

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа компьютерных технологий и информационных систем

Отчёт по курсовой работе CR_1

Дисциплина: Автоматизация проектирования дискретных устройств (на
английском языке)

Выполнил студент гр. 5130901/10101 _____ М.Т. Непомнящий
(подпись)

Руководитель _____ А.А. Федотов
(подпись)

Санкт-Петербург

2024

Оглавление

1.	Задание	4
1.1.	Цель работы	4
2.	Ход решения	5
2.1.	Модуль GEN	5
2.1.1.	Создание модуля	5
2.1.2.	Тестирование	7
2.2.	Модуль RaF	8
2.2.1.	Создание модуля	8
2.2.2.	Тестирование	9
2.3.	Модуль CR_1	10
2.3.1.	Создание модуля	10
2.3.2.	Тестирование	12
2.4.	Создание модуля для тестирования на плате	14
2.5.	Настройка Signal Tap II	15
2.6.	Тестирование на плате средствами Signal Tap II	16
2.7.	Создание модуля имплементации и запуск на плате	17
3.	Вывод	18

Список иллюстраций

Рис. 1 – Структура разрабатываемого устройства	4
Рис. 2 – Модуль интерфейса lab_MS_SV5_interface	5
Рис. 3 – Модуль gen.....	6
Рис. 4 – RTL Viewer модуля gen	6
Рис. 5 – Тестовый файл первого класса для модуля gen	7
Рис. 6 – Моделирование тестового файла средствами ModelSim (gen_tb)	7
Рис. 7 – Модуль RaF	8
Рис. 8 – RTL Viewer модуля RaF.....	8
Рис. 9 – Тестовый файл первого класса для модуля RaF.....	9
Рис. 10 – Моделирование тестового файла средствами ModelSim (RaF_tb)	9
Рис. 11 – Модуль CR_1	10
Рис. 12 – Настройка модуля FIFO.....	11
Рис. 13 – RTL Viewer модуля CR_1	11
Рис. 14 - Тестовый файл первого класса для модуля CR_1.....	12
Рис. 15 – Тестирование модуля CR_1 до сигнала full.....	13
Рис. 16 – Тестирования модуля CR_1 до сигнала empty	13
Рис. 17 – Тестирование модуля CR_1 до максимального значения генератора	13
Рис. 18 – Тестирования модуля CR_1 на сигнале RST	13
Рис. 19 – Модуль db для тестирования на плате	14
Рис. 20 – RTL Viewer для тестового файла модуля CR_1	14
Рис. 21 – Настройка окна Signal Tap II	15
Рис. 22 – Мнемоническая таблица	15
Рис. 23 – Временные характеристики устройства.....	15
Рис. 24 – Появление сигнала full при тестировании на плате.....	16
Рис. 25 – Появление сигнала empty при тестировании на плате	16
Рис. 26 – Появление значения 127 (максимального) при тестировании на плате	16
Рис. 27 – Отключение работы при тестировании на плате	16
Рис. 28 – Сброс при тестировании на плате.....	16
Рис. 29 – Модуль имплементации.....	17
Рис. 30 – RTL Viewer для impl_CR_1	17

1. Задание

1.1. Цель работы

Разработать устройство, структура которого будет выглядеть следующим образом:

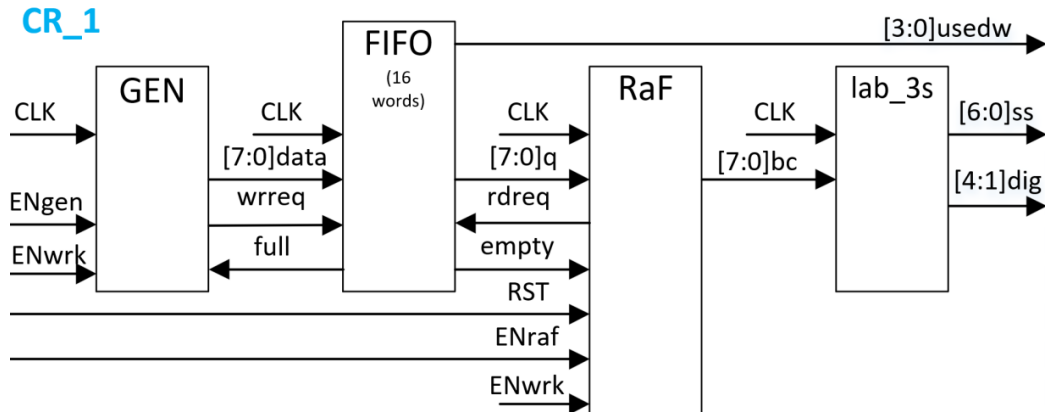


Рис. 1 – Структура разрабатываемого устройства

Он состоит из следующих элементов:

- GEN – генератор случайных чисел.
- FIFO – IP модуль очереди.
- RaF – модуль для поиска максимального числа, среди поданных на вход.
- lab_3s – модуль для вывода результата модуля RaF на 7-сегментный индикатор.

Для этого модуля разработать тестовый файл, программу отладки на плате и итоговый проект для интеграции устройства.

