**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования   
«Санкт-Петербургский государственный университет телекоммуникаций   
им. проф. М. А. Бонч-Бруевича»**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет Инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине:

Вычислительная и микропроцессорная техника

Тема: «Построение конечного автомата Мили»

Группа: ИКТ-801

Громов А.А.

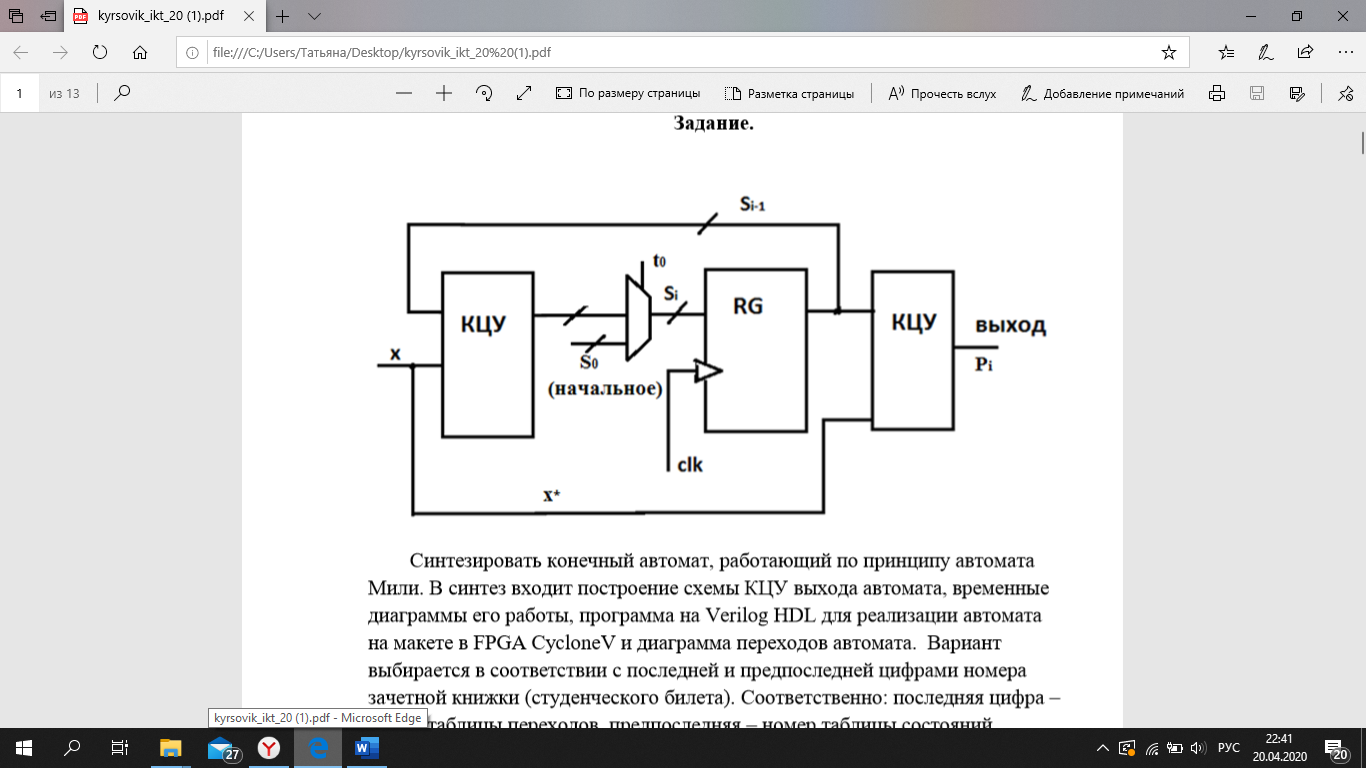
Проверила: Неелова О. Л.

