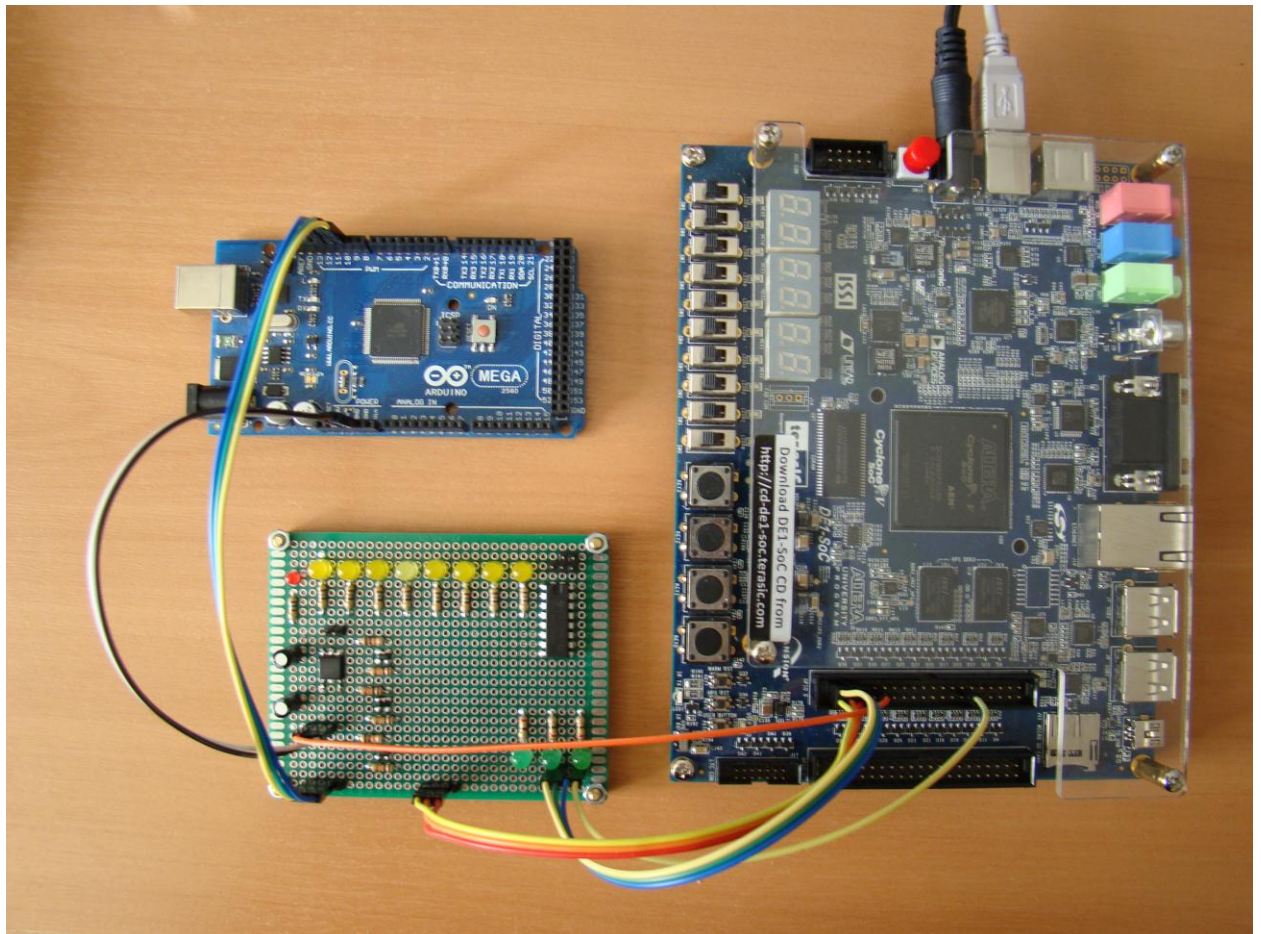


Рис. 12. Схема взаимодействия Arduino и платы De1-SoC.

