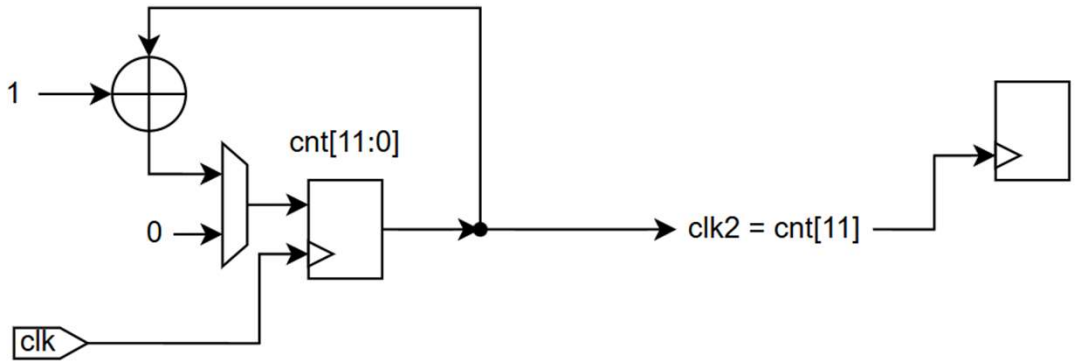
Clk divider

```
module clk_div(
    input clk,
    output clk2
);

    reg [11:0]cnt = 0;
    assign clk2 = cnt[11];

    always @(posedge clk) begin
        cnt <= cnt + 12'b1;
    end

endmodule
```

В чем заключается проблема ?

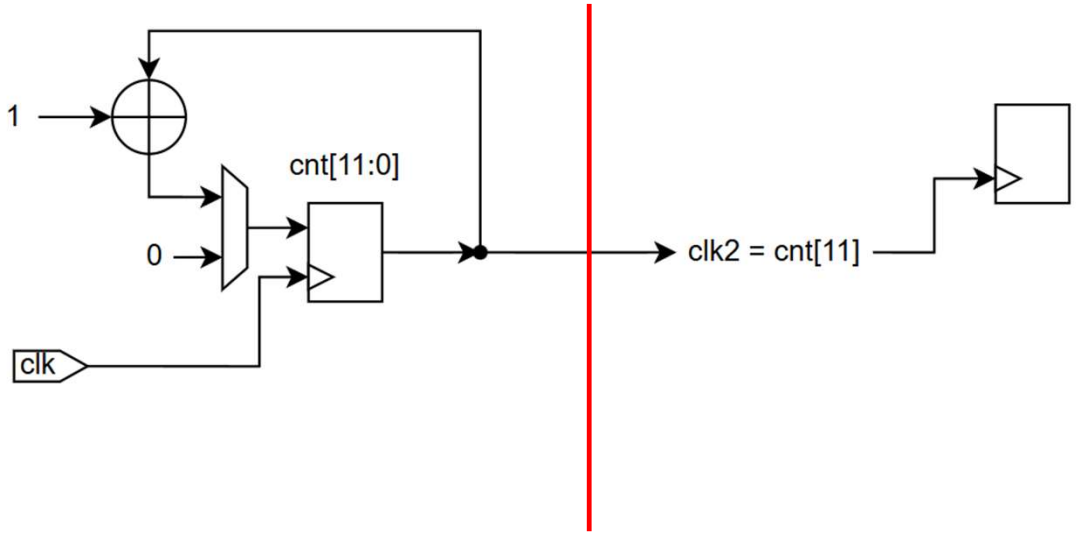


Clk divider

```
module clk_div(
    input clk,
    output clk2
);

    reg [11:0]cnt = 0;
    assign clk2 = cnt[11];

    always @(posedge clk) begin
        cnt <= cnt + 12'b1;
    end

endmodule
```

Мы создаем новый клоковый домен.

1. Усложнение разводки и затраты ресурсов разводки синхронизирующих сигналов.
2. Работа с разными клоковыми доменами и переходами между ними усложняет дизайн.

