

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ
«МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ»**



**Институт Интеллектуальных Кибернетических Систем
Кафедра «Компьютерные системы и технологии»**

**Отчёт о лабораторной работе
Разработка многофункционального
регистра на языке VHDL**

Студент группы Б20-503 Коломенский В. Г. / _____ /
Руководитель _____ / _____ /

Москва 2023

Оглавление

1. Введение.....	1
2. Структурная схема	2
3. Реализация на VHDL	4
4. Функциональное тестирование.....	12
5. Список литературы и ссылки.....	15

1. Введение

Цель лабораторной работы: изучить технологию проектирования многофункционального регистра (далее – МФР) при помощи языка описания аппаратуры интегральных схем – VHDL.

Задача – разработать МФР на языке VHDL, реализующий микрооперации, заданные вариантом. Перечень микроопераций варианта 106 приведён на рисунке 1.

ВАРИАНТ 106					
Микрооперация					
CLR	EN	Y0	Y1	Y2	
1	X	X	X	X	Асинхронная установка в «0»
0	1	0	0	0	Параллельная загрузка по каналу X
0	1	0	0	1	Логический сдвиг вправо на количество разрядов, определяемое значением суммы битов X0 и Z0
0	1	0	1	0	Арифм. сдвиг влево (обр. код)
0	1	0	1	1	Маскирование содержимого регистра кодом канала X
0	1	1	0	0	Изменение знака числа (обр. код)
0	1	1	0	1	Загрузка поразрядной дизъюнкции канала X и содержимого регистра
0	1	1	1	X	Загрузка суммы удвоенного количества нулей в разрядах 0 и 3 в канале X и удвоенного количества нулей в первой группе справа в канале Z

Рисунок 1. – Микрооперации МФР

2. Структурная схема

На рисунке 2 приведена структурная схема мультифункционального регистра. Описание компонент структурной схемы приведено в таблице 1.

Таблица 1. – Описание компонент МФР

CoZ1 (Count of Zeroes 1)	Если $Y0 = '0'$, то на выходе D0; иначе на выходе удвоенное количество нулей на входах D0 и D1.
CoZ2 (Count of Zeroes 2)	Если $Y0 = '0'$, то на выходе D0; иначе на выходе удвоенное количество в первой группе справа на входе D0-D3.
SM	На выходе $A + B$.
RG	Многофункциональной 4-х разрядный регистр с динамической синхронизацией по фронту (0/1).

