Математические модели и методы логического синтеза СБИС

Осень 2015

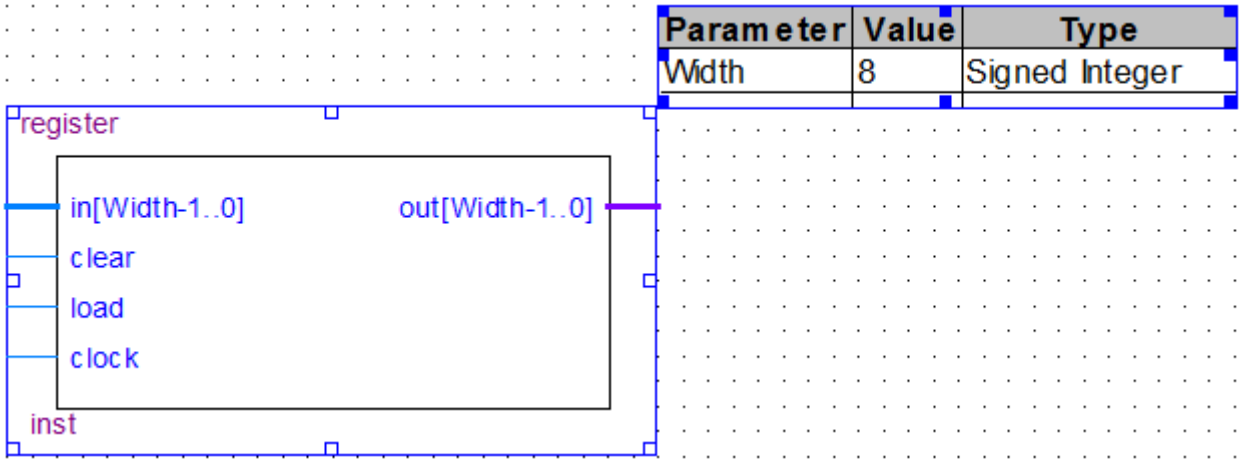
План семинара

- Понятие об операционном и управляющем автоматах
- Построение указанных автоматов по описанию алгоритма
- Использование элементов памяти
- Лабораторная работа – аппаратная реализация элементарного алгоритма.

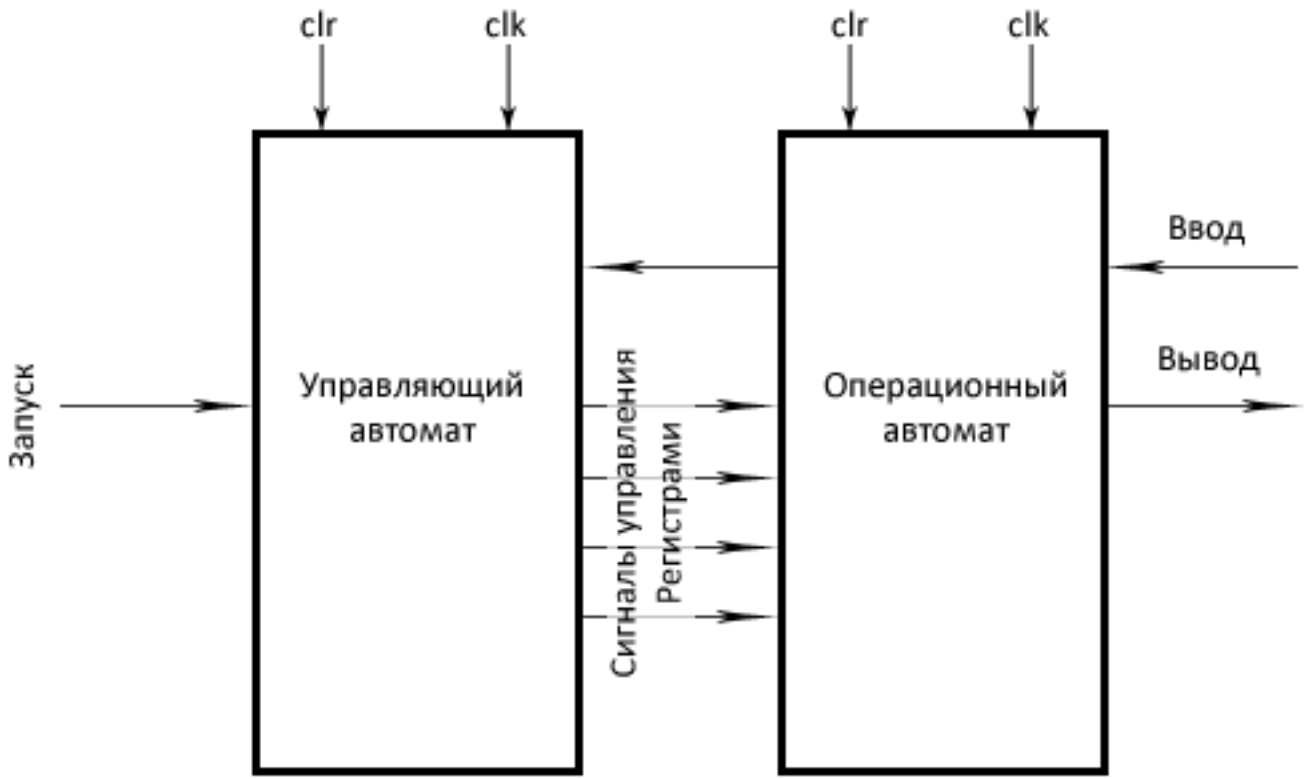
Регистры

```
module register