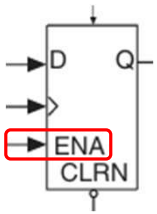


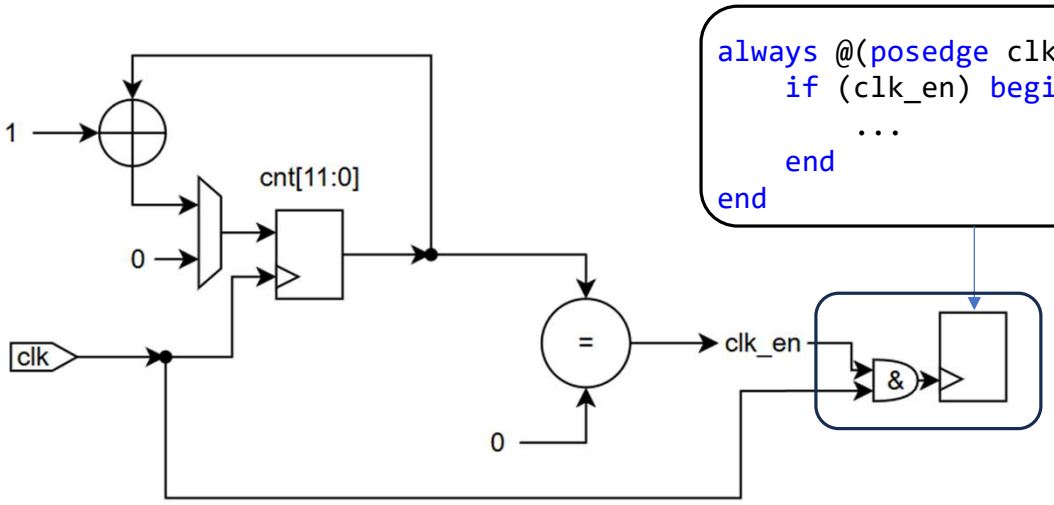
clk enable

```
module clk_div #(
    parameter P_DIVIDER=5
)(
    input clk,
    input clk_en
);

    localparam PW_CNT = $clog2(P_DIVIDER);
    reg [PW_CNT-1: 0] cnt = 0;

    always @(posedge clk) begin
        if (cnt==P_DIVIDER-1) begin
            cnt <= 0;
        end
        else begin
            cnt <= cnt + 1;
        end
    end

    assign clk_en = cnt==0;

endmodule
```

У регистра есть вывод ENA, который позволяет производить запись только в случае если на этом выводе 1.

(нельзя напрямую тактировать от clk&clk_en данная конструкция приведена только для демонстрации принципа!!!)



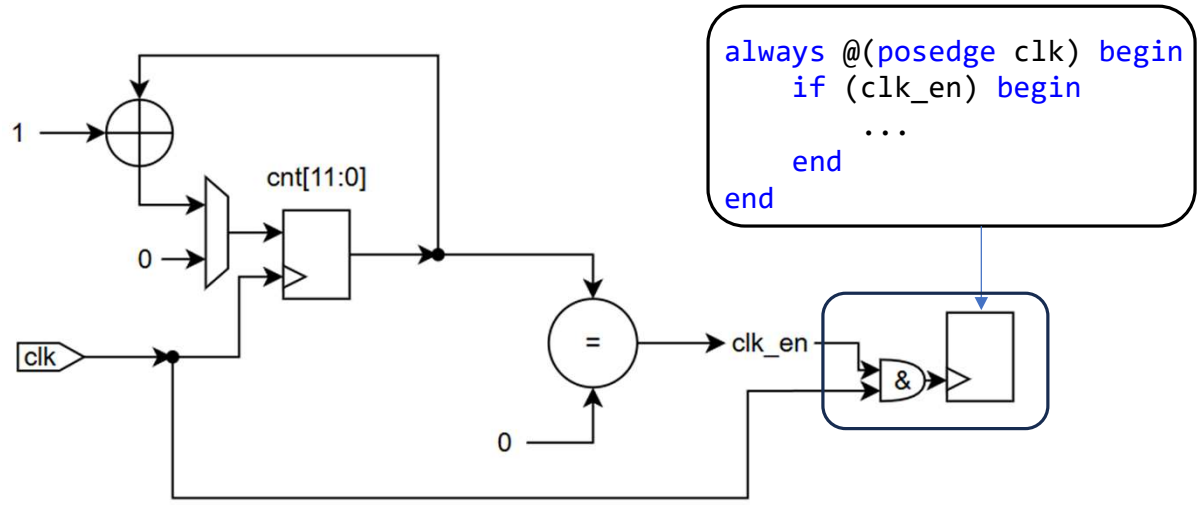



```
always @(posedge clk) begin
    if (clk_en) begin
        ...
    end
end
```

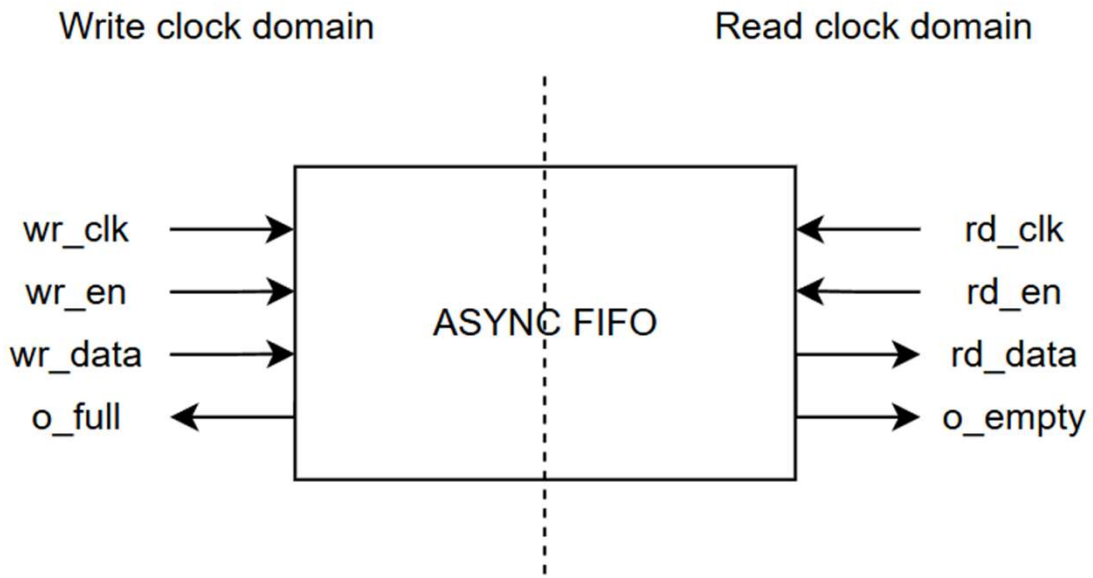
clk enable

```
module clk_div #(
    parameter P_DIVIDER=5
)(
    input clk,
    input clk_en
);

    localparam PW_CNT = $clog2(P_DIVIDER);
    reg [PW_CNT-1: 0] cnt = 0;

    always @(posedge clk) begin
        if (cnt==P_DIVIDER-1) begin
            cnt <= 0;
        end
        else begin
            cnt <= cnt + 1;
        end
    end

    assign clk_en = cnt==0;

endmodule
```