Подпись: \_\_\_\_\_\_\_\_

Санкт-Петербург

2020

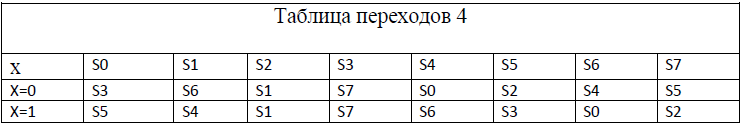
Задание

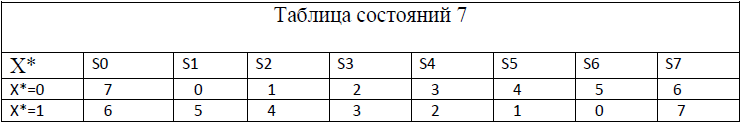


Синтезировать конечный автомат, работающий по принципу автомата Мили. В синтез входит построение схемы КЦУ выхода автомата, временные диаграммы его работы, программа на Verilog HDL для реализации автомата на макете в FPGA CycloneV и диаграмма переходов автомата. Вариант выбирается в соответствии с последней и предпоследней цифрами номера зачетной книжки (студенческого билета). Соответственно: последняя цифра – номер таблицы состояний, предпоследняя – номер таблицы переходов. Сигналы x и x\*определяются комбинацией связанных сигналов «a» и «b». Переходы регулирует «a», состояния – «b».

Выполнение работы

Построим схему автомата Мили, имеющего 3 разряда.





Значит, при x=0 автомат будет совершать переходы:

S0 ->S3->S7->S5->S2->S1->S6->S4->S0->…

А при x=1

S0->S5->S3->S7->S2->S1->S4->S6->S0->…

При этом изменение x\* будет сказываться на выходе автомата следующим образам:

При x\*=0 эквивалентность состояний следующая:

S0==7, S1==0, S2==1, S3==2, S4==3, S5==4, S6==5, S7==6;

При x\*=1:

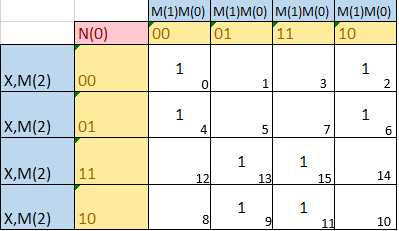
S0==6, S1==5, S2==4, S3==3, S4==2, S5==1, S6==0, S7==7;

Принимая во внимание, что S всего лишь – State (состояние), по этим данным запишем таблицу истинности для второго КЦУ автомата, именуя входы Si - M[2..0], а выходы Pi - N[2..0].

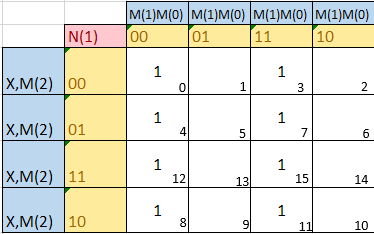
Таблица истинность для второго КЦУ

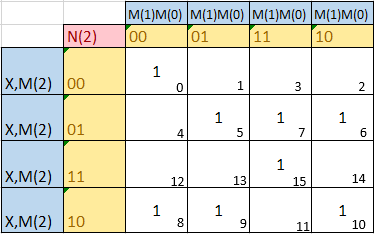
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | M(2) | M(1) | M(0) | N(2) | N(1) | N(0) |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Запишем выражения для выходов, используя карты Карно.