2. Ход решения

2.1. Модуль GEN

2.1.1. Создание модуля

Начнем разработку данного проекта с модуля GEN. Он должен работать по следующему алгоритму:

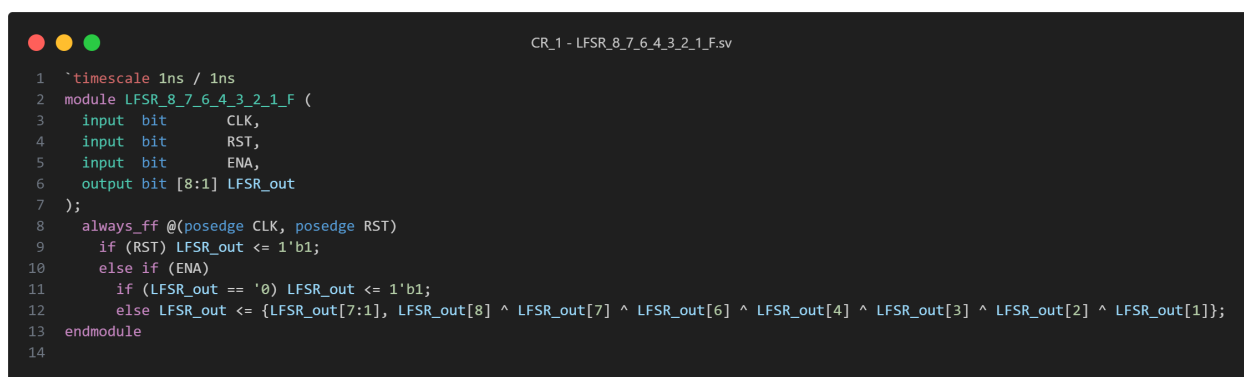
Если ($EN_{gen} = 1$ и $full = 0$ и $Enwrk = 1$) выполняется:

Модуль формирует сигнал $wrreq = 1$, работает генератор псевдослучайных чисел (данные выдаются на выход [7:0] $data$)

Если условие ($EN_{gen} = 1$ и $full = 0$ и $Enwrk = 1$) не выполняется:

Модуль формирует сигнал $wrreq = 0$, генератор псевдослучайных чисел находится в состоянии ожидания (данные выдаются на выход [7:0] $data$)

В качестве генератора псевдослучайных чисел используется модуль LFSR_8_7_6_4_3_2_1_F, разработанный ранее в ходе лабораторных:



```
CR_1 - LFSR_8_7_6_4_3_2_1_F.sv
1  `timescale 1ns / 1ns
2  module LFSR_8_7_6_4_3_2_1_F (
3      input bit    CLK,
4      input bit    RST,
5      input bit    ENA,
6      output bit [8:1] LFSR_out
7  );
8      always_ff @(posedge CLK, posedge RST)
9          if (RST) LFSR_out <= 1'b1;
10         else if (ENA)
11             if (LFSR_out == '0) LFSR_out <= 1'b1;
12             else LFSR_out <= {LFSR_out[7:1], LFSR_out[8] ^ LFSR_out[7] ^ LFSR_out[6] ^ LFSR_out[4] ^ LFSR_out[3] ^ LFSR_out[2] ^ LFSR_out[1]};
13     endmodule
14
```

Рис. 2 – Модуль интерфейса lab_MS_SV5_interface

В свою очередь модуль GEN является лишь контролирующей, когда модуль будет работать, а когда нет. Он будет выглядеть так, как показано на рис. 3 ниже.

Этот модуль принимает на вход сигналы, которые показаны на рис. 1. Далее объявляется модуль LFSR_8_7_6_4_3_2_1_F. Сигнал RST у него будет принимать постоянное значение нуля, т. е. никогда не будет сброшен, благодаря ENA будет выполняться остановка генерации, а на выход $data$ будет подаваться результат.

Логика включения ENA и сигнала $wrreq$ написана в блоке `always_ff` и соответствует поставленному заданию.

```

CR_1 - gen.sv
1  module gen (
2      input bit      clk,
3      input bit      ENgen,
4      input bit      ENwrk,
5      input bit      full,
6      output bit     wrreq,
7      output bit [7:0] data
8  );
9
10     bit      RST = 1'b0;
11     bit      ENA = 1'b0;
12     bit [7:1] LFSR_out = 1'b0;
13
14     LFSR_7_6_3_1_0_F u_LFSR_7_6_3_1_0_F (
15         .CLK(clk),
16         .RST,
17         .ENA,
18         .LFSR_out
19     );
20
21     assign data = {1'b0, LFSR_out};
22
23     always_ff @(posedge clk) begin
24         if (ENgen == 1'b1 && full == 0 && ENwrk == 1) begin
25             wrreq <= 1'b1;
26             ENA <= 1'b1;
27         end else begin
28             wrreq <= 1'b0;
29             ENA <= 1'b0;
30         end
31     end
32
33 endmodule
34

```

Рис. 3 – Модуль gen

Выполним компиляцию и посмотрим на получившуюся RTL схему:

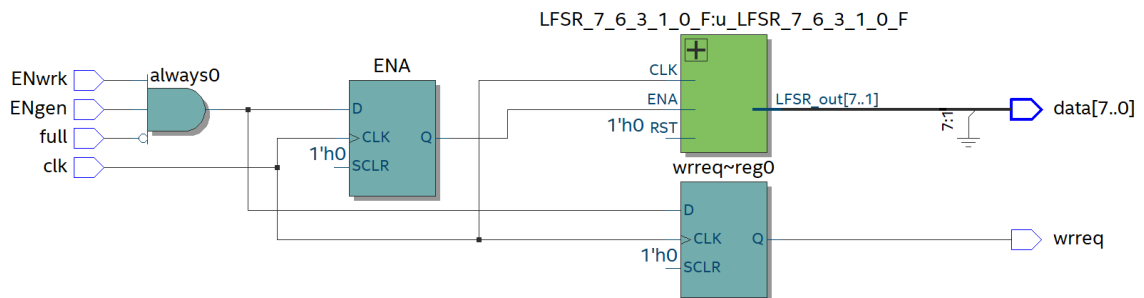


Рис. 4 – RTL Viewer модуля gen

Она соответствует ожиданиям, теперь необходимо провести тестирование данного модуля, для этого разработаем тест первого класса, он будет выглядеть следующим образом:

2.1.2. Тестирование

```
CR_1 - gen_tb.sv
1  `timescale 1ns / 1ns
2  module tb_gen;
3
4      parameter PERIOD = 10;
5
6      bit      clk = 1'b0;
7      bit      ENgen = 1'b0;
8      bit      ENwrk = 1'b0;
9      bit      full = 1'b0;
10     bit      wrreq;
11     bit [7:0] data;
12
13     initial begin
14         forever #(PERIOD / 2) clk = ~clk;
15     end
16
17     gen u_gen (.*);
18
19     initial begin
20         #(PERIOD * 1);
21         ENgen = 1'b1;
22         #(PERIOD * 1);
23         ENwrk = 1'b1;
24         #(PERIOD * 10);
25         full = 1'b1;
26         #(PERIOD * 2);
27         full = 1'b0;
28         #(PERIOD * 4);
29         $stop;
30     end
31
32 endmodule
33
```

Рис. 5 – Тестовый файл первого класса для модуля gen

В этом тесте мы проверяем, что счет происходит только при заданных требованиях, а также продолжается, когда требования вновь выполняются.

Запустим его и посмотрим на результат:

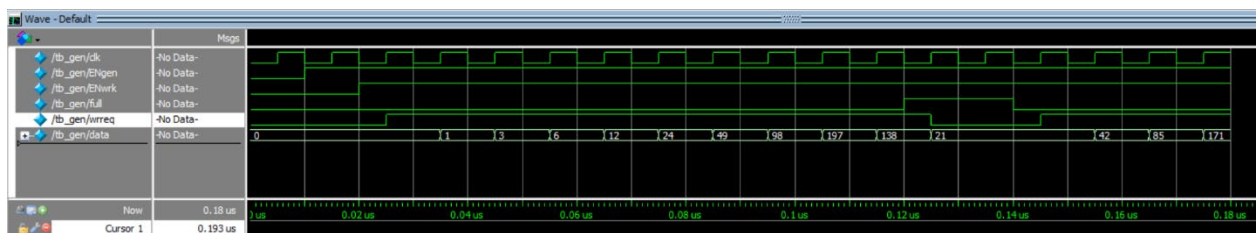


Рис. 6 – Моделирование тестового файла средствами ModelSim (gen_tb)

Как можно заметить, тест соответствует разработанным требованиям.

2.2. Модуль RaF

2.2.1. Создание модуля

Следующим разработаем модуль RaF, который будет считывать данные из FIFO и сохранять максимальное число. Алгоритм выглядит следующим образом:

- Если ($ENraf = 1$ и $empty = 0$ и $Enwrk = 1$)
 - Модуль считывает данные из FIFO (формирует сигнал $rdreq = 1$)
 - Ищет среди принимаемых от FIFO данных максимальное значение
 - Передает текущее максимальное значение на выход $[7:0]bc$
- Если условие ($ENraf = 1$ и $empty = 0$ и $Enwrk = 1$) не выполняется
 - Модуль формирует сигнал $rdreq = 0$
 - Модуль находится в состоянии ожидания (текущее максимальное значение выдается на выход $[7:0]bc$)
- Сигнал RST асинхронно сбрасывает найденное максимальное значение в 0

Его описание на языке System Verilog будет выглядеть следующим образом:

```
CR_1 - RaF.sv

1  module RaF (
2      input bit      CLK,
3      input bit      ENraf,
4      input bit      empty,
5      input bit      Enwrk,
6      input bit      RST,
7      input bit [7:0] q,
8      output bit      rdreq,
9      output bit [7:0] bc
10 );
11
12     always_ff @(posedge CLK, posedge RST) begin
13         if (RST) bc <= 1'b0;
14         else if (ENraf == 1'b1 && empty == 1'b0 && Enwrk == 1'b1) begin
15             rdreq <= 1'b1;
16             bc <= (bc < q ? q : bc);
17         end else begin
18             rdreq <= 1'b0;
19         end
20     end
21
22 endmodule
23
```

Рис. 7 – Модуль RaF

Посмотрим на RTL Viewer разработанного модуля:

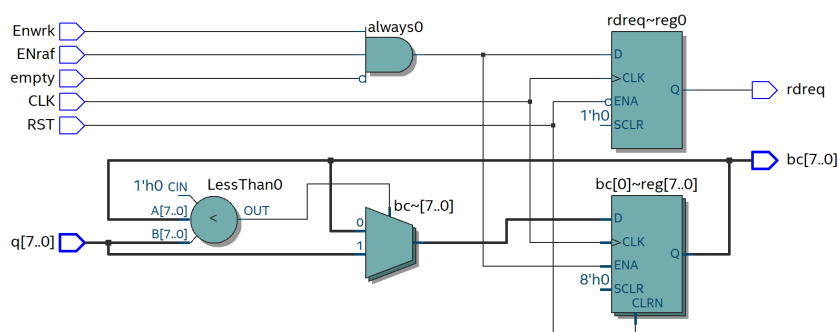


Рис. 8 – RTL Viewer модуля RaF

2.2.2. Тестирование

Далее необходимо провести тестирование исследуемого модуля. Для этого разработаем тест первого уровня:

```
CR_1 - RaF_tb.sv
1 `timescale 1ns / 1ns
2 module tb_RaF;
3
4     parameter PERIOD = 10;
5
6     bit      CLK = 1'b0;
7     bit      ENraf = 1'b0;
8     bit      empty = 1'b0;
9     bit      Enwrk = 1'b0;
10    bit      RST = 1'b0;
11    bit [7:0] q;
12    bit      rdreq;
13    bit [7:0] bc;
14
15    initial begin
16        forever #(PERIOD / 2) CLK = ~CLK;
17    end
18
19    RaF u_RaF (.*);
20
21    initial begin
22        #(PERIOD * 1);
23        q = 8'd7;
24        #(PERIOD * 1);
25        ENraf = 1'b1;
26        #(PERIOD * 1);
27        Enwrk = 1'b1;
28        #(PERIOD * 2);
29        q = 8'd4;
30        #(PERIOD * 2);
31        RST = 1'b1;
32        #(PERIOD * 1);
33        RST = 1'b0;
34        #(PERIOD * 1);
35        q = 8'd120;
36        #(PERIOD * 2);
37        $stop;
38    end
39
40 endmodule
41
```

Рис. 9 – Тестовый файл первого класса для модуля RaF

В этом тесте мы подаем на вход q какое-то значение, но т.к. условия не выполняются записано оно не будет, после чего подаем все значения для выполнения условия и теперь запись должна быть выполнена. После чего пытаемся перезаписать меньшим значением.