Таким образом мы совершаем действия с требуемой частотой, но не создаем новые клоковые домены.



Async FIFO



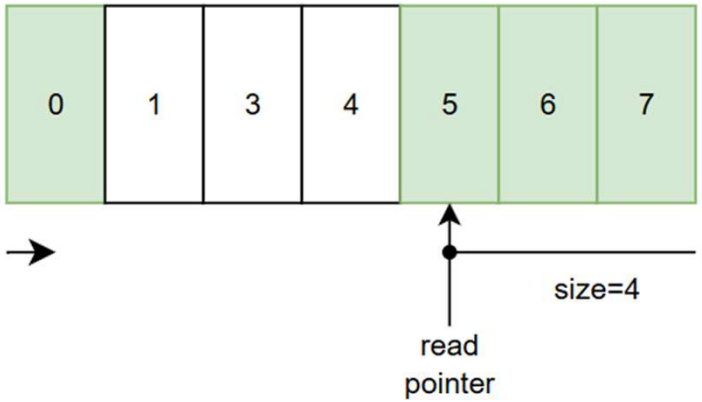
Clifford E. Cummings “Simulation and Synthesis Techniques for Asynchronous FIFO Design”

Циклический буфер (Empty and full)

Для определения мест для чтения и записи можно использовать переменные:

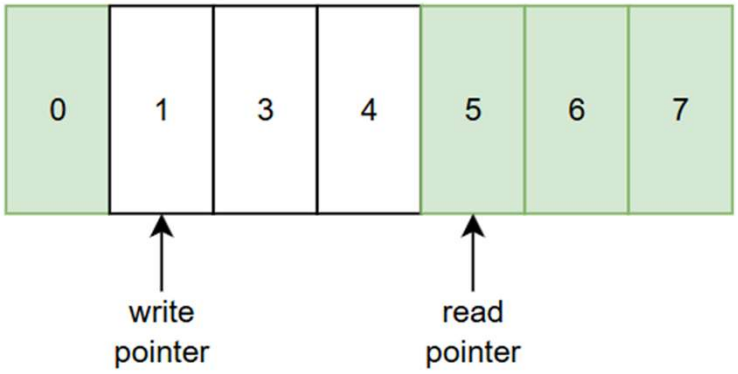
1. Read pointer – следующий для чтения
2. Size – размер очереди
3. Empty = (size==0)
4. Full = (size==size_max-1)

Непонятно как синхронизовать между доменами



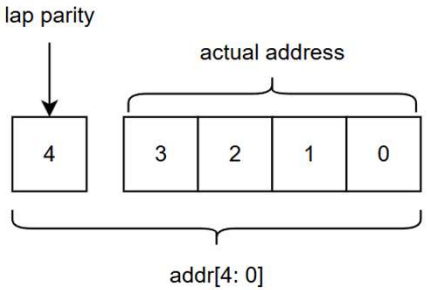
Альтернативно можно использовать указатели на точку чтения и записи:

1. Read pointer (rptr) – следующий для чтения
2. Write pointer (wptr) – следующий для записи
3. Empty = (rptr==wptr)?
4. Full = (rptr==wptr)?



Full and Empty

Рассмотрим FIFO размера 16.
Для указателей понадобится по 4 бита.
Добавим к каждому указателю по 1 дополнительному биту который будет определять **четность “круга”**.



Теперь этот бит определяет full или empty:

1. Если этот бит у указателей разный, то waddr догнал raddr на круг. Значит флаг full должен быть активен
2. Если этот бит одинаковый, то значит колличество кругов пройденных указателями равно. Флаг empty активен.