        #(parameter              Width = 8)
        (output reg      [Width-1:0] out,
        input            [Width-1:0] in,
        input                        clear, load, clock);

        always @(posedge clock)
                if (~clear)
                        out <= 0;
                else if (~load)
                        out <= in;
endmodule
```



Операционный и управляющий автоматы

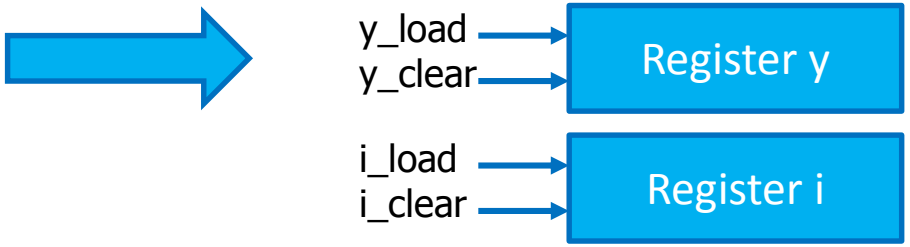


Пример алгоритма

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

Определение регистров

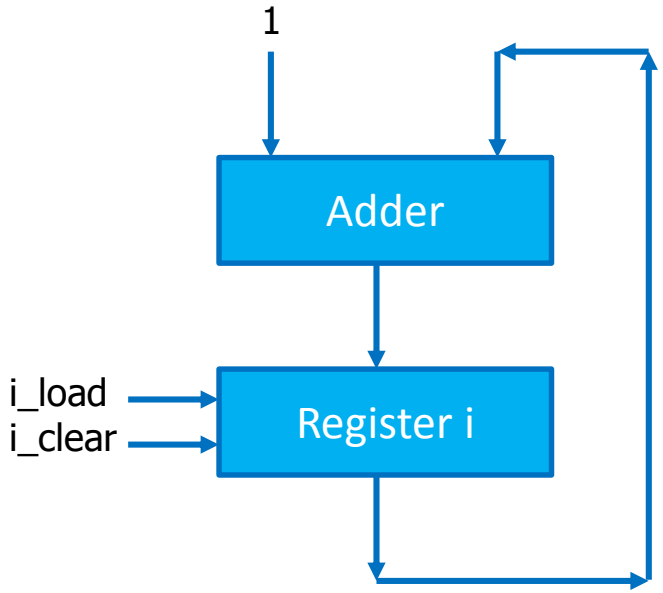
```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



- Каждая переменная порождает свой регистр
- Возможно, что выбранная реализация требует дополнительные (служебные) регистры
- x_load = 0 – загрузка нового значения в регистр
- x_clear = 0 – сброс регистра в значение «0»