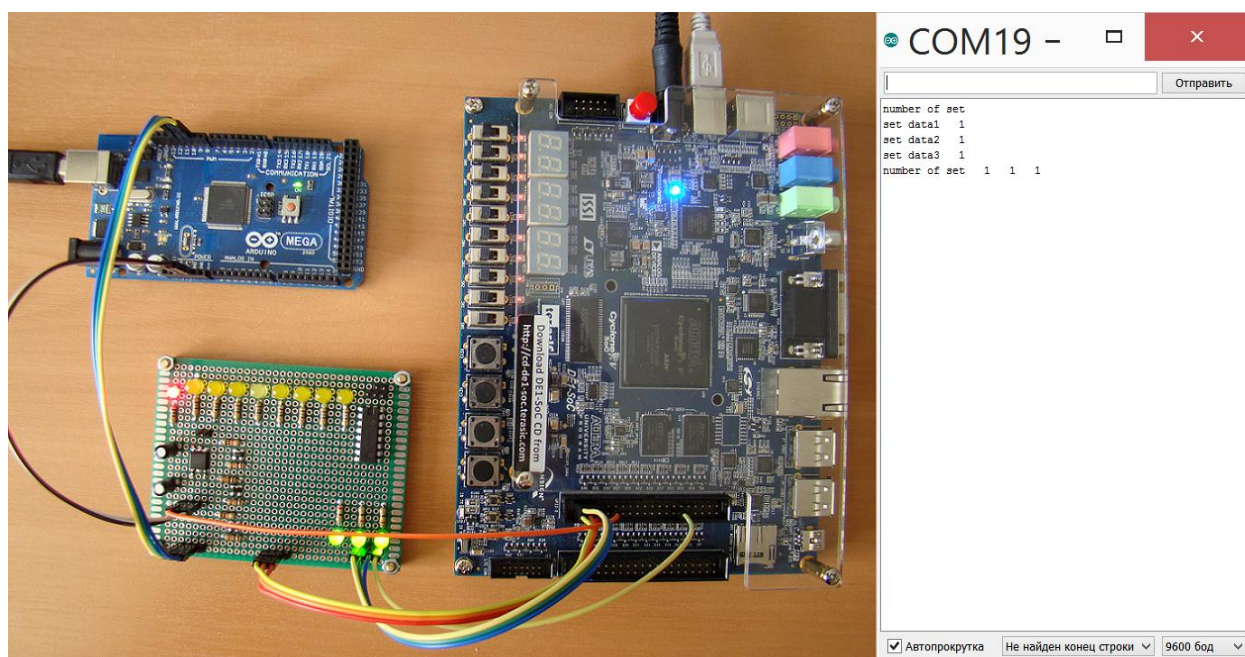


Рис. 13. Демонстрация передачи данных с COM порта через Arduino на плату De1-SoC (1 часть).