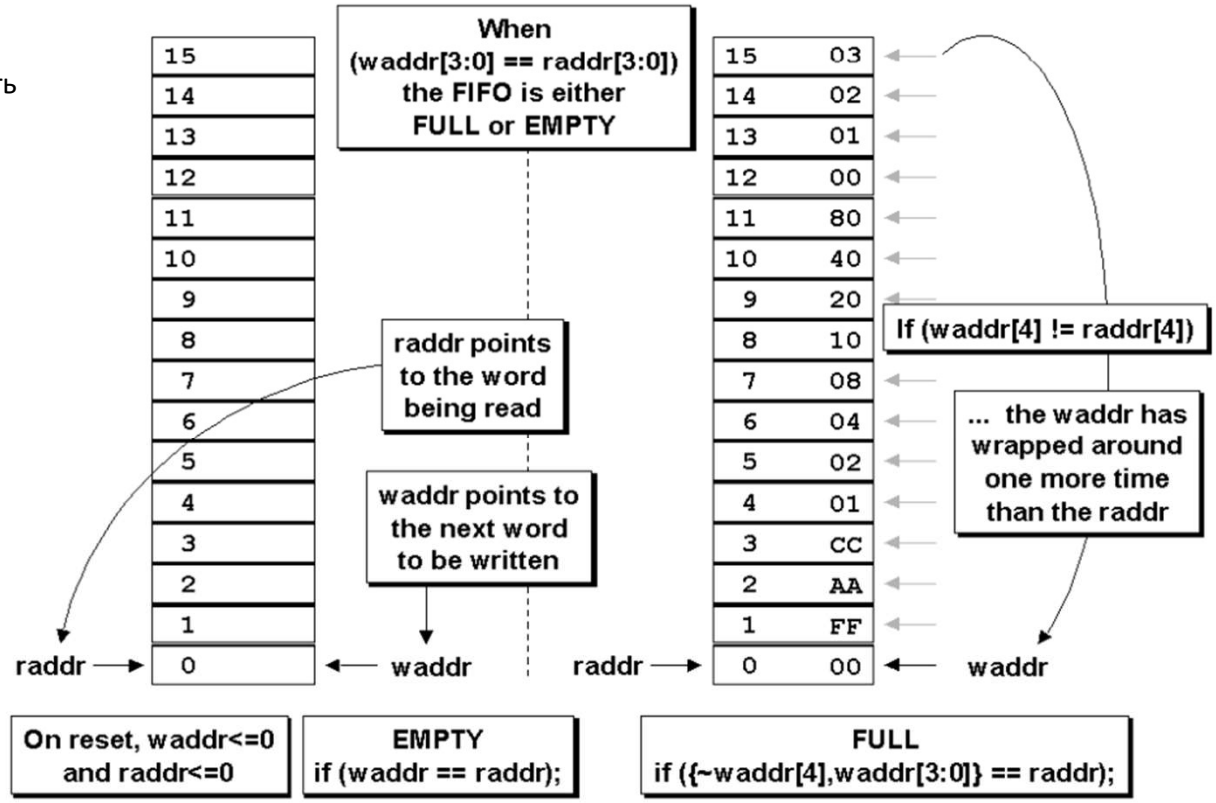
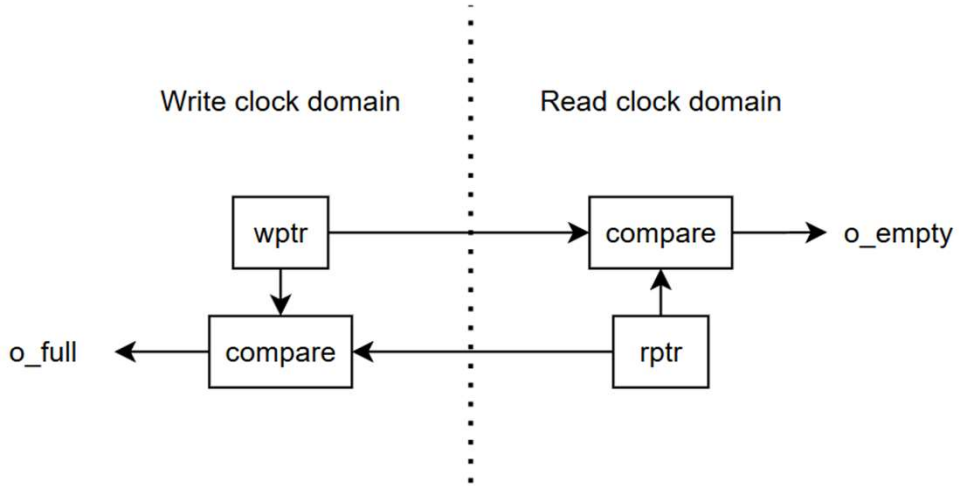