Далее происходит сброс и пытаемся записать после сброса другое значение.

Запустим данный тест и посмотрим на результат тестирования:

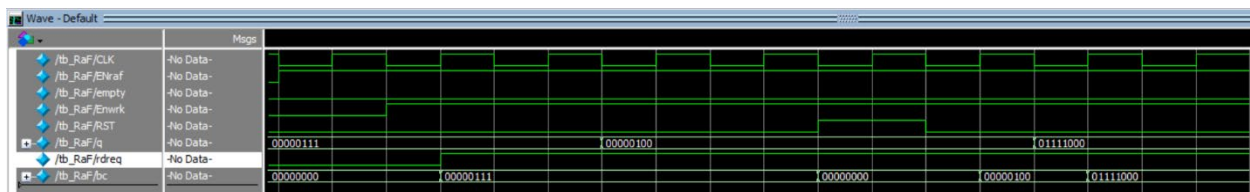


Рис. 10 – Моделирование тестового файла средствами ModelSim (RaF_tb)

Как мы видим все результаты тестирования совпадают с ожиданиями.

Модуль FIFO мы возьмем из IP библиотеки, разрабатывать его не нужно, а модуль для вывода значений на 7-сегментный дисплей мы возьмем из лабораторной работы прошлого семестра, это значит мы готовы перейти к разработке основного модуля, представленном на рис. 1.

2.3. Модуль CR_1

2.3.1. Создание модуля

Данное описание на языке System Verilog приведено ниже:

```
CR_1 - CR_1.sv
1  module CR_1 (
2      input bit      CLK,
3      input bit      ENgen,
4      input bit      ENwrk,
5      input bit      RST,
6      input bit      ENraf,
7      output bit [3:0] usedw,
8      output bit [6:0] ss,
9      output bit [4:1] dig
10 );
11
12 bit      full;
13 bit      wrreq;
14 bit [7:0] data;
15
16 gen u_gen (
17     .clk (CLK),
18     .ENgen,
19     .ENwrk(ENwrk),
20     .full,
21     .wrreq,
22     .data
23 );
24
25 bit      rdreq;
26 bit      empty;
27 bit [7:0] q;
28
29 FIFO u_fifo (
30     .clock(CLK),
31     .data(data),
32     .rdreq(rdreq),
33     .wrreq(wrreq),
34     .empty(empty),
35     .full(full),
36     .q(q),
37     .usedw(usedw)
38 );
39
40 bit [7:0] bc;
41
42 RaF u_RaF (
43     .CLK,
44     .ENraf,
45     .empty,
46     .ENwrk,
47     .RST,
48     .q,
49     .rdreq,
50     .bc
51 );
52
53 lab_3s u_lab_3s (
54     .clk(CLK),
55     .bc(bc),
56     .ss(ss),
57     .dig(dig)
58 );
59
60
61 endmodule
62
```

Рис. 11 – Модуль CR_1

Как можно заметить это просто описание соединения разработанных модулей а также очереди и модуля, лабораторной номер 3.

Очередь будет иметь следующие настройки:

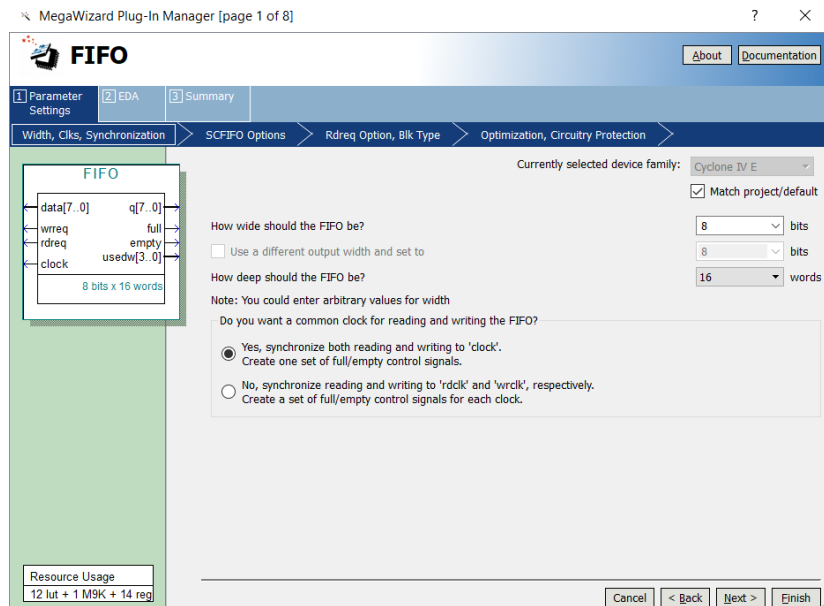


Рис. 12 – Настройка модуля FIFO

Выполним компиляцию и посмотрим на RTL Viewer, чтоб убедиться в корректности построенной схемы:

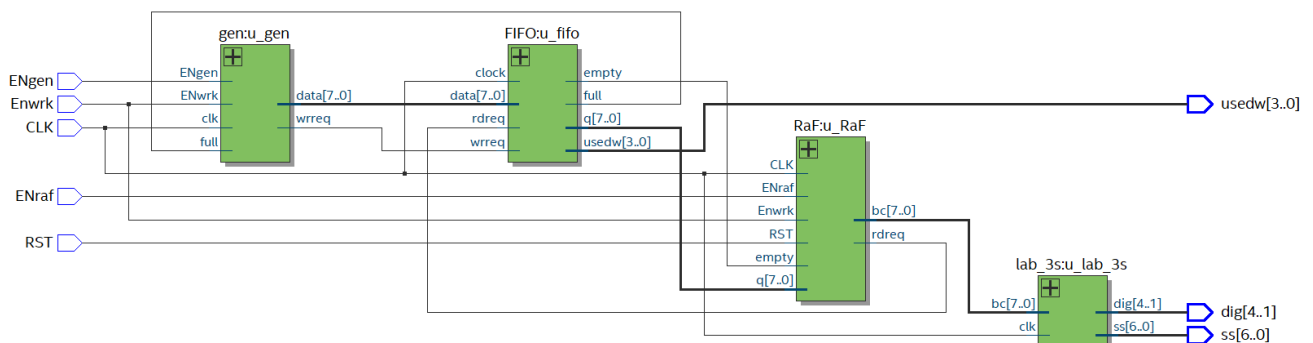


Рис. 13 – RTL Viewer модуля CR_1

Как мы видим, полученная схема похожа на Рис. 2.1. что свидетельствует о корректности разработанного модуля. Теперь выполним тестирование этого модуля, для этого разработает следующий тест первого класса:

2.3.2. Тестирование

```
CR_1 - CR_1_tb.sv
1  `timescale 1ns / 1ns
2  module tb_CR_1;
3
4      parameter PERIOD = 10;
5
6      bit        CLK = 1'b0;
7      bit        ENgen = 1'b0;
8      bit        Enwrk = 1'b0;
9      bit        RST = 1'b0;
10     bit        ENraf = 1'b0;
11     bit [3:0]  usedw;
12     bit [6:0]  ss;
13     bit [4:1]  dig;
14
15     initial begin
16         forever #(PERIOD / 2) CLK = ~CLK;
17     end
18
19     CR_1 u_CR_1 (.*);
20
21     initial begin
22         #(PERIOD * 1);
23         ENraf = 1'b0;
24         ENgen = 1'b1;
25         Enwrk = 1'b1;
26         RST = 1'b0;
27         #(PERIOD * 18);
28         ENraf = 1'b1;
29         ENgen = 1'b0;
30         Enwrk = 1'b1;
31         RST = 1'b0;
32         #(PERIOD * 18);
33         ENraf = 1'b1;
34         ENgen = 1'b1;
35         Enwrk = 1'b1;
36         RST = 1'b0;
37         #(PERIOD * 27);
38         ENraf = 1'b1;
39         ENgen = 1'b1;
40         Enwrk = 1'b0;
41         RST = 1'b0;
42         #(PERIOD * 3);
43         ENraf = 1'b1;
44         ENgen = 1'b1;
45         Enwrk = 1'b1;
46         RST = 1'b1;
47         #(PERIOD * 3);
48         $stop;
49     end
50
51 endmodule
52
```

Рис. 14 - Тестовый файл первого класса для модуля CR_1

Данный тест проверяет работу модуля по следующему алгоритму:

- Запись в FIFO до появления сигнала *full* ($ENraf = 0$ $Engen = 1$ $Enwrk = 1$ $RST = 0$)
- Чтение из FIFO до появления сигнала *empty* ($ENraf = 1$ $Engen = 0$ $Enwrk = 1$ $RST = 0$)
- Чтение и запись FIFO до появления на выводе [7:0]bc максимального значения ($ENraf = 1$ $Engen = 1$ $Enwrk = 1$ $RST = 0$)
- Запрет работы ($ENraf = 1$ $Engen = 1$ $Enwrk = 0$ $RST = 0$)
- Сброс максимального значения ($ENraf = x$ $Engen = x$ $Enwrk = x$ $RST = 1$)

Проведем первый тест, его результат выглядит следующим образом:

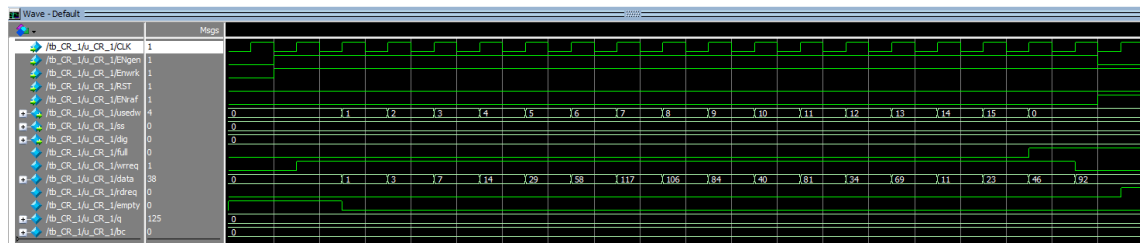


Рис. 15 – Тестирование модуля CR_1 до сигнала full

Как мы видим, все работает корректно и данные выдаются до появления сигнала full. В bc данные не записываются т.к. он не активирован.

Теперь перейдем к следующему тесту:

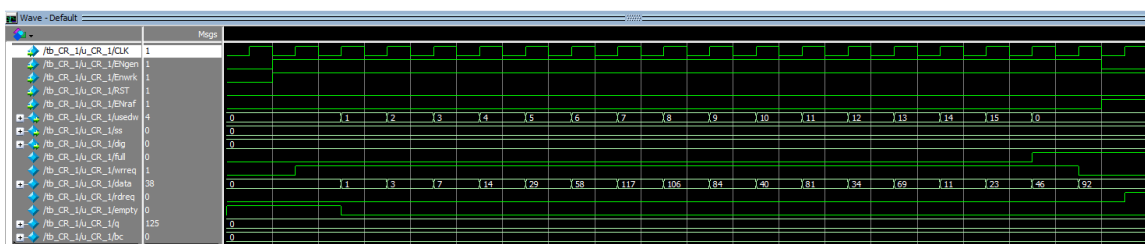


Рис. 16 – Тестирования модуля CR_1 до сигнала empty

Система корректно показывает и на этом тесте, выдавая содержимое очереди до сигнала empty. В bc данные аналогично не попадают т.к. он не активирован.

