Лабораторная работа №6 по дисциплине

«Архитектура вычислительных систем»

Программное проектирование конечного автомата

Вариант 8

Выполнили:

Студенты 3 курса:

Волосников К.Д.

Добродеев Г.С.

Дунаев В.Е.

Группа:

ИКПИ-11

Преподаватель:

Неелова О.Л.

Оглавление

[1. Цель работы 3](#__RefHeading___Toc1564_1127844969)

[2. Задание 3](#__RefHeading___Toc1566_1127844969)

[3. Блок-схема устройства 4](#__RefHeading___Toc1568_1127844969)

[4. Граф переключений конечного автомата 4](#__RefHeading___Toc1570_1127844969)

[5. Результат работы testbench для модуля crane 6](#__RefHeading___Toc1572_1127844969)

[6. Заключение 6](#__RefHeading___Toc1574_1127844969)

[7. Приложение 6](#__RefHeading___Toc1576_1127844969)

[crane.v 6](#__RefHeading___Toc1578_1127844969)

[count\_div2.v 12](#__RefHeading___Toc1580_1127844969)

[coder.v 12](#__RefHeading___Toc1582_1127844969)

# Цель работы

Создать конечный автомат, управляющий работой подъемного крана. Кран может двигаться вверх и вниз, поворачивать на заданный с тумблеров градус кратный 90, а так же хватать и отцеплять груз.

# Задание

1. Состояние 0 — начальное состояние , соответствующее удержанию сигнала reset
2. Состояние 1 — считывание установленного режима в зависимости от угла поворота
3. Состояния 2 — 4 Ожидание окончания поворота. Возможные угла поворота стрелы и сигнал управления:

0 С — x=2’b00

90 С — x=2’b01

180 С — x=2’b10

-90 С — x=2’b11

1. Время поворота отсчитывается в соответствии с тактами внутреннего таймера, и единица отсчета может выбираться самостоятельно
2. Состояние выходной шины:

После поворота на крайнем правом индикаторе устанавливается и удерживается номер режима согласно сигналу управления x (0, 1, 2, 3)

Слева от режима устанавливаются 2 символа, отображающие движение стрелы или состояние покоя.