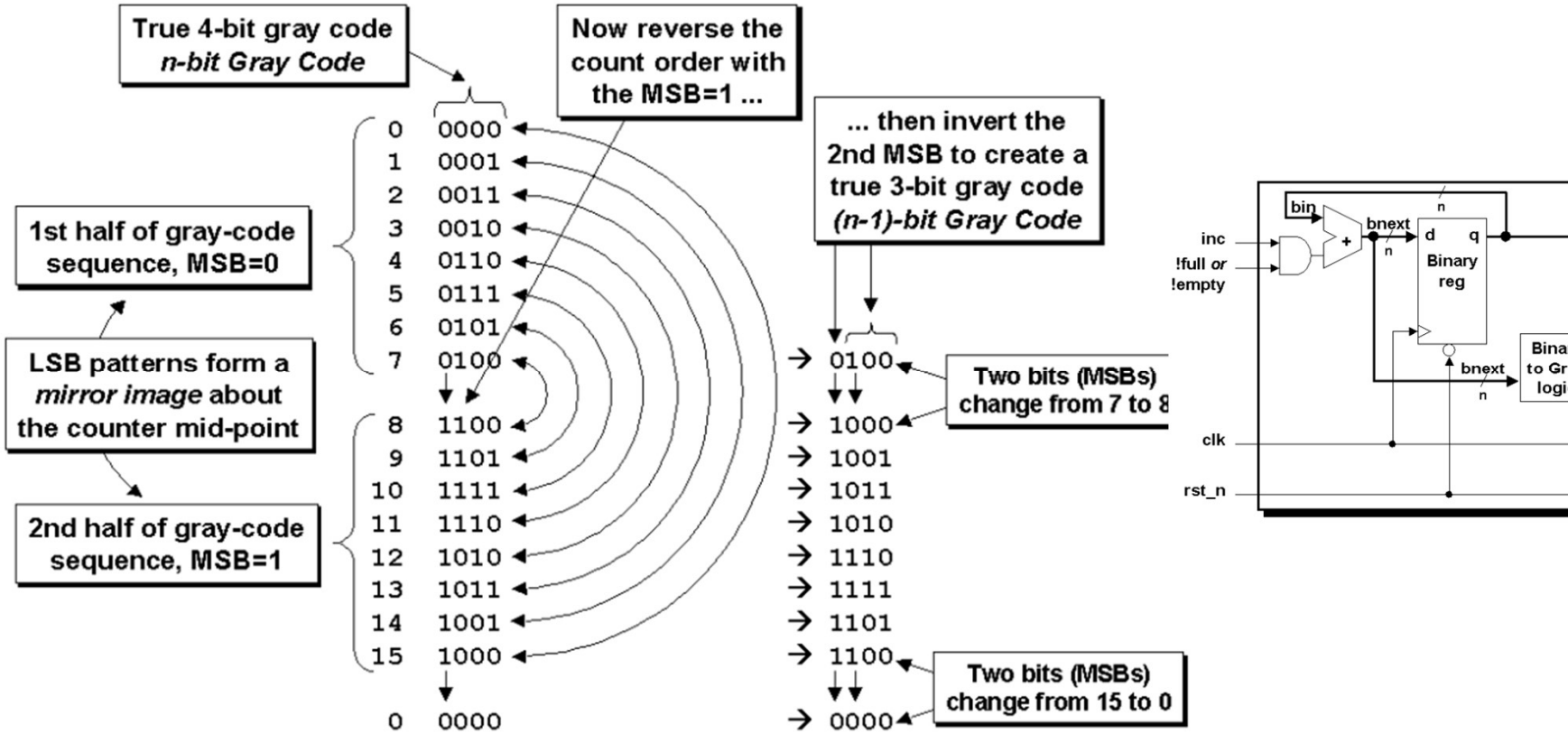
Figure 1 - FIFO full and empty conditions

Синхронизация указателей

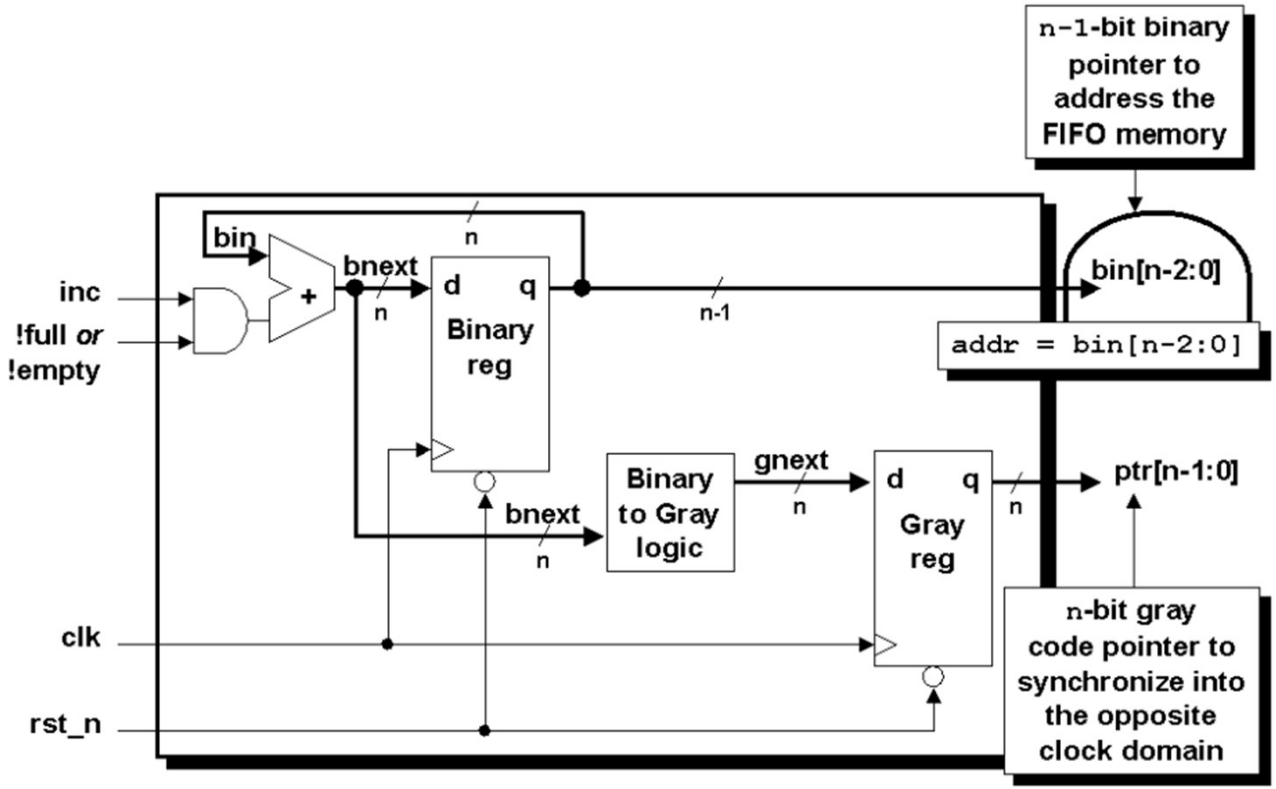
Наиболее логично поместить wptr в домене записи, а rptr в домене чтения. Но для генерации флагов full, empty требуются оба указателя. А это опять синхронизация многобитного значения?