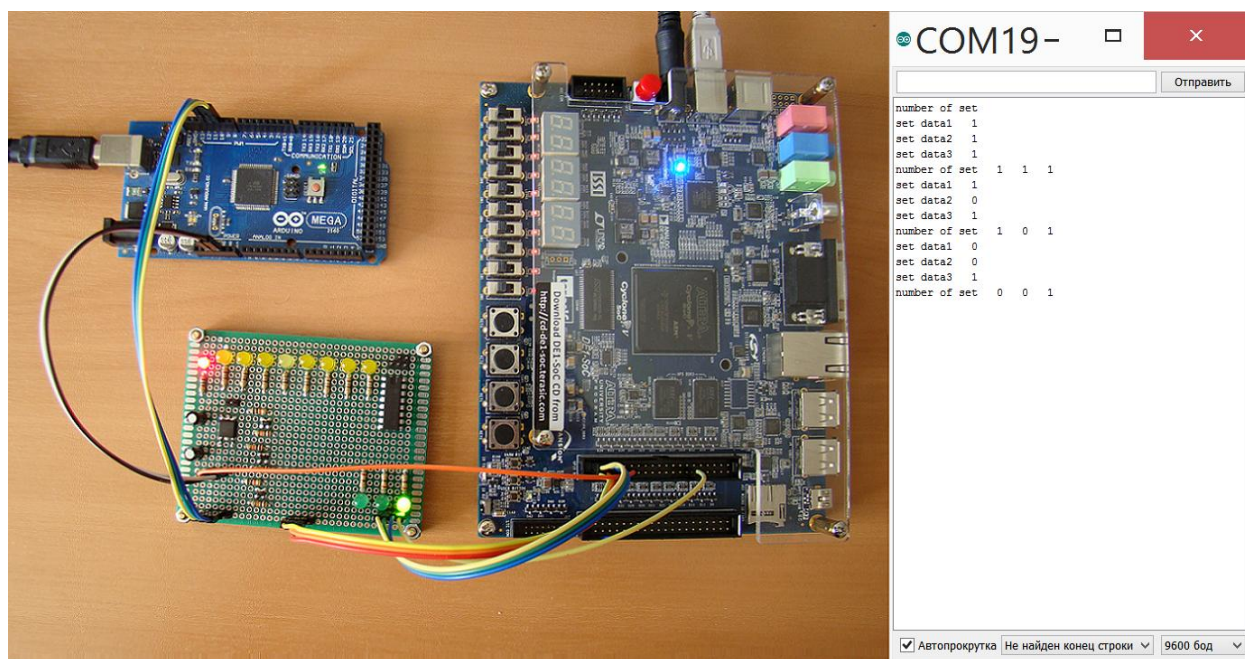


Рис. 14. Демонстрация передачи данных с COM порта через Arduino на плату De1-SoC (2 часть).

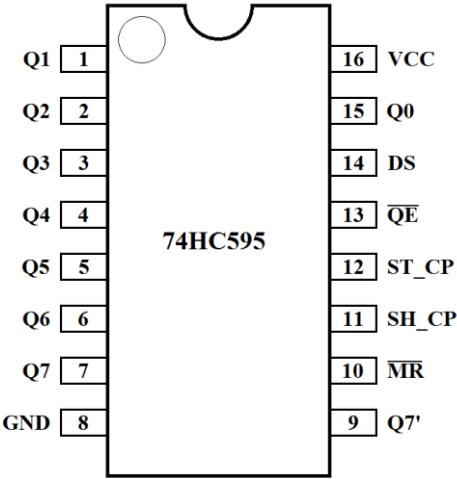
Подключение сдвигового регистра 74НС595.

Сдвиговые регистры представляют собой набор последовательно соединенных триггеров. В микросхеме 74НС595 таких триггеров 8. Такое соединение позволяет преобразовывать последовательный код в параллельный и наоборот. Последовательный код используется для передачи двоичной информации по ограниченному количеству проводников, тем самым значительно упрощая подключение различных элементов и уменьшая количество используемых выводов микроконтроллера.

Назначение выводов сдвигового регистра 74НС595 приведено в таблице 1.

Таблица 1.

Выводы сдвигового регистра 74HC595

	Выводы 1-7, 15	Q0-Q7	Параллельные выходы
	Вывод 8	GND	Земля
	Вывод 9	Q7'	Выход для последовательного соединения регистров
	Вывод 10	MR	Сброс значений регистра. Сброс происходит при получении LOW
	Вывод 11	SH_CP	Вход для тактовых импульсов (clockPin)
	Вывод 12	ST_CP	Синхронизация (защелкивание) выходов (latchPin)
	Вывод 13	OE	Вход для переключения состояния выходов из высокоомного в рабочее
	Вывод 14	DS	Вход для последовательных данных (dataPin)
	Вывод 16	Vcc	Питание

Сдвиговый регистр 74HC595 работает по интерфейсу SPI и способен переводить из последовательного кода в параллельный 1 байт данных. Выводы DS, ST_CP, SH_CP – это шины управления (шина данных, защелка, линия тактирования). Выводы VCC и GND служат для подключения питания микросхемы. Через выводы Q0–Q7 происходит управление подключенными устройствами – последовательно принятый микросхемой байт, побитно передается на выходы Q0-Q7.