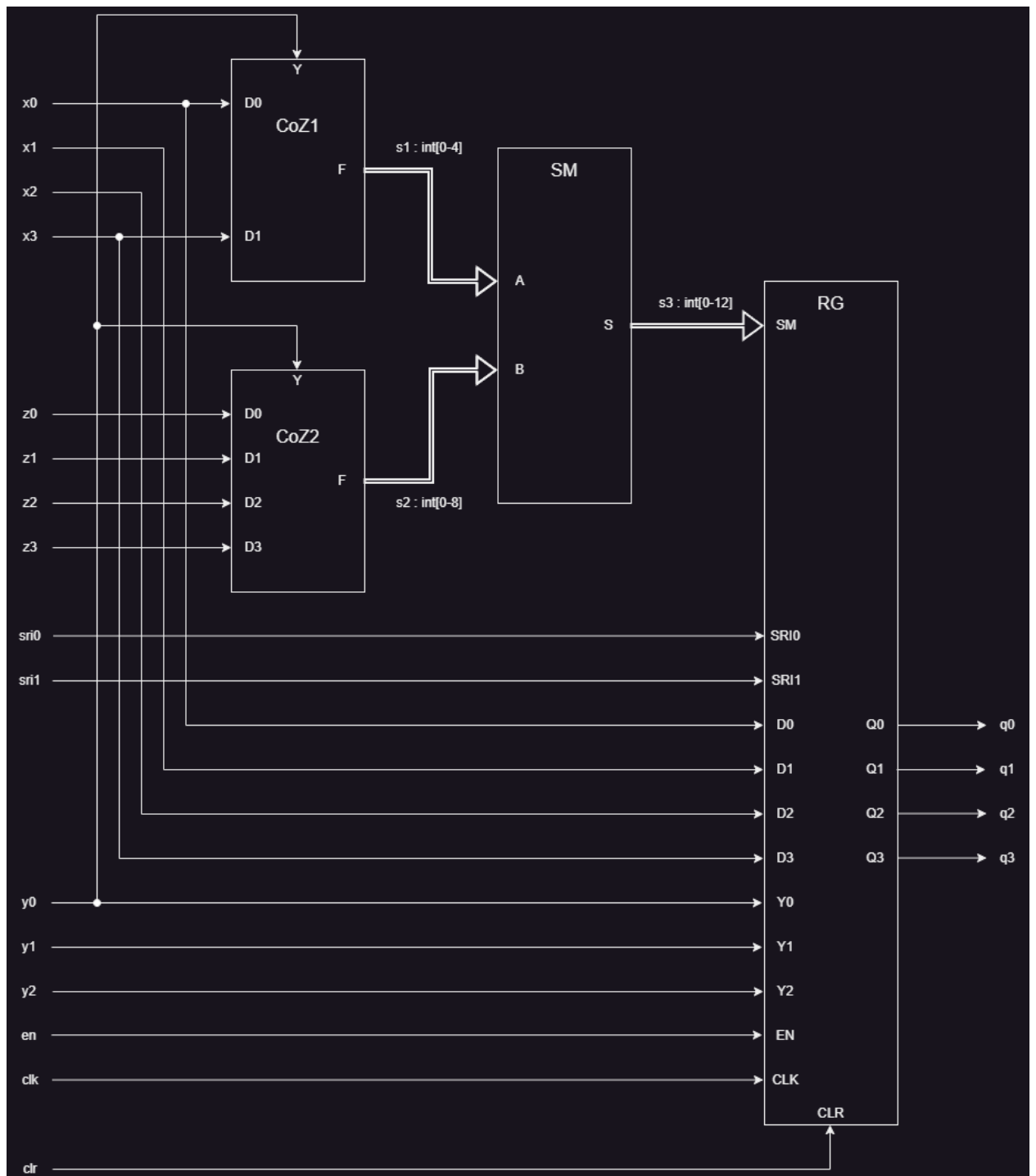


Рисунок 2. – Структурная схема МФР

3. Реализация на VHDL

Реализация МФР на языке VHDL приведена в приложении 1.

Приложение 1. – реализация МФР на VHDL

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4
5 entity MFReg is
6     port (
7         x0: in BIT;
8         x1: in BIT;
9         x2: in BIT;
10        x3: in BIT;
11        z0: in BIT;
12        z1: in BIT;
13        z2: in BIT;
14        z3: in BIT;
15        sri0: in BIT;
16        sri1: in BIT;
17        y0: in BIT;
18        y1: in BIT;
19        y2: in BIT;
20        en: in BIT;
21        clk: in BIT;
22        clr: in BIT;
23        q0: buffer BIT;
24        q1: buffer BIT;
25        q2: buffer BIT;
26        q3: buffer BIT
27    );
28 end MFReg;
29
30
31 architecture MFReg_arch of MFReg is
32
33     signal s1: INTEGER range 0 to 4;
34     signal s2: INTEGER range 0 to 8;
35     signal s3: INTEGER range 0 to 12;
36
37     component CoZ1
```

```

38         port (
39             D0: in BIT;
40             D1: in BIT;
41             Y: in BIT;
42             F: out INTEGER range 0 to 4
43         );
44     end component;
45
46     component CoZ2
47     port (
48         D0: in BIT;
49         D1: in BIT;
50         D2: in BIT;
51         D3: in BIT;
52         Y: in BIT;
53         F: out INTEGER range 0 to 8
54     );
55     end component;
56
57     component SM
58     port (
59         A: in INTEGER range 0 to 4;
60         B: in INTEGER range 0 to 8;
61         S: out INTEGER range 0 to 12
62     );
63     end component;
64
65     component RG
66     port (
67         SM: in INTEGER range 0 to 12;
68         SRI0: in BIT;
69         SRI1: in BIT;
70         D0: in BIT;
71         D1: in BIT;
72         D2: in BIT;
73         D3: in BIT;
74         Y0: in BIT;
75         Y1: in BIT;
76         Y2: in BIT;
77         EN: in BIT;
78         CLK: in BIT;
79         CLR: in BIT;
80         Q0: buffer BIT;
81         Q1: buffer BIT;
82         Q2: buffer BIT;

```

```

83             Q3: buffer BIT
84         );
85     end component;
86
87 begin
88
89     OCoZ1: CoZ1
90         port map (
91             D0 => x0,
92             D1 => x3,
93             Y => y0,
94             F => s1
95         );
96
97     OCoZ2: CoZ2
98         port map (
99             D0 => z0,
100            D1 => z1,
101            D2 => z2,
102            D3 => z3,
103            Y => y0,
104            F => s2
105        );
106
107     OSM: SM
108         port map (
109             A => s1,
110             B => s2,
111             S => s3
112        );
113
114     ORG: RG
115         port map (
116             SM => s3,
117             SRI0 => sri0,
118             SRI1 => sri1,
119             D0 => x0,
120             D1 => x1,
121             D2 => x2,
122             D3 => x3,
123             Y0 => y0,
124             Y1 => y1,
125             Y2 => y2,
126             EN => en,
127             CLK => clk,

```