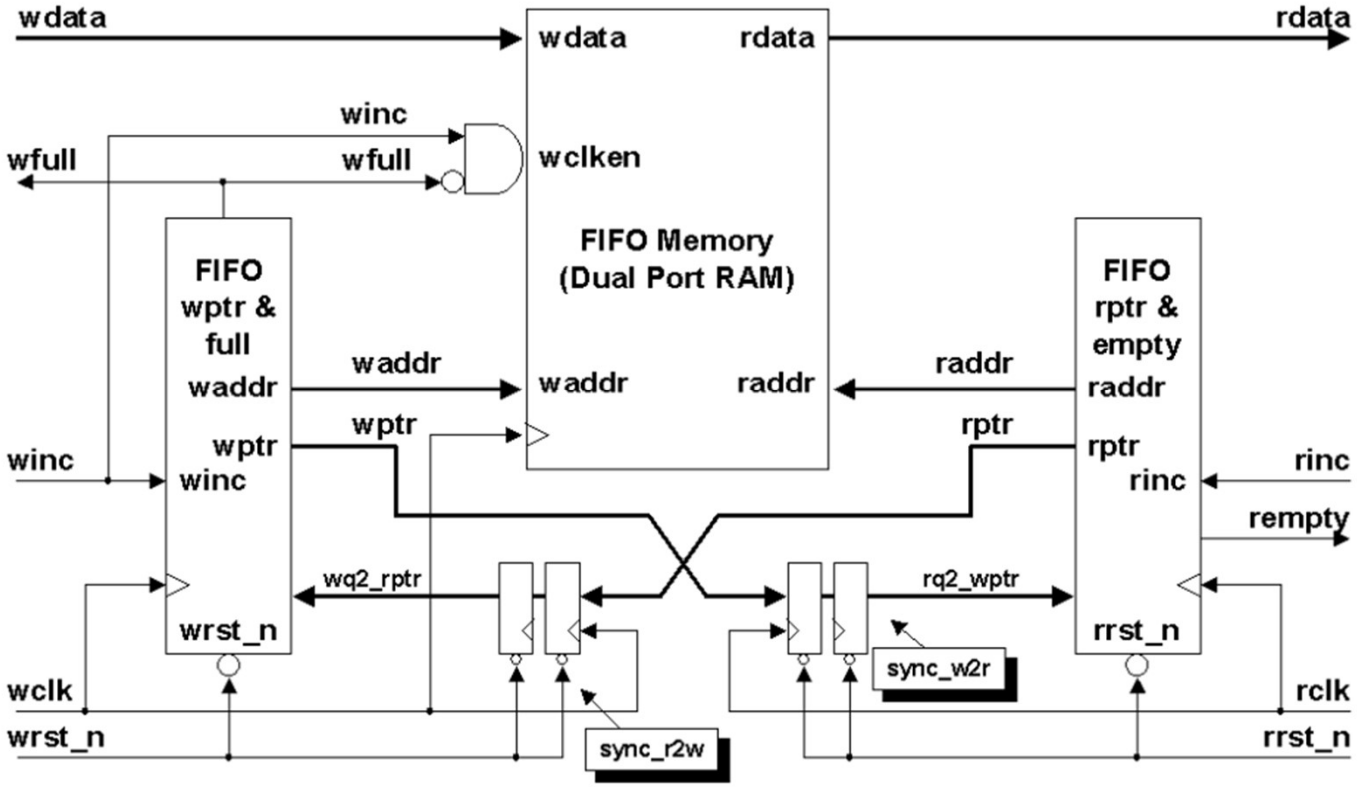
Gray code pointers



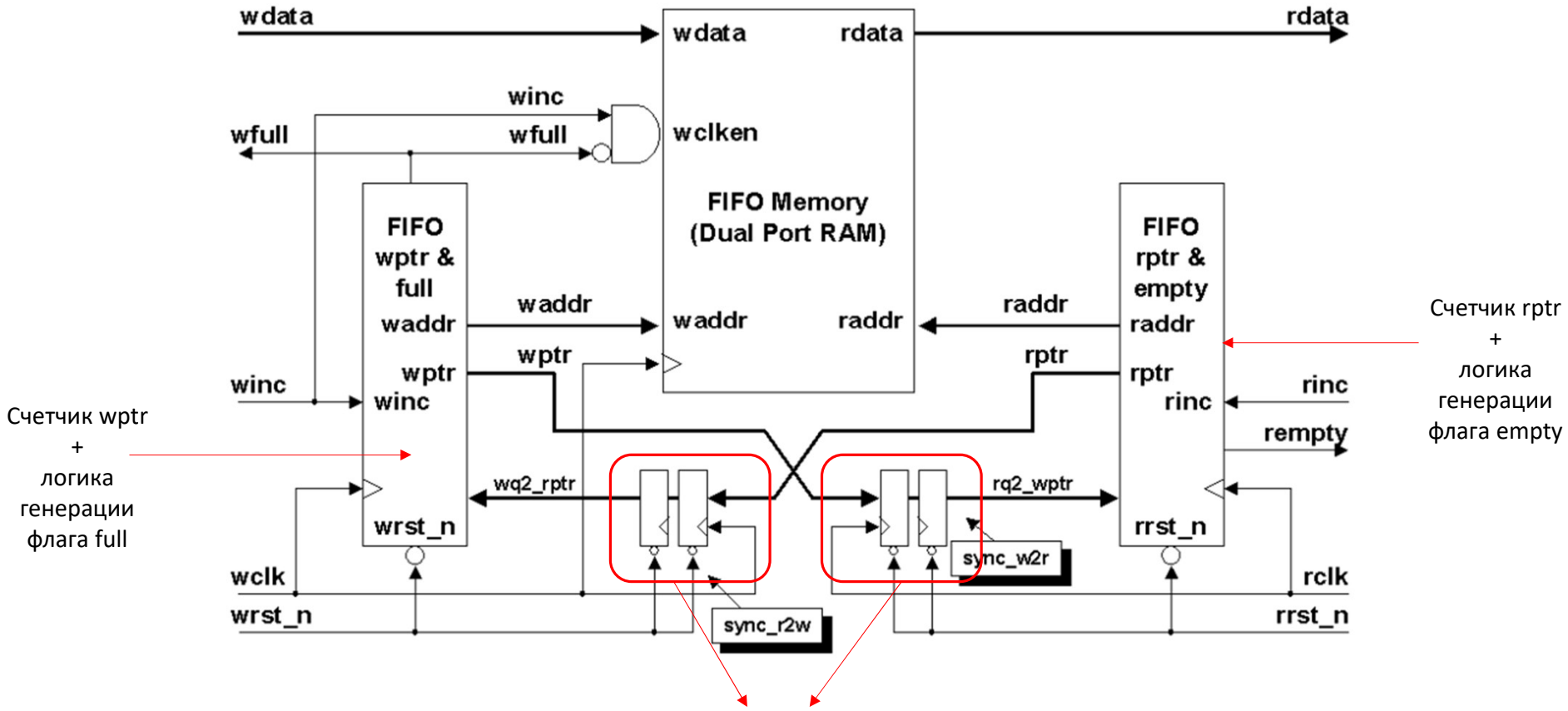
Gray code pointers implementation



Asynchronous FIFO



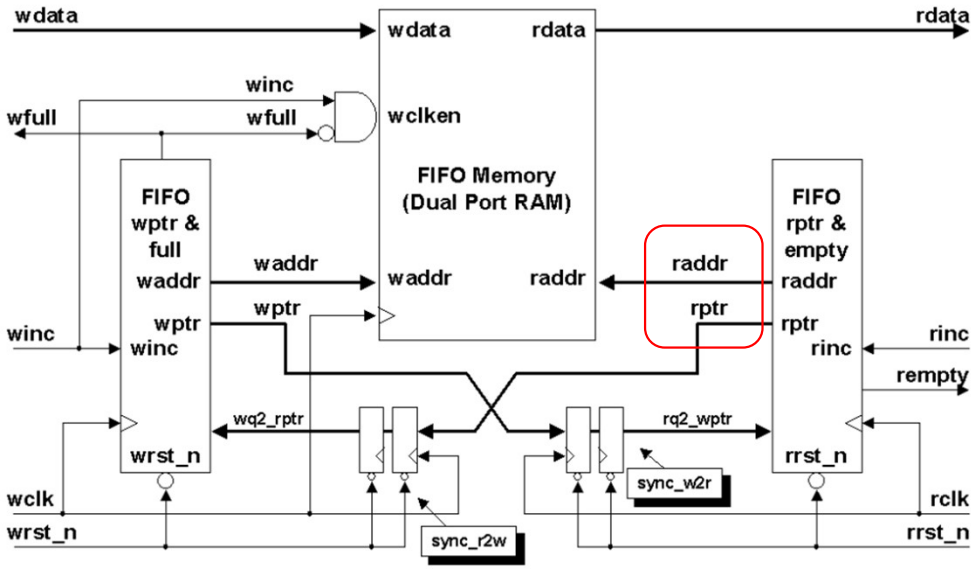
Asynchronous FIFO



Синхронизация указателей в Gray Code формате из одного домена в другой

empty

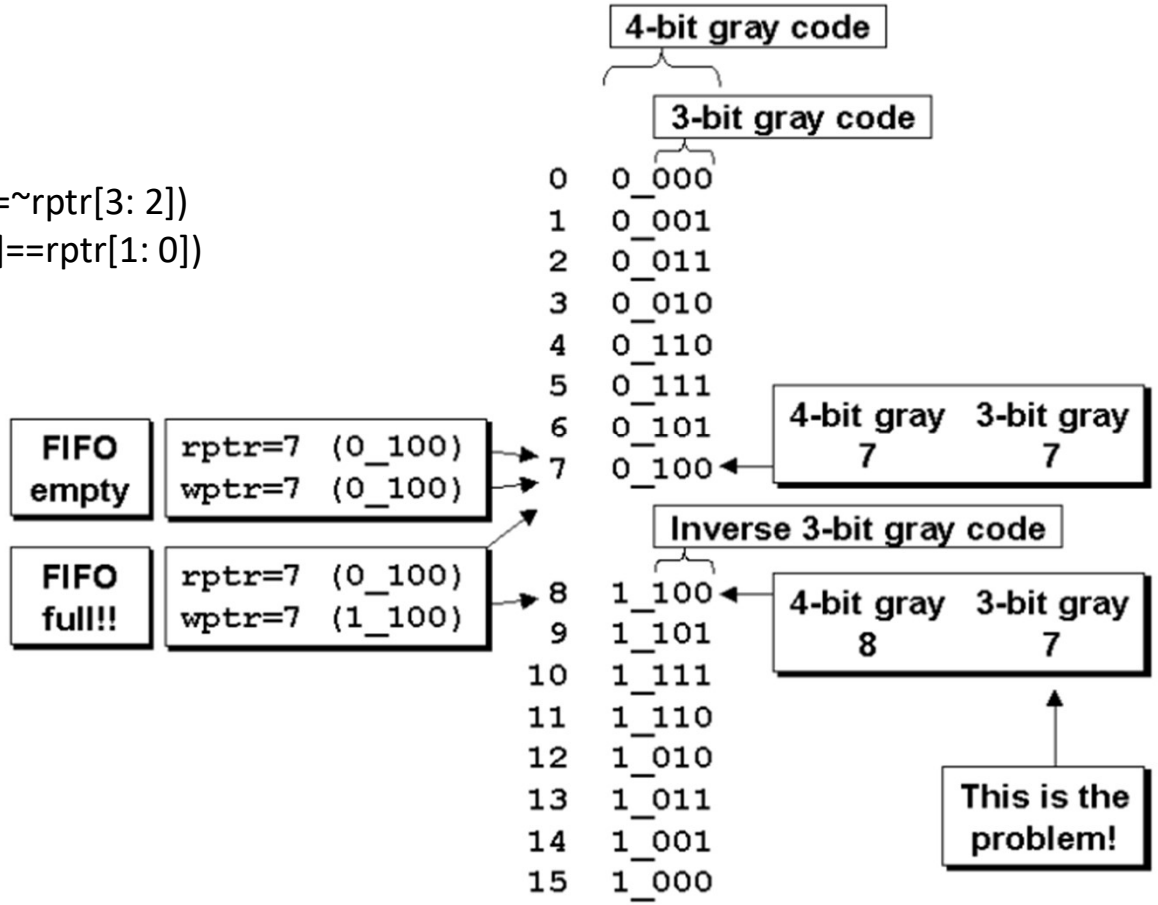
Флаг empty на стороне чтения получается просто: если указатель rptr равен синхронизированному в домен чтения указателю wptr.



full

Условие full:

1. 2 старших бита не равны (wptr[3: 2]==~rptr[3: 2])
2. Все остальные биты равны (wptr[1: 0]==rptr[1: 0])



Потенциальные проблемы в дизайне.

Представим что частота wr_clk значительно выше rd_clk. За время периода rd_clk указатель на запись (wptr) инкрементируется несколько раз. Тогда получается что даже с использованием кода Грея поменяется сразу несколько бит. Не приведет ли это к проблеме в синхронизации указателей?

Ответ: Нет. Проблема возникает когда несколько бит меняются одновременно вблизи фронта синхросигнала и часть из бит имеет предыдущее значение, часть новое, а какие то попадают в метастабильность. Здесь же в момент фронта будет меняться только один сигнал.

Потенциальные проблемы в дизайне.

Представим что частота wr_clk значительно выше rd_clk. За время периода rd_clk указатель на запись (wptr) может инкрементироваться несколько раз. Не возникнет ли ситуация в которой fifo первой записью заполнится, а второй записью переполнится?

Ответ: Нет. Т.к. в данной реализации используется pessimistic full&empty. Флаг full определяется на стороне записи сразу же как наступает данное событие