Подключение сдвигового регистра не представляет особой трудности: на вывод VCC подается питание +5V, вывод GND соединяется с аналогичным выводом DE1-SoC. Так как микросхема работает со стандартом логических уровней 5V, то управляющие выводы SH_CP, ST_CP, DS подключаются к выводам GPIO_0[34], GPIO_0[32], GPIO_0[30] через преобразователь логических уровней. Выводы MR и OE подключит к выводам +5V и GND соответственно. Полная схема подключения показана на рисунке 15. Это упрощенный вариант подключения, т.к. в момент подачи питания на микросхему, на ее выходах будут случайные значения. Чтобы это исправить, можно контролировать сигналы на выводах MR и OE.

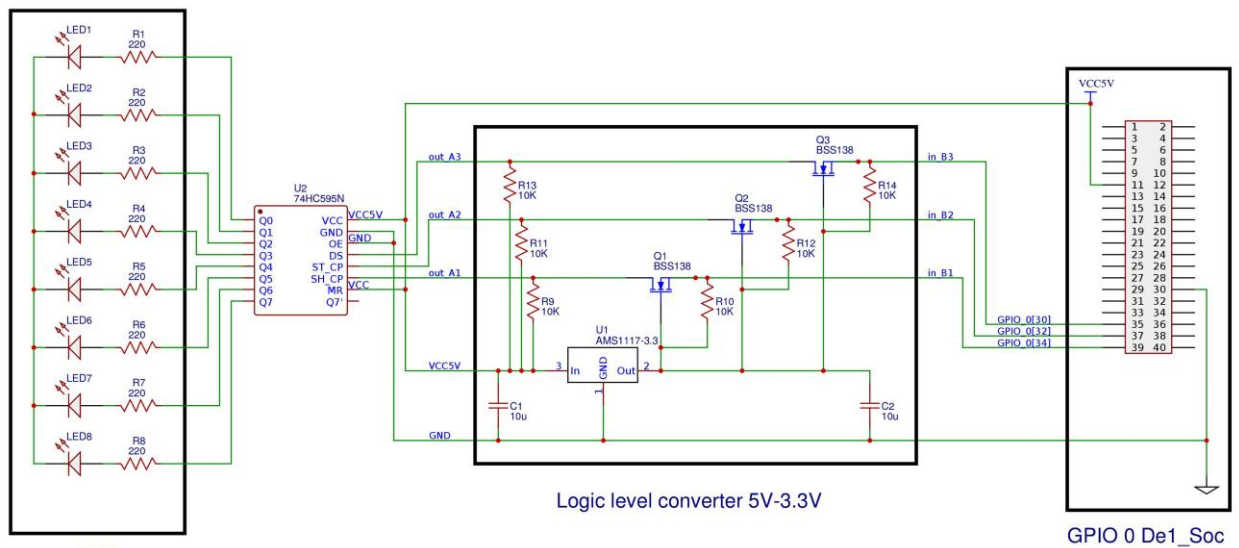


Рис. 15. Схема подключения сдвигового регистра.

На временной диаграмме показаны сигналы на выводах сдвигового регистра. Следует отметить, что порядок следования импульсов очень важен. Сначала на выводе DS устанавливают нужный уровень сигнала, и только потом подается импульс на SH_CP. Когда SH_CP выход переключается с LOW на HIGH, регистр считывает значения с DS вывода. По мере считывания данные записываются во внутреннюю память регистра. Когда ST_CP вывод переключается с LOW на HIGH, данные "защелкиваются", то есть передаются на выходы регистра, и от туда – на светодиоды.

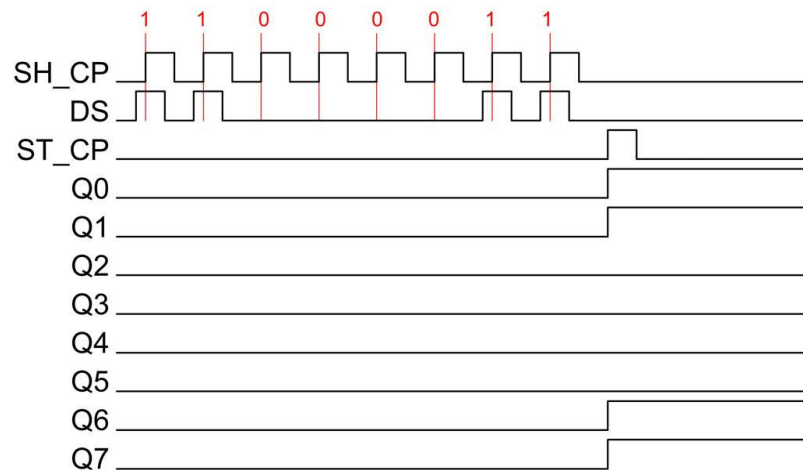


Рис. 16. Управление сдвиговым регистром на логическом уровне.