Вниз — dn

Ожидание закрепления груза - A1

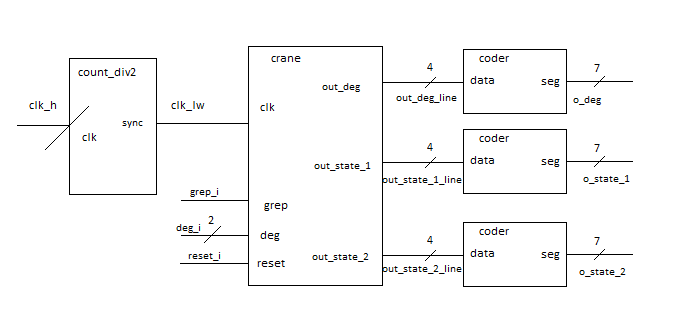
Вверх - up

Ожидание закрепления груза - A2

Возврат - r1

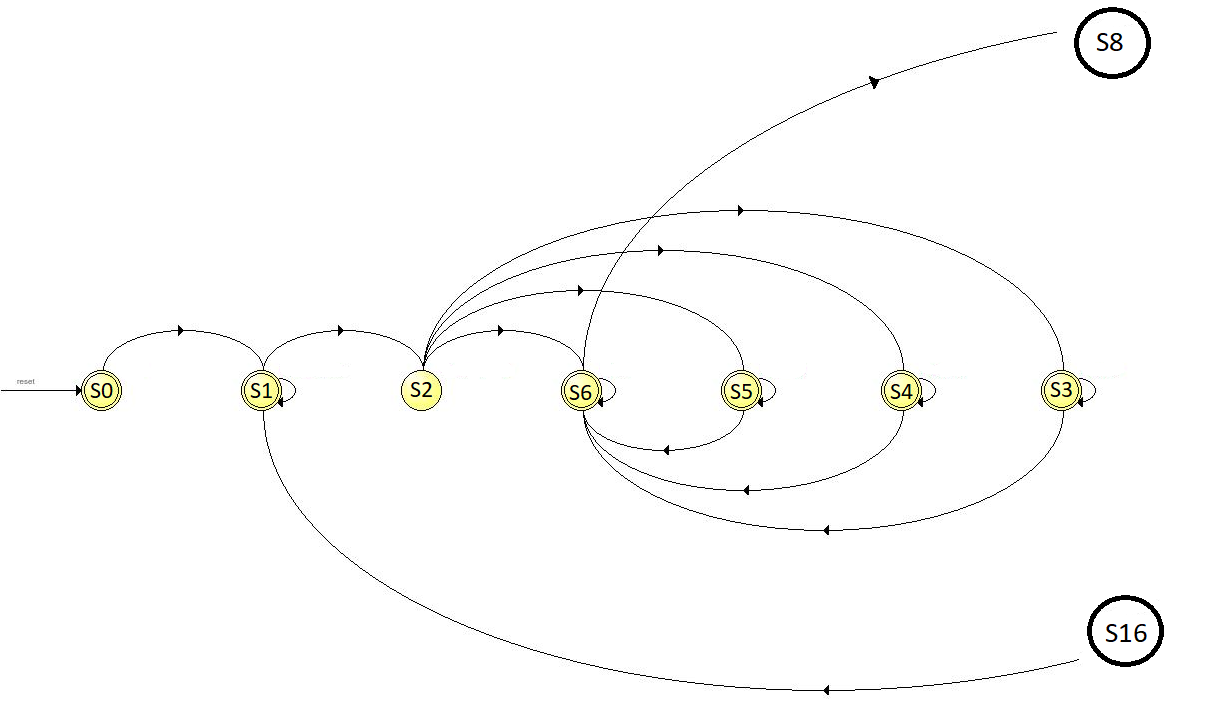
После окончательного возврата стрелы в исходное положение устанавливается режим 00.

3. Блок-схема устройства



*Рисунок 1 – Блок-схема устройства, описанного в результирующем файле.*

# Граф переключений конечного автомата



*Рисунок 2 – Граф переключений конечного автомата*



*Рисунок 3 – Граф переключений конечного автомата*

S0 – сбрасывает кран к начальному

S1 – кран ждет 3 секунды для считывания угла поворота

S2 – кран считывает угол с тумблеров и либо начинает опускаться либо поворачивает на необходимый угол, при этом он сохраняет угол поворота в своем регистре

S3 – кран поворачивает на 90 градусов за 3 секунды

S4 – кран поворачивает на 180 градусов за 6 секунд

S5 – кран поворачивает на -90 градусов за 3 секунды

S6 – кран 30 секунд опускается вниз

S8 – в этом состоянии кран ждет уведомления о зацеплении груза

S9 – если команда поступила, то поднимаем груз наверх, иначе возвращаемся к S8

S7 – кран 30 секунд поднимается наверх вместе с грузом

S10 – в этом состоянии кран ждет уведомления об отцеплении груза

S11- если команда поступила, то возвращаем кран в исходное положение, иначе возвращаемся к S10

S12 – состояние, в котором кран в зависимости от сохраненного угла поворота выбирает обратный угол, чтобы вернутся в начальное состояние

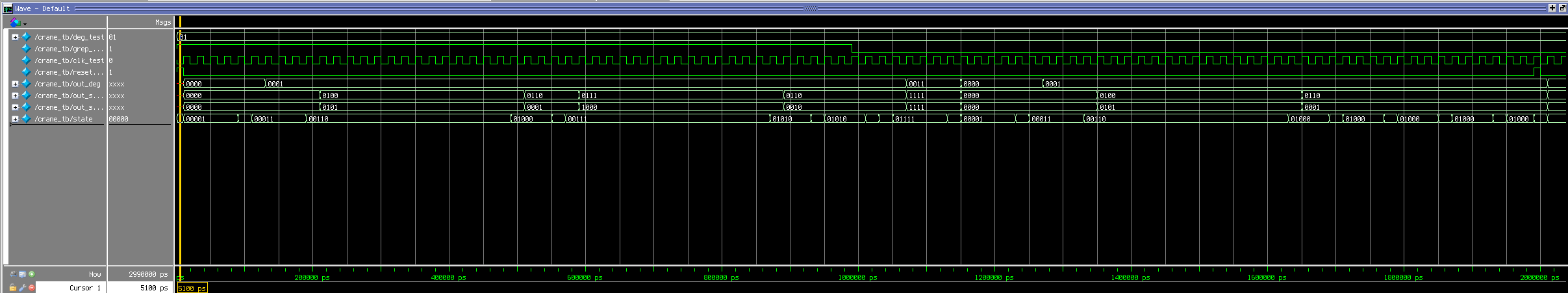
S13 – кран поворачивает на 90 градусов за 3 секунды