Для N0:   
N0 =

  
Для N1:  
N1 =

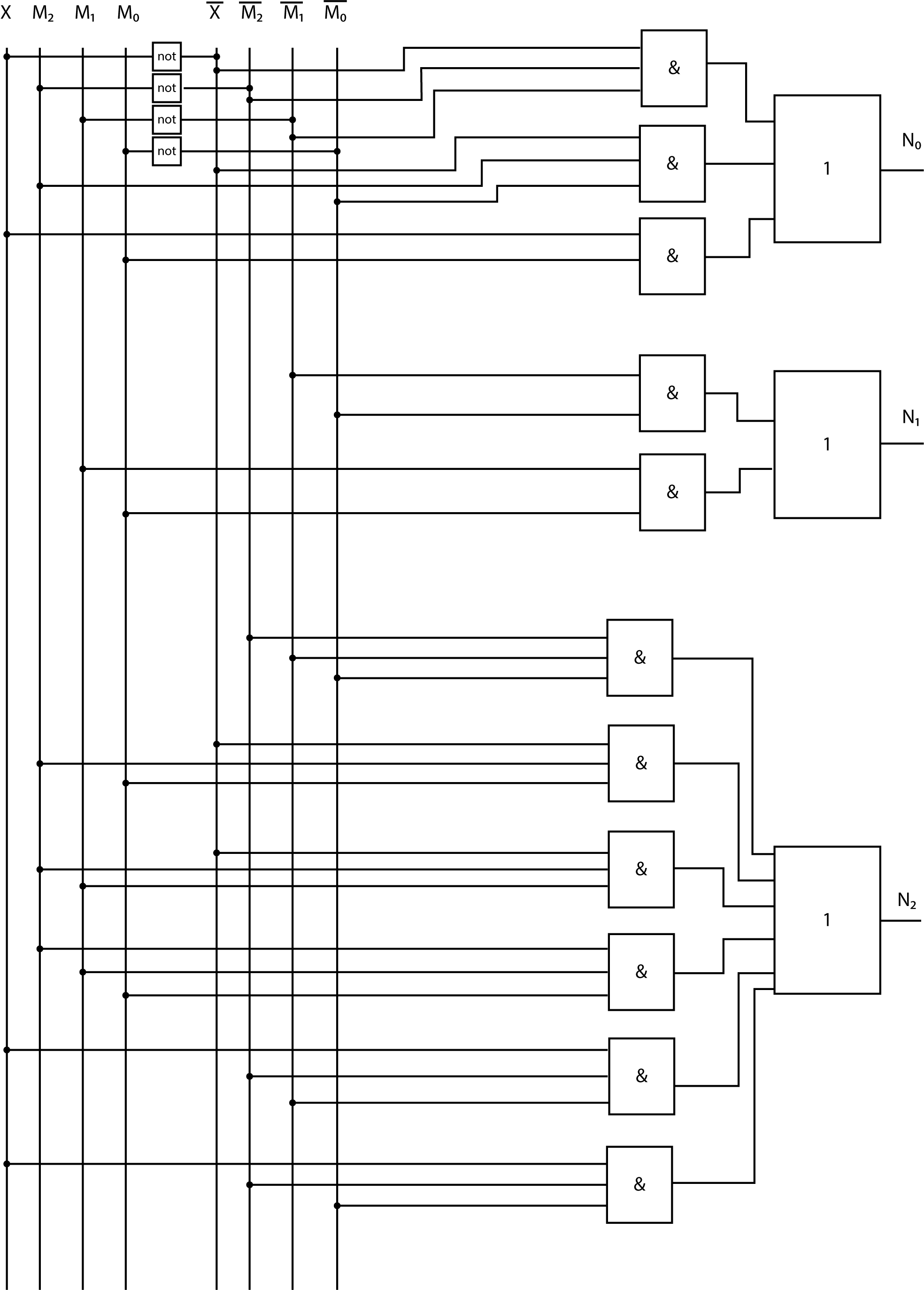


Для N2:

N2 =

На основании выведенных уравнений построим структурную схему второго КЦУ.

Структурная схема КЦУ.