Теперь запустим систему в стандартном для неё режиме работы:

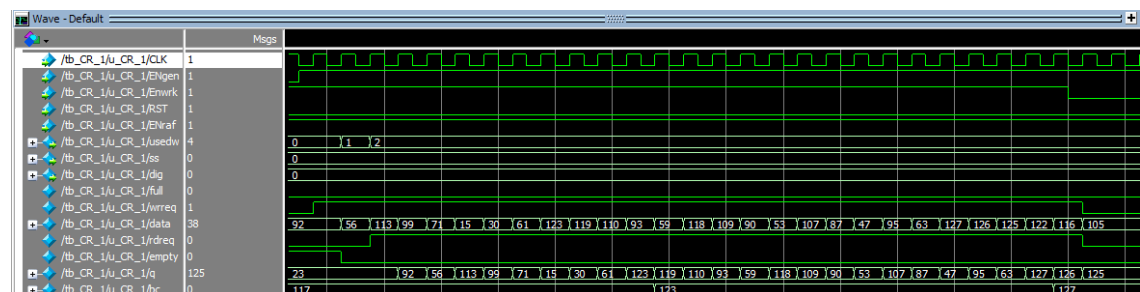


Рис. 17 – Тестирование модуля CR_1 до максимального значения генератора

Как мы видим, система работает корректно и действительно сохраняет значение максимума.

Теперь проверим сигнал RST:

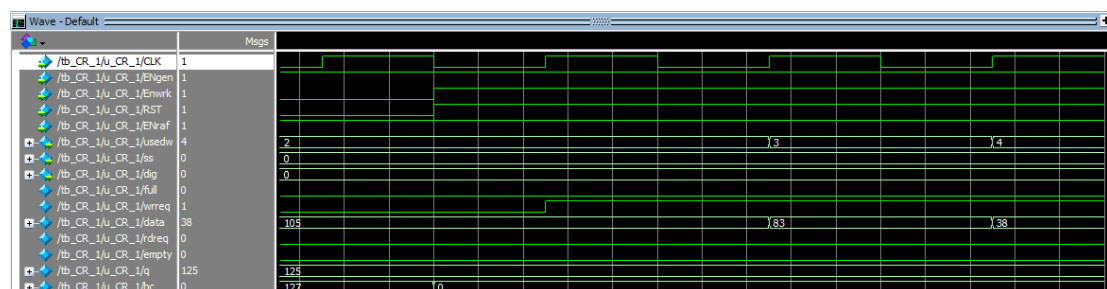


Рис. 18 – Тестирования модуля CR_1 на сигнале RST

На рисунке выше видно, что устройство корректно работает и с этими сигналами.

Стоит отметить, что во время тестирования выходы ss и dig оставались нулевыми. Это связано с тем, что модуле вывода данных на 7-сегментный индикатор стоит делитель, чтоб избавиться от мигания индикатора от частных обновлений. Его значение составляет около 10000, что много меньше количество тактов в тесте.

Изменим это, передав в него значение 1, чтоб при проверке на плате уже видеть нормальные сигналы.

2.4. Создание модуля для тестирования на плате

Перейдем непосредственно к тестированию на плате, для этого создадим следующий модуль, благодаря которому сможем менять входные значение, используя ISSPE и смотреть на результат, используя Signal Tap II:

```
CR_1 - db_CR_1.sv
1 module db_CR_1 (
2     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVC MOS\"", chip_pin = "23" *)
3     input bit CLK
4 );
5     bit      ENgen;
6     bit      Enwrk;
7     bit      RST;
8     bit      ENraf;
9     bit [3:0] usedw;
10    bit [6:0] ss;
11    bit [4:1] dig;
12
13    CR_1 u_CR_1 (
14        .CLK,
15        .ENgen,
16        .Enwrk,
17        .RST,
18        .ENraf,
19        .usedw,
20        .ss,
21        .dig
22    );
23
24    SP_unit u0 (
25        .source ({ENgen, Enwrk, RST, ENraf}), // sources.source
26        .source_clk(CLK) // source_clk.clk
27    );
28
29 endmodule
30
```

Рис. 19 – Модуль db для тестирования на плате

Его RTL Viewer приведен ниже:

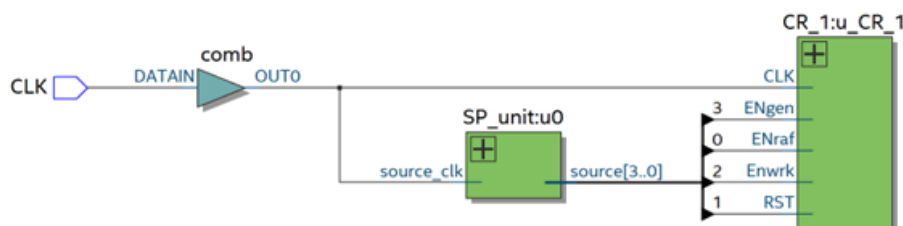


Рис. 20 – RTL Viewer для тестового файла модуля CR_1

2.5. Настройка Signal Tap II

Для ввода значений в модуль будем использовать ISSP, там же будем смотреть результат. Дополнительно добавим Signal Tap II, в котором будем получать значения по изменению состояний и получать результат в виде названия состояния и адреса:

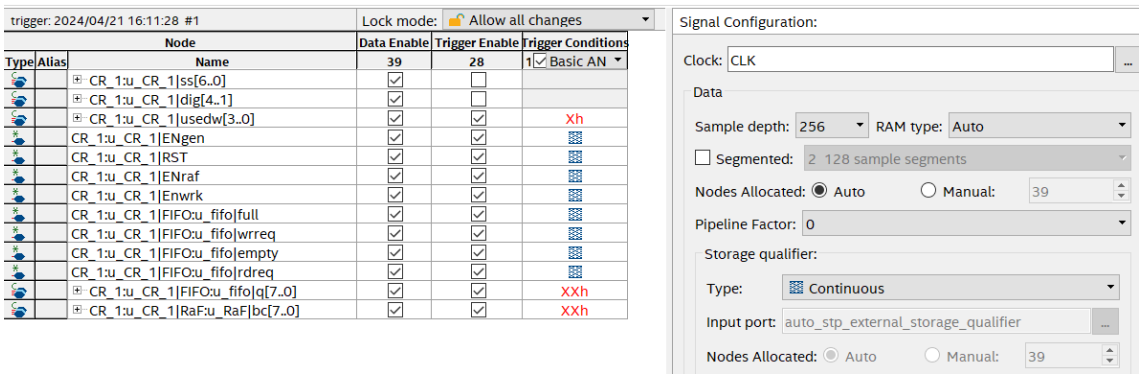


Рис. 21 – Настройка окна Signal Tap II

Для выхода ss создадим маску, чтоб смотреть значение, которое он выводит:

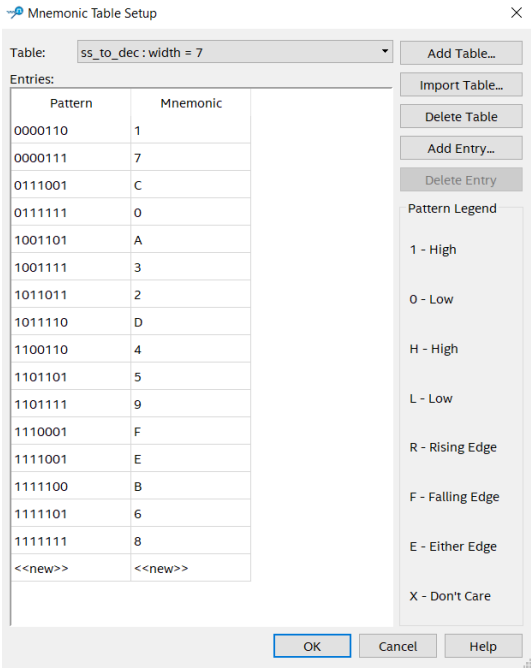


Рис. 22 – Мнемоническая таблица

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

Slow 1200mV OC Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	455.58 MHz	250.0 MHz	CLK	limit due to minimum period restricti

Рис. 23 – Временные характеристики устройства

2.6. Тестирование на плате средствами Signal Tap II

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и рассмотрим интересные нас случаи:

Запись в FIFO до появления сигнала full ($ENraf = 0$ $Engen = 1$ $Enwrk = 1$ $RST = 0$):

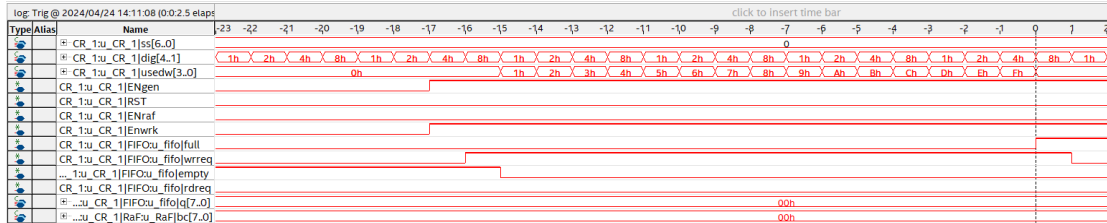


Рис. 24 – Появление сигнала full при тестировании на плате

Чтение из FIFO до появления сигнала empty ($ENraf = 1$ $Engen = 0$ $Enwrk = 1$ $RST = 0$):

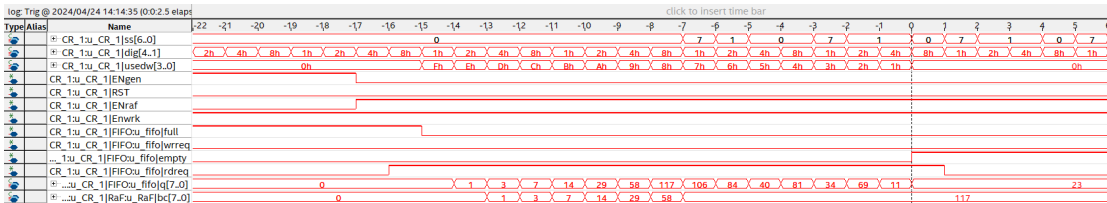


Рис. 25 – Появление сигнала empty при тестировании на плате

Чтение и запись FIFO до появления на выводе $[7:0]bc$ максимального значения ($ENraf = 1$ $Engen = 1$ $Enwrk = 1$ $RST = 0$):

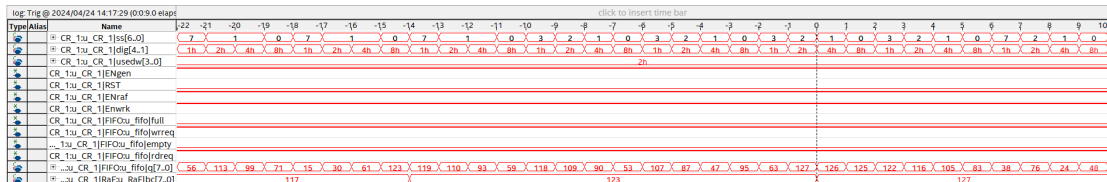


Рис. 26 – Появление значения 127 (максимального) при тестировании на плате

Запрет работы ($ENraf = 1$ $Engen = 1$ $Enwrk = 0$ $RST = 0$):