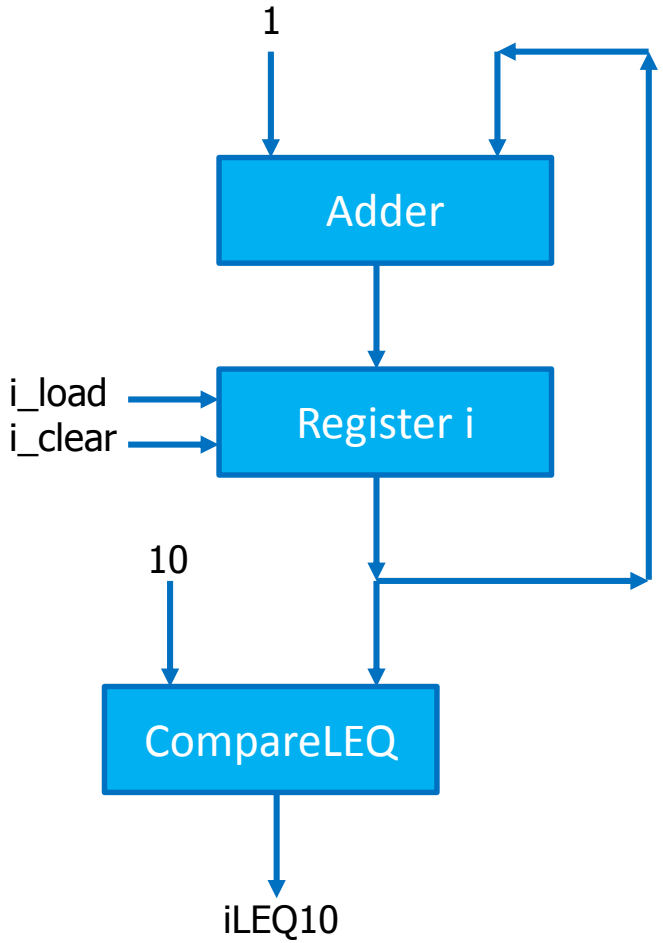
Построение операционного автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



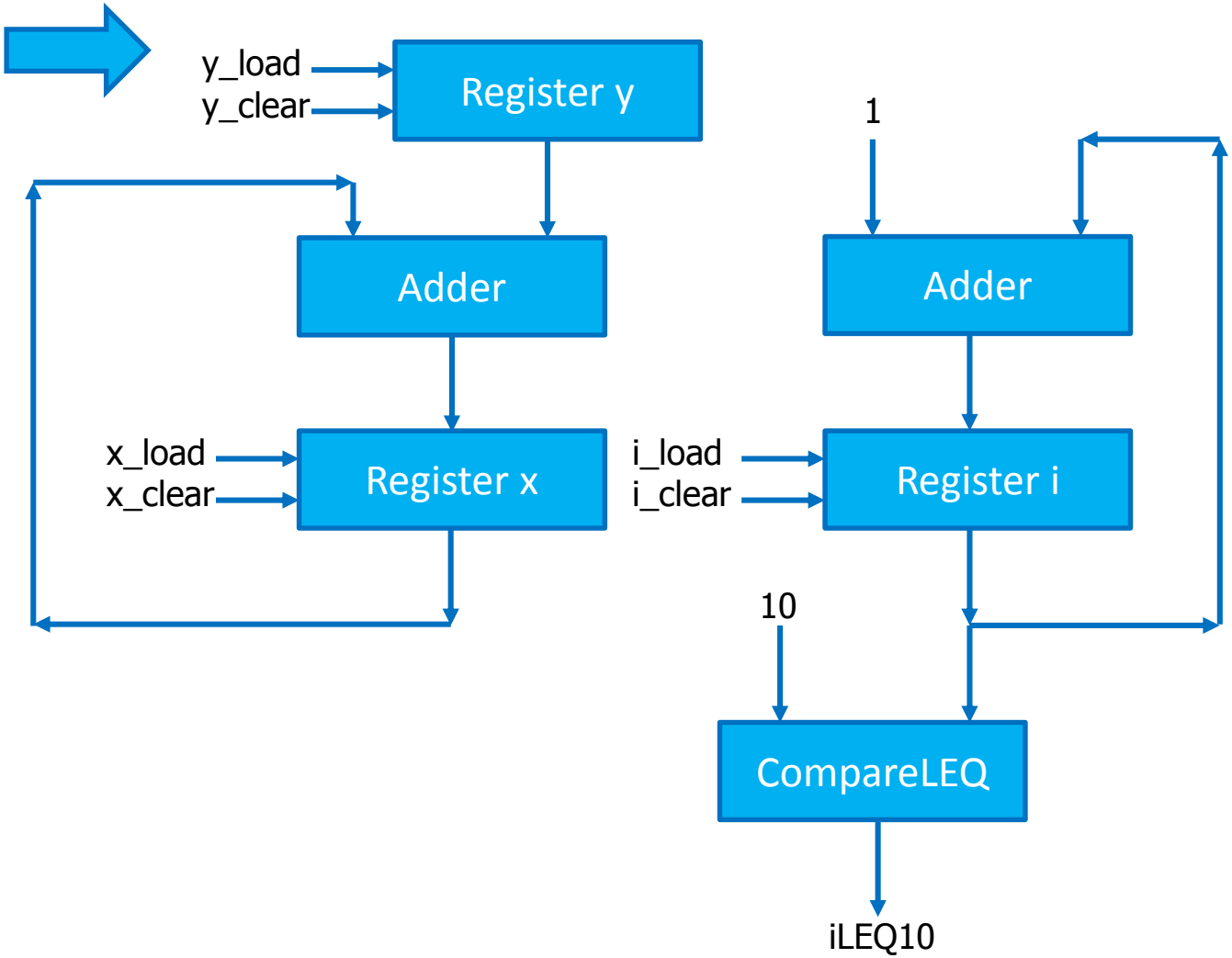
Построение операционного автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



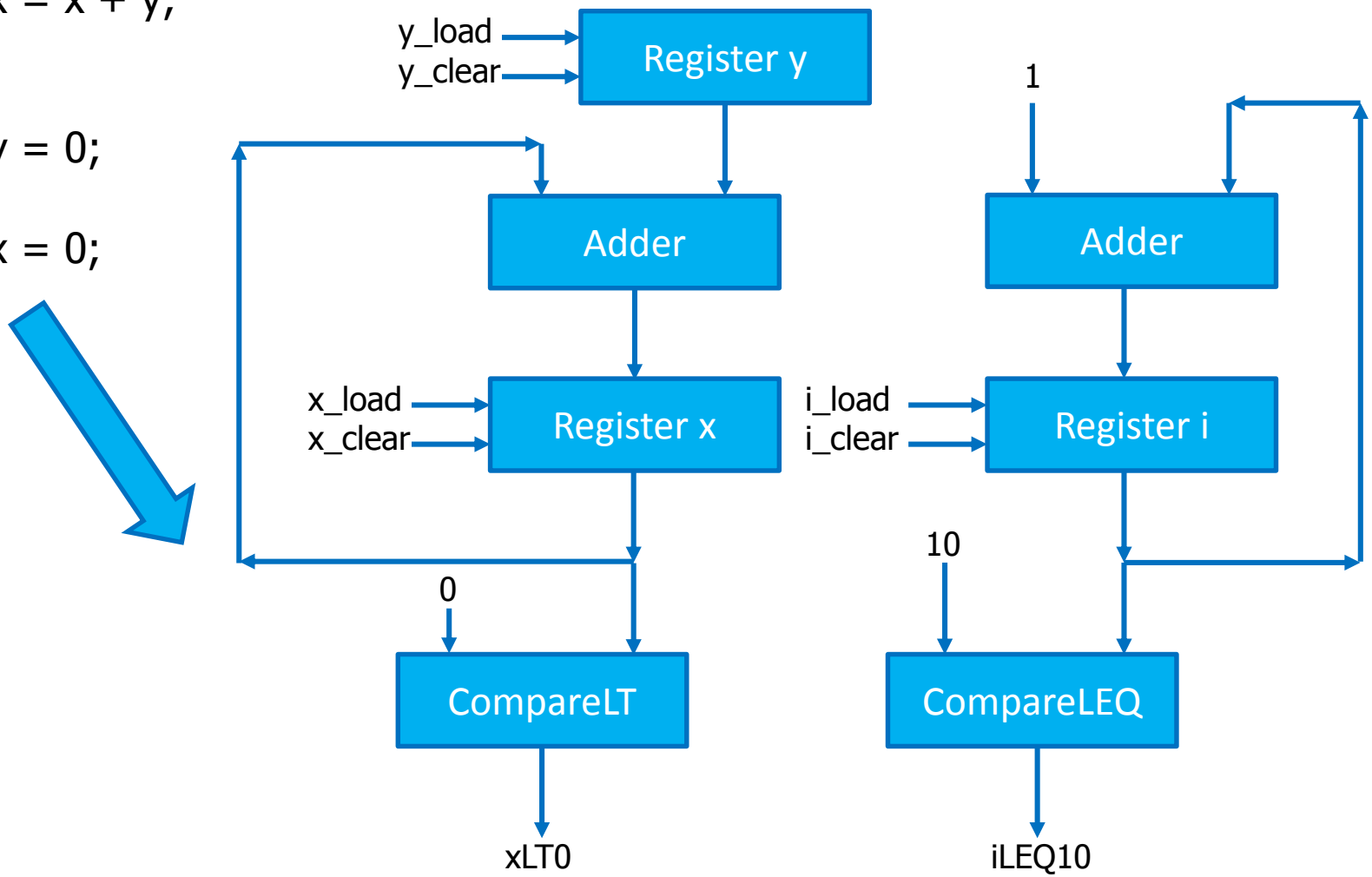
Построение операционного автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



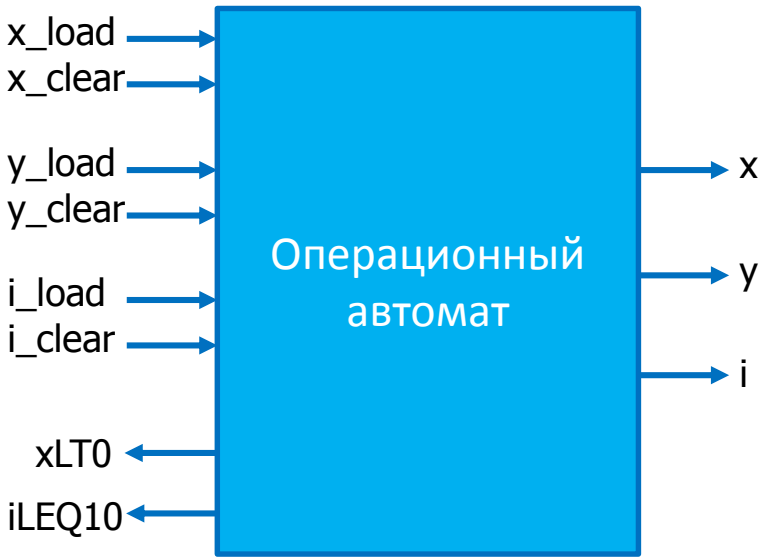