Пример кода Verilog для управления сдвиговым регистром приведен на рисунке 17.

```

1 module shift_register(
2     clk,
3     data_pin,
4     latch_pin,
5     test_out_led_data_pin,
6     test_out_led_latch_pin,
7     test_out_led_counter
8 );
9
10 input clk;
11 output reg data_pin;
12 output reg latch_pin;
13
14 reg[0:7] data;
15 reg transfer_data;
16 integer counter;
17
18 initial
19 begin
20     transfer_data = 1'b0;
21     data_pin = 1'b0;
22     latch_pin = 1'b1;
23     data = 8'b00111100;
24 end
25
26 always@(posedge clk)
27 begin
28     if(transfer_data == 1'b0)
29     begin
30         transfer_data = 1'b1;
31         latch_pin = 1'b0;
32         counter = 0;
33     end
34     if(transfer_data == 1'b1)
35     begin
36         if(counter == 8)
37         begin
38             latch_pin = 1'b1;
39             counter = 0;
40             transfer_data = 1'b0
41         end else
42         begin
43             data_pin = data[counter];
44             counter = counter+1;
45         end
46     end
47 end
48 endmodule
49

```

Рис. 17. Управление сдвиговым регистром (код Verilog).

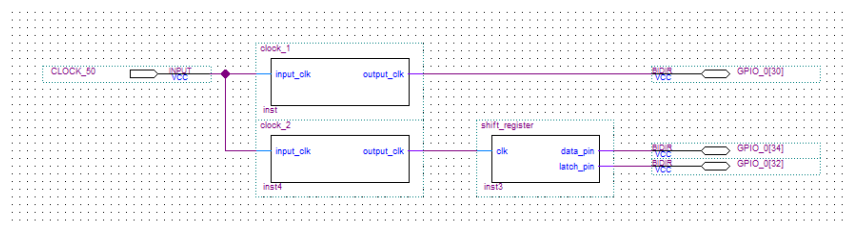


Рис. 18. BDF схема управления сдвиговым регистром.

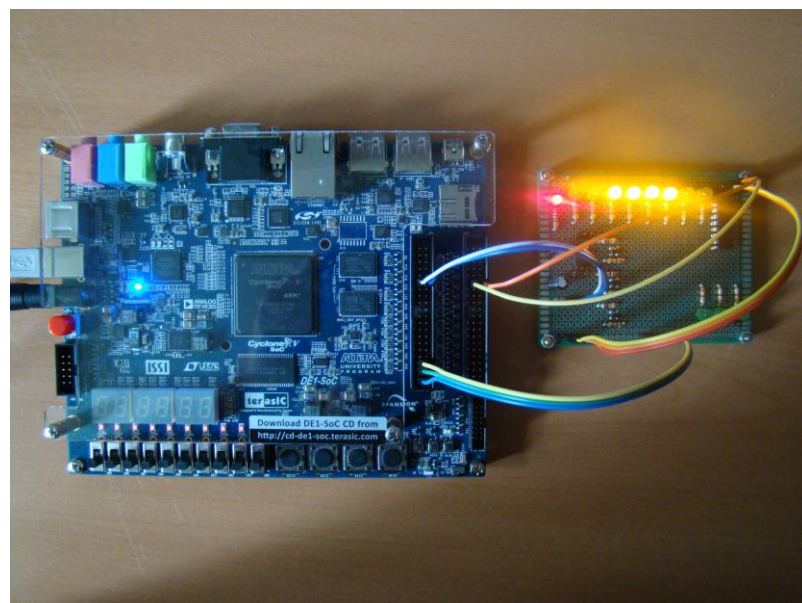


Рис. 19. Пример работы сдвигового регистра.