S14 – кран поворачивает на 180 градусов за 6 секунд

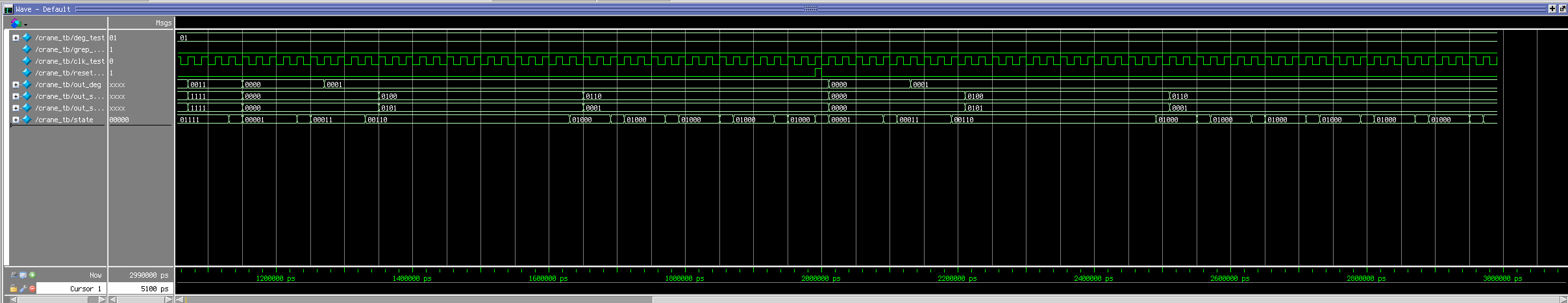
S15 – кран поворачивает на -90 градусов за 3 секунды

S16 – конечное состояние, возвращает кран к первому состоянию для повтора работы

# Результат работы testbench для модуля crane



*Рисунок 4 – Результат работы testbench для модуля crane*



*Рисунок 5 – Результат работы testbench для модуля crane*

# Заключение

В ходе лабораторной работы был спроектирован автомат имитирующий подъемный кран. Функционирование устройства было поблочно проверено с помощью пакета Modelsim. Устройство реализовано на макете DE1-SoC с использованием пакета Quartus 15.

# Приложение

## **crane.v**

module crane

(input clk, reset, grep,

input [1:0] deg,

output reg [3:0] out\_deg,

output reg [3:0] out\_state\_1,

output reg [3:0] out\_state\_2);

reg [5:0] state;

reg [3:0] cnt;

reg [1:0] angle;

parameter Res = 0, WaitDegree = 1, ReadDegree = 2, Turn90 = 3, Turn180 = 4, Turn\_90 = 5, Down = 6, Up = 7, WaitGrep = 8, Grep = 9, WaitDrop = 10, Drop = 11, Return = 12, SecondTurn90 = 13, SecondTurn180 = 14, SecondTurn\_90 = 15, Final = 16;

always @(posedge clk or posedge reset)

begin

if (reset)

state <= Res;

else

begin

case (state)

Res:

state <= WaitDegree;

WaitDegree:

if(cnt == 4'd3)

state <= ReadDegree;

ReadDegree:

case(deg)

2'b00 :

begin

state <= Down;

end

2'b01 :

begin

state <= Turn90;

end

2'b10 :

begin

state <= Turn180;

end

2'b11 :

begin

state <= Turn\_90;

end

endcase

Turn90:

if(cnt == 4'd3)

state <= Down;

Turn180:

if(cnt == 4'd6)

state <= Down;

Turn\_90:

if(cnt == 4'd3)

state <= Down;

Down:

if(cnt == 4'd30)

state <= WaitGrep;

WaitGrep:

if(cnt == 4'd2)

state <= Grep;

Grep:

if(!grep)

state <= WaitGrep;

else

state <= Up;

Up:

if(cnt == 4'd30)

state <= WaitDrop;

WaitDrop:

if(cnt == 4'd2)

state <= Drop;

Drop:

if(!grep)

state <= Return;

else

state <= WaitDrop;