Построение операционного автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



Построение операционного автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



Построение управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

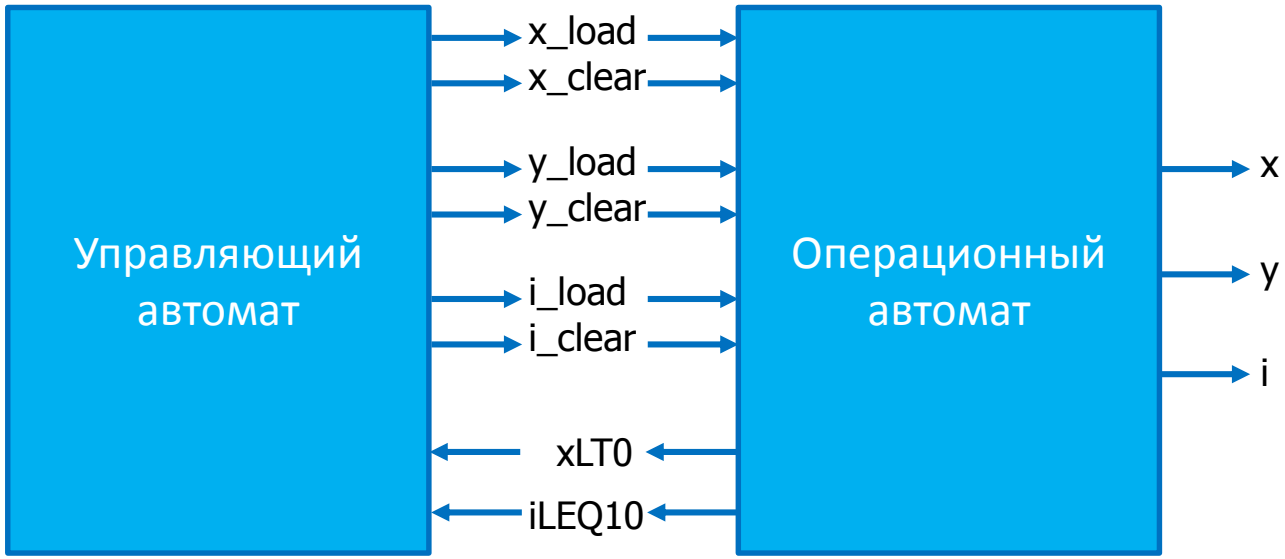


Диаграмма Мура управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

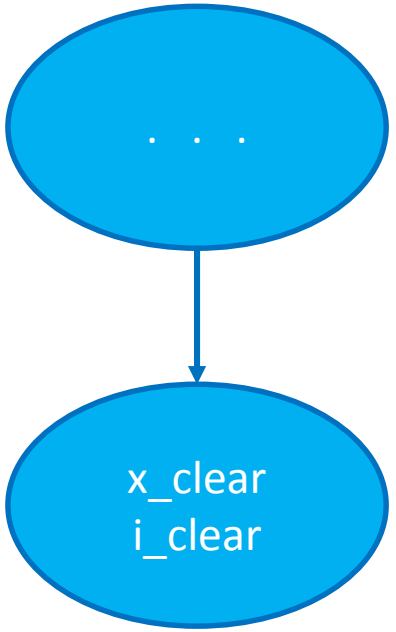