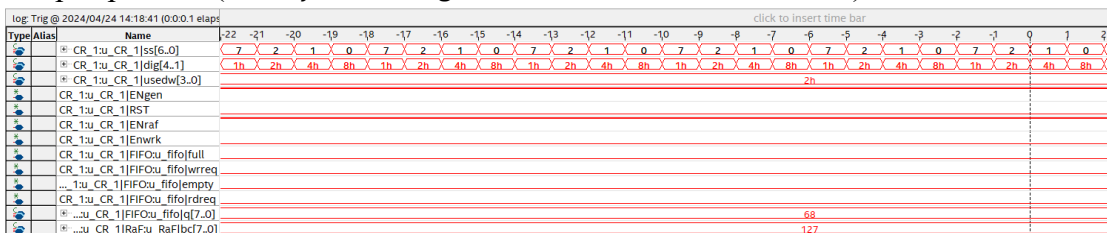


Рис. 27 – Отключение работы при тестировании на плате

Сброс максимального значения ($ENraf = x$ $Engen = x$ $Enwrk = x$ $RST = 1$):

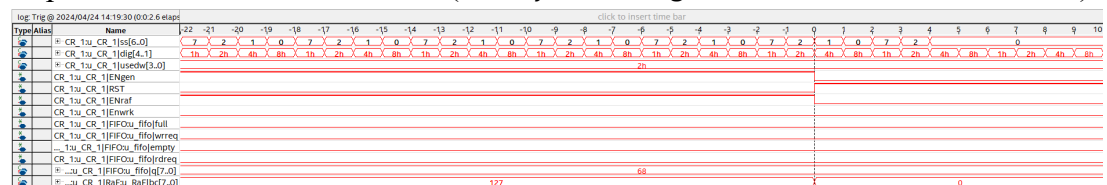


Рис. 28 – Сброс при тестировании на плате

2.7. Создание модуля имплементации и запуск на плате

Как мы видим, все работает корректно. Раз устройство работает верно, перейдем к созданию модуля имплементации:

Как мы видим, все работает корректно. Раз устройство работает верно, перейдем к созданию модуля имплементации:

```

1 module impl_CR_1 (
2     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "23" *)
3     input bit CLK,
4     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "46, 25, 24" *)
5     input bit [2:0] SW,
6     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "84, 76, 85, 77, 86, 133, 87" *)
7     output bit [6:0] ss,
8     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "73, 80, 74, 83" *)
9     output bit [4:1] dig,
10    (* altera_attribute = "-name IO_STANDARD \"2.5-V\"", chip_pin = "69, 70, 71, 72" *)
11    output bit [3:0] LED
12);
13    bit Enwrk = 1'b0;
14
15    CR_1 u_CR_1 (
16        .CLK,
17        .ENgen(SW[1]),
18        .Enwrk,
19        .RST (RST_snh[1]),
20        .ENnraf(SW[2]),
21        .usedw(LED),
22        .ss,
23        .dig
24    );
25
26    bit [1:0] RST_snh = 1'b0;
27
28    always_ff @(posedge CLK) begin
29        RST_snh <= {RST_snh[0], SW[0]};
30    end
31
32    int counter = 0;
33    always_ff @(posedge CLK) begin
34        if (counter > 8000000) begin
35            Enwrk <= 1'b1;
36            counter <= 1'b0;
37        end else begin
38            Enwrk <= 1'b0;
39            counter <= counter + 1;
40        end
41    end
42
43 endmodule
44

```

Рис. 29 – Модуль имплементации

Также не забудем выполнить изменения в модуле вывода значений на 7-сегментный индикатор, поставив делитель обратно на 10000. Выполним компиляцию и посмотрим на получившуюся RTL схему:

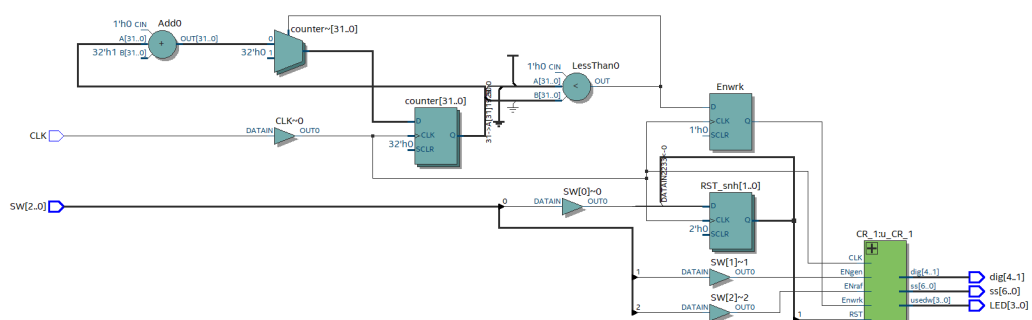


Рис. 30 – RTL Viewer для impl CR 1

Запишем получившуюся программу на плату. Результат работы продемонстрирован преподавателю.

3. Вывод

В результате выполнения курсовой работы была успешно осуществлена разработка заданного устройства с использованием System Verilog в качестве основного инструмента. Кроме того, произведено комплексное тестирование устройства как на симуляторе, так и на плате. Этот процесс охватывал все этапы разработки - от первоначального проектирования до завершения работы над полностью функциональным устройством, готовым к применению. Полученный результат свидетельствует о стабильной работоспособности разработанного устройства, что подтверждает успешное завершение проекта. Эти навыки разработки и тестирования электронных устройств могут быть полезными для будущих проектов в области аппаратной разработки, системной интеграции и производства электроники.