Для построения функциональной диаграммы работы КЦУ записываем программу:

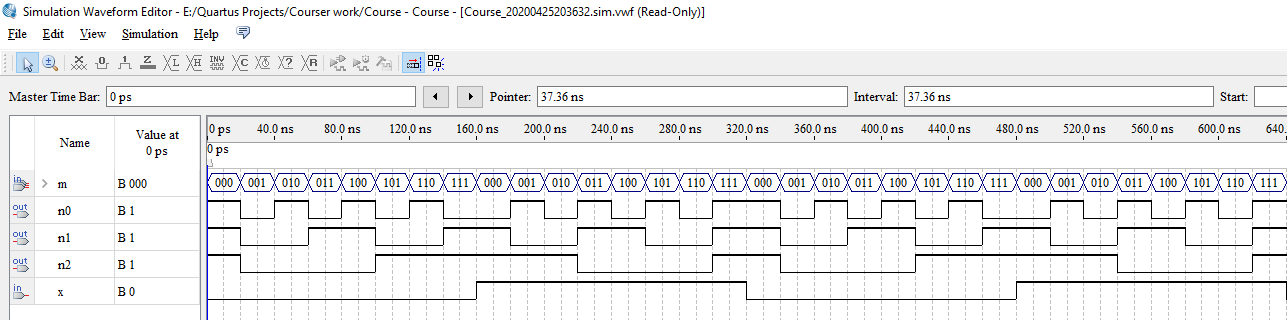
**module** block2

(input x, input [2:0]m, output n0, n1, n2);

assign n0 = ~x & ~m[2] & m[1] | ~x & m[2] & ~m[0] | x & m[0];

assign n1= ~m[1] & ~m[0] | m[1] & m[0] ;

assign n2= ~m[2] & ~m[1] & ~m[0] | ~x & m[2] & m[0] | ~x & m[2] & m[1] | m[2] & m[1] & m[0] | x & ~m[2] & ~m[1] | x & ~m[2] & ~m[0];

**endmodule**

Программа работы устройства будет состоять из двух частей: программирования переходов в следующее состояние на момент подачи импульса синхронизации (первое КЦУ), и программирование соответствия состояния определенному числу (второе КЦУ). Принимаем x==a; x\*==b.

module mealy\_3

(input clk,a, b, reset,

output reg [2:0] out);

reg [2:0] state;

assign in1=a&(~b) | a;

assign in2=b&(~a) | b;

parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3, S4 = 4, S5 = 5,S6 = 6, S7 = 7;

always @ (posedge clk or posedge reset)

begin

if (reset)

state <= S0;

else

case (state)

S0:

if (in1)

begin

state <= S5;

end

else

begin

state <= S3;

end

S1:

if (in1)

begin

state <= S4;

end

else

begin

state <= S6;

end

S2:

if (in1)

begin

state <= S1;

end

else

begin

state <= S1;

end

S3:

if (in1)

begin

state <= S7;

end

else

begin

state <= S7;

end

S4:

if (in1)

begin

state <= S6;

end

else

begin

state <= S0;

end

S5:

if (in1)

begin

state <= S3;

end

else

begin

state <= S2;

end

S6:

if (in1)

begin

state <= S0;

end

else

begin

state <= S4;

end

S7:

if (in1)

begin

state <= S2;

end

else

begin

state <= S5;

end

endcase

end

always @ (state or in2)

begin

case (state)

S0:

if (in2)

begin

out = 3'd6;

end

else

begin

out = 3'd7;

end

S1:

if (in2)

begin

out = 3'd5;

end

else

begin

out = 3'd0;

end

S2:

if (in2)

begin

out = 3'd4;

end

else

begin

out = 3'd1;

end

S3:

if (in2)

begin

out = 3'd3;

end

else

begin

out = 3'd2;

end

S4:

if (in2)

begin

out = 3'd2;

end

else

begin

out = 3'd3;

end

S5:

if (in2)

begin

out = 3'd1;

end

else

begin

out = 3'd4;

end

S6:

if (in2)

begin

out = 3'd0;

end

else

begin

out = 3'd5;

end

S7:

if (in2)

begin

out = 3'd7;

end

else

begin

out = 3'd6;

end

endcase

end

endmodule