Диаграмма Мура управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

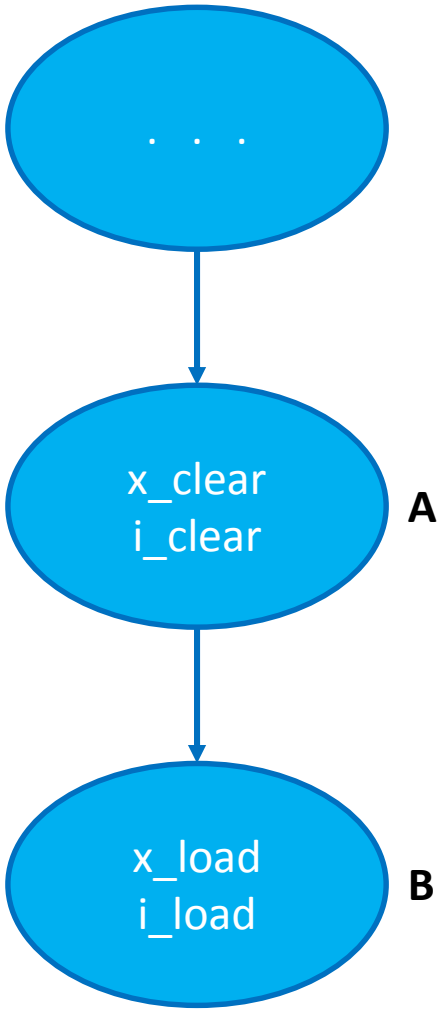


Диаграмма Мура управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

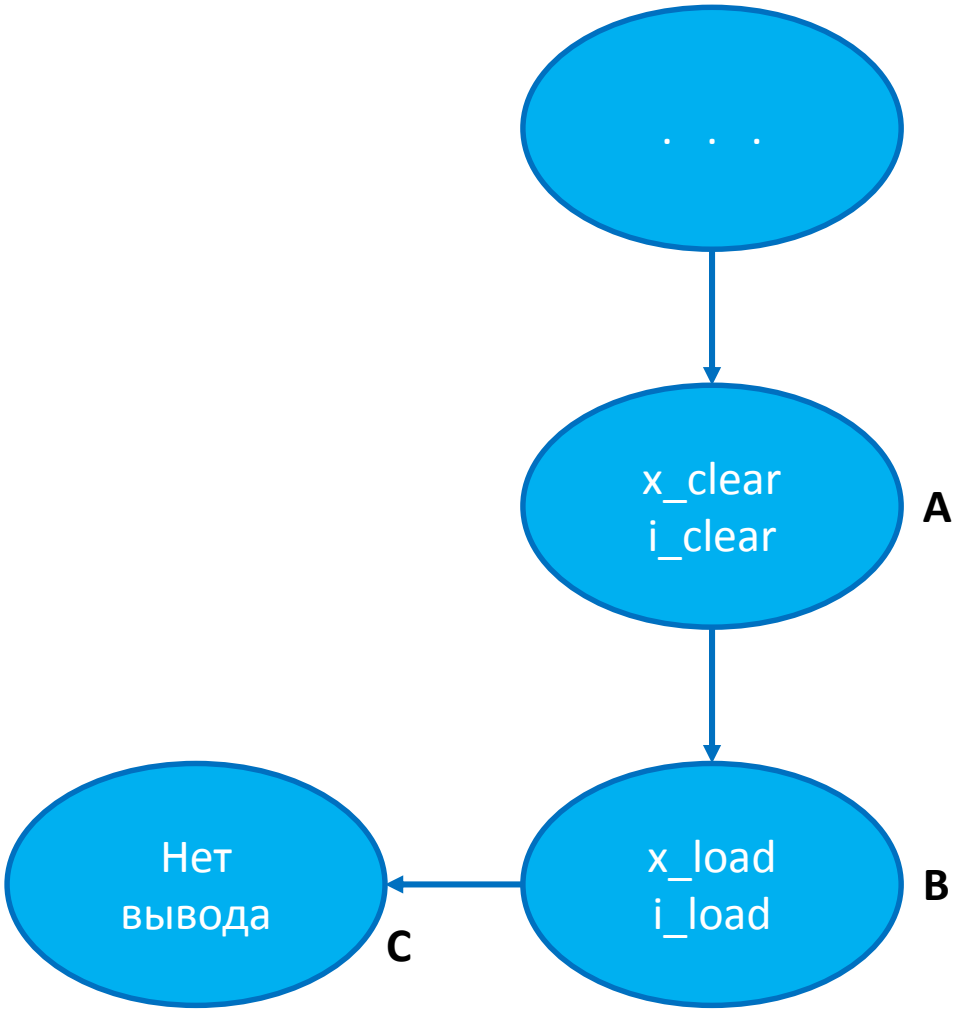


Диаграмма Мура управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```

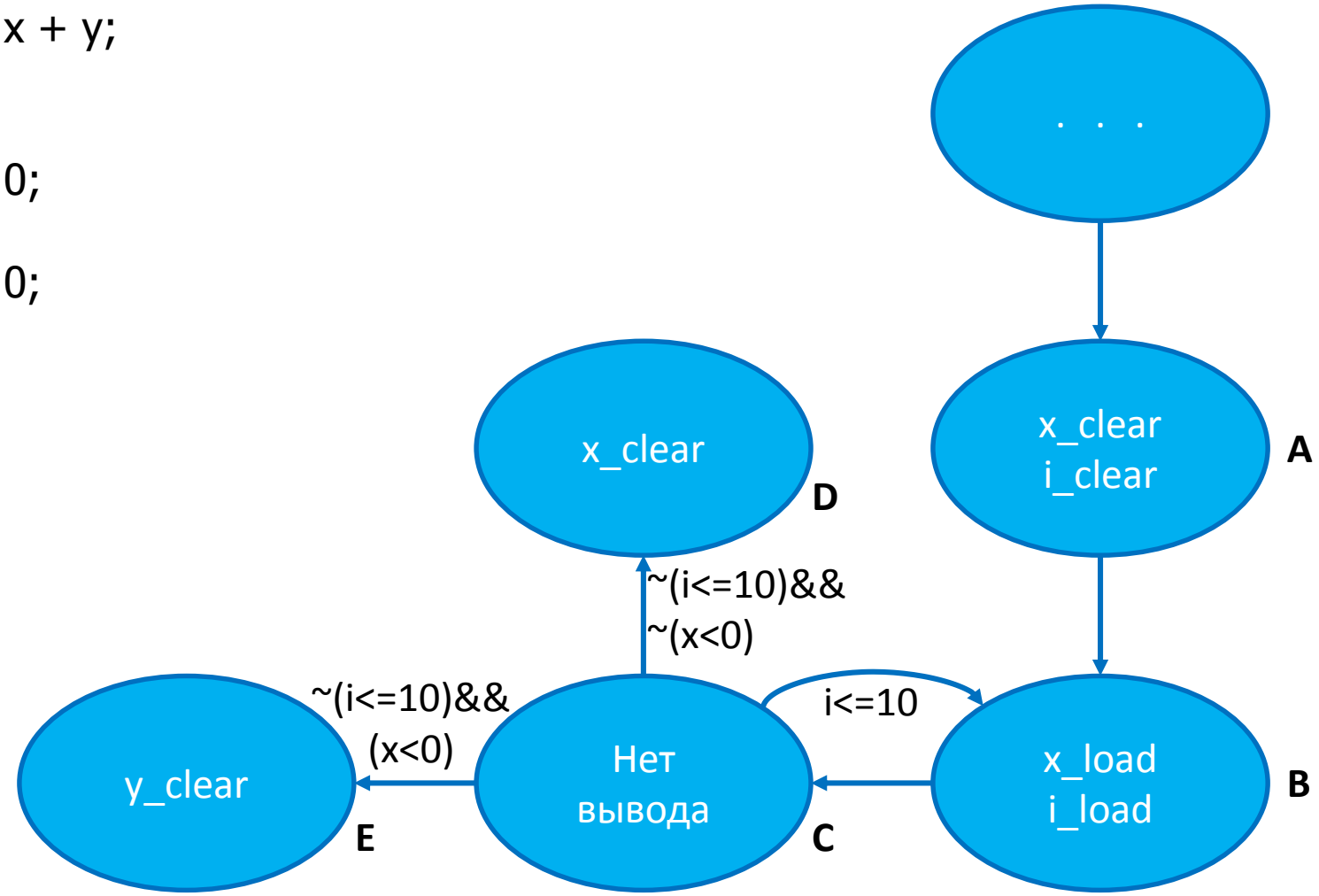
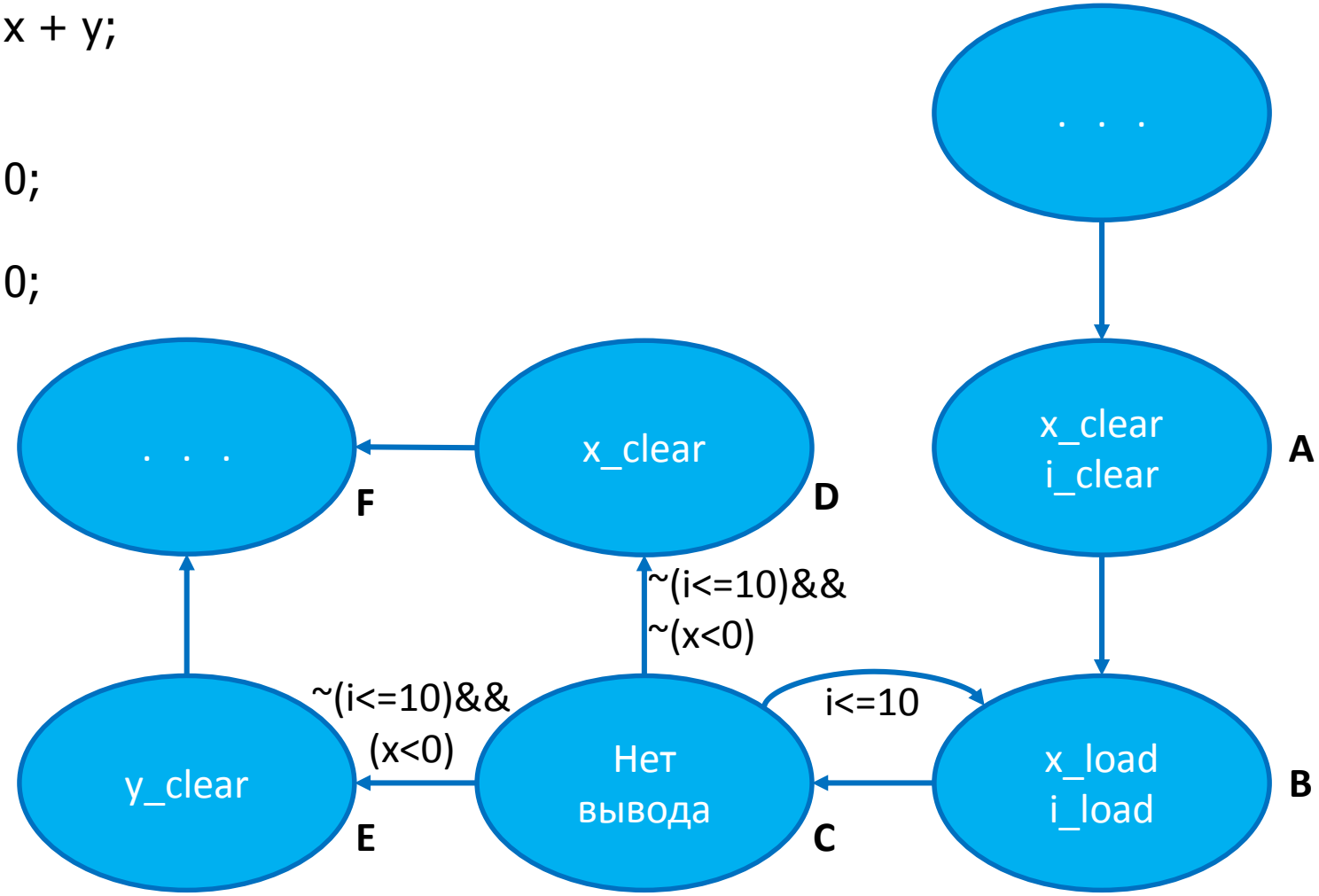


Диаграмма Мура управляющего автомата

```
for(x = 0, i = 0; i <= 10; i++) {
        x = x + y;
}
if(x<0) {
        y = 0;
} else {
        x = 0;
}
```



Построение управляющего автомата

- Кодирование состояний

A: 3'b000

B: 3'b001

C: 3'b010

D: 3'b011

E: 3'b100

F: 3'b101

Построение управляющего автомата

- Реализация комбинационной части

```
always @ (c_state, xLT0, iLEQ10)
        case (c_state)
                3'b000: //State A
                        ...