```

128         CLR => clr,
129         Q0 => q0,
130         Q1 => q1,
131         Q2 => q2,
132         Q3 => q3
133     );
134
135 end MFReg_arch;
136
137
138 entity CoZ1 is
139     port (
140         D0: in BIT;
141         D1: in BIT;
142         Y: in BIT;
143         F: out INTEGER range 0 to 4
144     );
145 end CoZ1;
146
147
148 architecture CoZ1_arch of CoZ1 is
149 begin
150     process (D0, D1, Y)
151         variable D: BIT_VECTOR (1 downto 0);
152     begin
153         if Y='1' then
154             D:= D1 & D0;
155             case D is
156                 when "00" => F<= 4;
157                 when "01" => F<= 2;
158                 when "10" => F<= 2;
159                 when "11" => F<= 0;
160             end case;
161         else
162             case D0 is
163                 when '0' => F <= 0;
164                 when '1' => F <= 1;
165             end case;
166         end if;
167     end process;
168 end CoZ1_arch;
169
170
171 entity CoZ2 is
172     port (

```

```

173         D0: in BIT;
174         D1: in BIT;
175         D2: in BIT;
176         D3: in BIT;
177         Y: in BIT;
178         F: out INTEGER range 0 to 8
179     );
180 end CoZ2;
181
182
183 architecture CoZ2_arch of CoZ2 is
184 begin
185     process (D0, D1, D2, D3, Y)
186         variable D: BIT_VECTOR (3 downto 0);
187     begin
188         if Y='1' then
189             D := D3 & D2 & D1 & D0;
190             case D is
191                 when "0000" => F<= 8;
192                 when "0001" => F<= 6;
193                 when "0010" => F<= 2;
194                 when "0011" => F<= 4;
195                 when "0100" => F<= 4;
196                 when "0101" => F<= 2;
197                 when "0110" => F<= 2;
198                 when "0111" => F<= 2;
199                 when "1000" => F<= 6;
200                 when "1001" => F<= 4;
201                 when "1010" => F<= 2;
202                 when "1011" => F<= 2;
203                 when "1100" => F<= 4;
204                 when "1101" => F<= 2;
205                 when "1110" => F<= 2;
206                 when "1111" => F<= 0;
207             end case;
208         else
209             case D0 is
210                 when '0' => F <= 0;
211                 when '1' => F <= 1;
212             end case;
213         end if;
214     end process;
215 end CoZ2_arch;
216
217

```

```

218 entity SM is
219     port (
220         A: in INTEGER range 0 to 4;
221         B: in INTEGER range 0 to 8;
222         S: out INTEGER range 0 to 12
223     );
224 end SM;
225
226
227 architecture SM_arch of SM is
228 begin
229     S <= A + B;
230 end SM_arch;
231
232
233 entity RG is
234     port (
235         SM: in INTEGER range 0 to 12;
236         SRI0: in BIT;
237         SRI1: in BIT;
238         D0: in BIT;
239         D1: in BIT;
240         D2: in BIT;
241         D3: in BIT;
242         Y0: in BIT;
243         Y1: in BIT;
244         Y2: in BIT;
245         EN: in BIT;
246         CLK: in BIT;
247         CLR: in BIT;
248         Q0: buffer BIT;
249         Q1: buffer BIT;
250         Q2: buffer BIT;
251         Q3: buffer BIT
252     );
253 end RG;
254
255
256 architecture RG_arch of RG is
257 begin
258     process (CLR, EN, CLK)
259         variable Y: BIT_VECTOR (2 downto 0);
260         variable Q: BIT_VECTOR (3 downto 0);
261     begin
262         if CLR='1' then

```

```

263         Q0 <= '0';
264         Q1 <= '0';
265         Q2 <= '0';
266         Q3 <= '0';
267     elsif EN = '0' then
268         null;
269     elsif CLK'event and CLK='1' then
270         Y := Y0 & Y1 & Y2;
271         if Y = "000" then
272             Q0 <= D0;
273             Q1 <= D1;
274             Q2 <= D2;
275             Q3 <= D3;
276         elsif Y = "001" then
277             if SM = 1 then
278                 Q0 <= Q1;
279                 Q1 <= Q2;
280                 Q2 <= Q3;
281                 Q3 <= SRI0;
282             elsif SM = 2 then
283                 Q0 <= Q2;
284                 Q1 <= Q3;
285                 Q2 <= SRI0;
286                 Q3 <= SRI1;
287             end if;
288         elsif Y = "010" then
289             Q0 <= Q3;
290             Q1 <= Q0;
291             Q2 <= Q1;
292         elsif Y = "011" then
293             Q0 <= Q0 and D0;
294             Q1 <= Q1 and D1;
295             Q2 <= Q2 and D2;
296             Q3 <= Q3 and D3;
297         elsif Y = "100" then
298             Q := Q0 & Q1 & Q2 & Q3;
299             if Q = "0000" then
300                 Q0 <= '0';
301                 Q1 <= '0';
302                 Q2 <= '0';
303                 Q3 <= '0';
304             else
305                 Q0 <= not Q0;
306                 Q1 <= not Q1;
307                 Q2 <= not Q2;

```