Return:

case(angle)

2'b00 :

begin

state <= Final;

end

2'b01 :

begin

state <= SecondTurn\_90;

end

2'b10 :

begin

state <= SecondTurn180;

end

2'b11 :

begin

state <= SecondTurn90;

end

endcase

SecondTurn90:

if(cnt == 4'd3)

state <= Final;

SecondTurn180:

if(cnt == 4'd6)

state <= Final;

SecondTurn\_90:

if(cnt == 4'd3)

state <= Final;

Final:

state <= WaitDegree;

default:

state <= Res;

endcase

end

end

always @ (posedge clk)

begin

case (state)

Res:

begin

angle <= 4'd0;

cnt <= 4'd0;

out\_deg <= 4'd0;

out\_state\_1 <= 4'd0;

out\_state\_2 <= 4'd0;

end

WaitDegree:

begin

if(cnt == 4'd3)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'd0;//0

end

Turn90:

begin

angle <= 2'd1;

if(cnt == 4'd3)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0001;//1

end

Turn180:

begin

angle <= 2'd2;

if(cnt == 4'd6)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0010;//2

end

Turn\_90:

begin

angle <= 2'd3;

if(cnt == 4'd3)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0011;//3

end

Down:

begin

if(cnt == 4'd30)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_state\_1 <= 4'd4;// d

out\_state\_2 <= 4'd5;// n

//out\_deg <= 4'b1111;// void

end

WaitGrep:

begin

if(cnt == 4'd2)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_state\_1 <= 4'd6;// A

out\_state\_2 <= 4'd1;// 1

end

Up:

begin

if(cnt == 4'd30)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_state\_1 <= 4'd7;// u

out\_state\_2 <= 4'd8;// p

end

WaitDrop:

begin

if(cnt == 4'd2)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_state\_1 <= 4'd6;// A

out\_state\_2 <= 4'd2;// 2

end

Return:

begin

out\_state\_1 <= 4'd6;// r

out\_state\_2 <= 4'd2;// 1

end

SecondTurn90:

begin

if(cnt == 4'd3)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0001;// 1

out\_state\_1 <= 4'b1111;

out\_state\_2 <= 4'b1111;

end

SecondTurn180:

begin

if(cnt == 4'd6)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0010;// 2

out\_state\_1 <= 4'b1111;

out\_state\_2 <= 4'b1111;

end

SecondTurn\_90:

begin

if(cnt == 4'd3)

cnt <= 4'd0;

else

cnt <= cnt + 4'd1;

out\_deg <= 4'b0011;// 3

out\_state\_1 <= 4'b1111;

out\_state\_2 <= 4'b1111;

end

Final:

begin

out\_deg <= 4'd0;//void

out\_state\_1 <= 4'd0;// 0

out\_state\_2 <= 4'd0;// 0

end

endcase

end

endmodule

## **count\_div2.v**

module count\_div2

#(parameter N=29,M=29'd50000000)

//#(parameter N=4, M=4'd10)

(input wire clk,

output reg sync=0,

output wire [N-1:0]q);

reg [N-1:0]cnt=0;

wire [N-1:0]cnt\_next;

assign q=cnt;

assign cnt\_next=cnt+1'b1;

always@(posedge clk)

begin

cnt<=cnt\_next;

sync<=sync;

if (cnt==M-1)

begin

cnt<=0;

sync<=~sync;

end

end

endmodule

## **coder.v**

module coder

(input wire [3:0] data,

output wire [6:0] seg);

reg [6:0]code;

assign seg=code;

always @\*

case(data)

4'b0000:code = 7'b1000000;// 0

4'b0001:code = 7'b1111001;// 1

4'b0010:code = 7'b0100100;// 2

4'b0011:code = 7'b0110000;// 3

4'b0100:code = 7'b0100001;// d

4'b0101:code = 7'b0101011;// n

4'b0110:code = 7'b0001000;// A

4'b0111:code = 7'b1100011;// u

4'b1000:code = 7'b0001100;// p

4'b1001:code = 7'b0101111;// r

4'b1010:code = 7'b0001000;//A

4'b1011:code = 7'b0000011;//b

4'b1100:code = 7'b1000110;//C

4'b1101:code = 7'b0100001;//D

4'b1110:code = 7'b0000110;//E

4'b1111:code = 7'b1111111;// void

endcase

endmodule