                3'b001: //State B
                        ...
                3'b010: //State C
                        ...
                3'b011: //State D
                        ...
                3'b100: //State E
                        ...
                3'b101: //State F
                        ...
                default:
        endcase
```

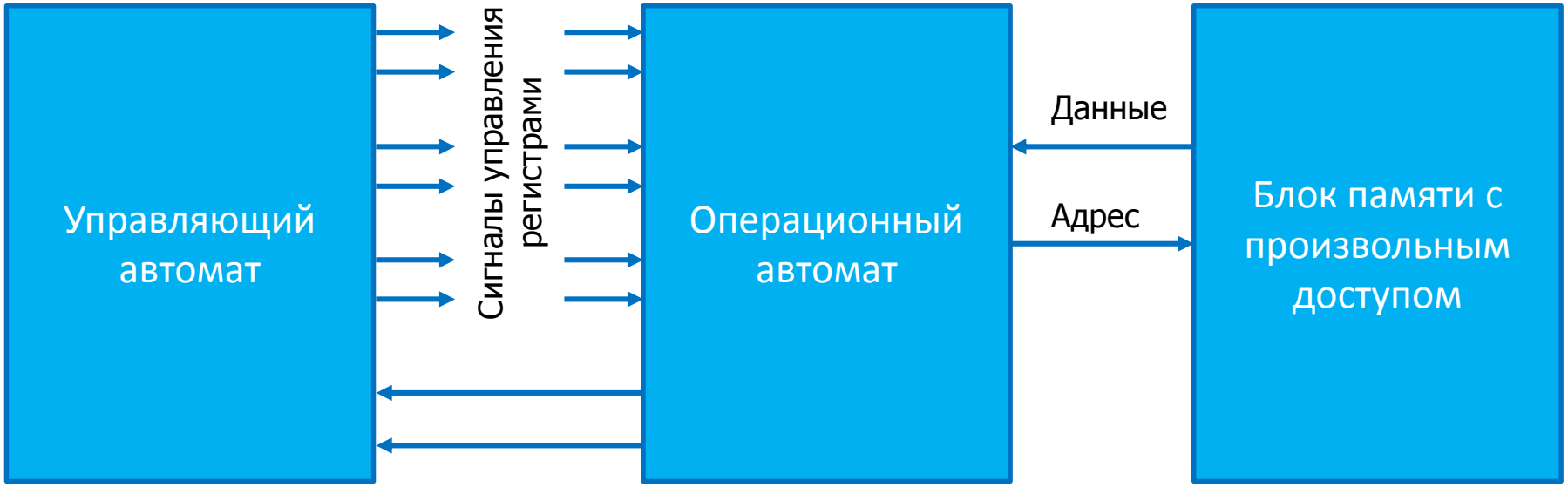
```
3'b000: //State A
        begin
                i_load = 1;
                i_clear = 0;
                x_load = 1;
                x_clear = 0;
                y_load = 1;
                y_clear = 1;
                n_state = 3'b001;
        end
```

Построение управляющего автомата

- Реализация элементов задержки

```
always @ (posedge clock, negedge reset)
        if (~reset)
                c_state <= 0;
        else
                n_state <= c_state;
```

Использование элементов памяти



Файл инициализации памяти (.MIF)

```
DEPTH= 32;
WIDTH= 4;
ADDRESS_RADIX= HEX;
DATA_RADIX= BIN;
CONTENT
        BEGIN
                0 : 0000;
                1 : 0001;
                2 : 0010;
                3 : 0011;
                …
                1E : 1110;
                1F : 1111;
        END;
```