```

308         Q3 <= not Q3;
309     end if;
310     elsif Y = "101" then
311         Q0 <= Q0 or D0;
312         Q1 <= Q1 or D1;
313         Q2 <= Q2 or D2;
314         Q3 <= Q3 or D3;
315     elsif Y = "110" or Y = "111" then
316         Q0 <= '0';
317         case SM is
318             when 2 | 6 | 10 => Q1 <= '1';
319             when others => Q1 <= '0';
320         end case;
321         case SM is
322             when 4 | 6 | 12 => Q2 <= '1';
323             when others => Q2 <= '0';
324         end case;
325         case SM is
326             when 8 | 10 | 12 => Q3 <= '1';
327             when others => Q3 <= '0';
328         end case;
329     end if;
330 end if;
331 end process;
332 end RG_arch;

```

4. Функциональное тестирование

На рисунке 3.1 и 3.2 представлены тесты, на которых проверялся многофункциональный регистр.

Тестирование МФР

CLR	CE	Y0	Y1	Y2	
0	1	0	0	0	Параллельная загрузка по каналу X

1) 0000 → 1111
2) 1111 → 0000

* 0000 → 1111

CLR	CE	Y0	Y1	Y2	
1	X	X	X	X	Асинхронная установка в «0»

1) 1111 → 0000

* 0000 → 0101

CLR	CE	Y0	Y1	Y2	
0	1	0	0	1	Логический сдвиг вправо на количество разрядов, определяемое значением суммы битов X0 и Z0

1) 0101 → 1010 (SRIO=1)
2) 1010 → 0101
3) 0101 → 1010 (SRIO=1)
4) 1010 → 0101
5) 0101 → 1010 (SRIO=1)
6) 1010 → 0101

(X0=1; Z0=0)

(X0=0; Z0=1)

* 0101 → 1100

7) 1100 → 0011
8) 0011 → 1100 (SRIO=1; SRTI=1)
9) 1100 → 0011

(X0=1; Z0=1)

* 0011 → 0101

CLR	CE	Y0	Y1	Y2	
0	1	0	1	0	Арифм. сдвиг влево (обр. код)

1) 0101 → 0010
2) 0010 → 0100

* 0100 → 1010

3) 1010 → 1101
4) 1101 → 1011

Рисунок 3.1. – Набор тестов

* 1011 → 1111					
CLR	CE	Y0	Y1	Y2	
0	1	0	1	1	Маскирование содержимого регистра кодом канала X
1) 1111 → 1001 (X = 1001)					
2) 1001 → 0000 (X = 0110)					
* 0000 → 0101					
CLR	CE	Y0	Y1	Y2	
0	1	1	0	0	Изменение знака числа (обр. код)
1) 0101 → 1010					
2) 1010 → 0101					
3) 0000 → 0000					
4) 1111 → 0000					
* 0101 → 0000					
CLR	CE	Y0	Y1	Y2	
0	1	1	0	1	Загрузка поразрядной дизъюнкции канала X и содержимого регистра
1) 0000 → 0101 (X = 0101)					
2) 0101 → 1111 (X = 1111)					
CLR	CE	Y0	Y1	Y2	
0	1	1	1	X	Загрузка суммы удвоенного количества нулей в разрядах 0 и 3 в канале X и удвоенного количества нулей в первой группе справа в канале Z
SM	X	Z	Q		
1) 2	0001	1111	1	→	0010 2
2) 4	1110	1101	2	→	0100 4
3) 6	0010	1010	3	→	0110 6
4) 8	1101	0000	4	→	1000 8
5) 10	0000	0001	5	→	1010 10
6) 12	0110	0000	6	→	1100 12
7) 0	1011	1111	0	→	0000 0

Рисунок 3.2. – Набор тестов

Результаты тестирования МФР представлены на рисунках 4.1, 4.2 и 4.3.

5. Список литературы и ссылки

1. Ковригин Б. Н. Введение в инструментальные средства проектирования и отладки цифровых устройств на ПЛИС: Учебно-методическое пособие. М.: МИФИ, 2006.
2. Исходный код МФР на языке VHDL [Электронный ресурс] // <https://github.com/Hypex146/FLMnDA-Lab>.