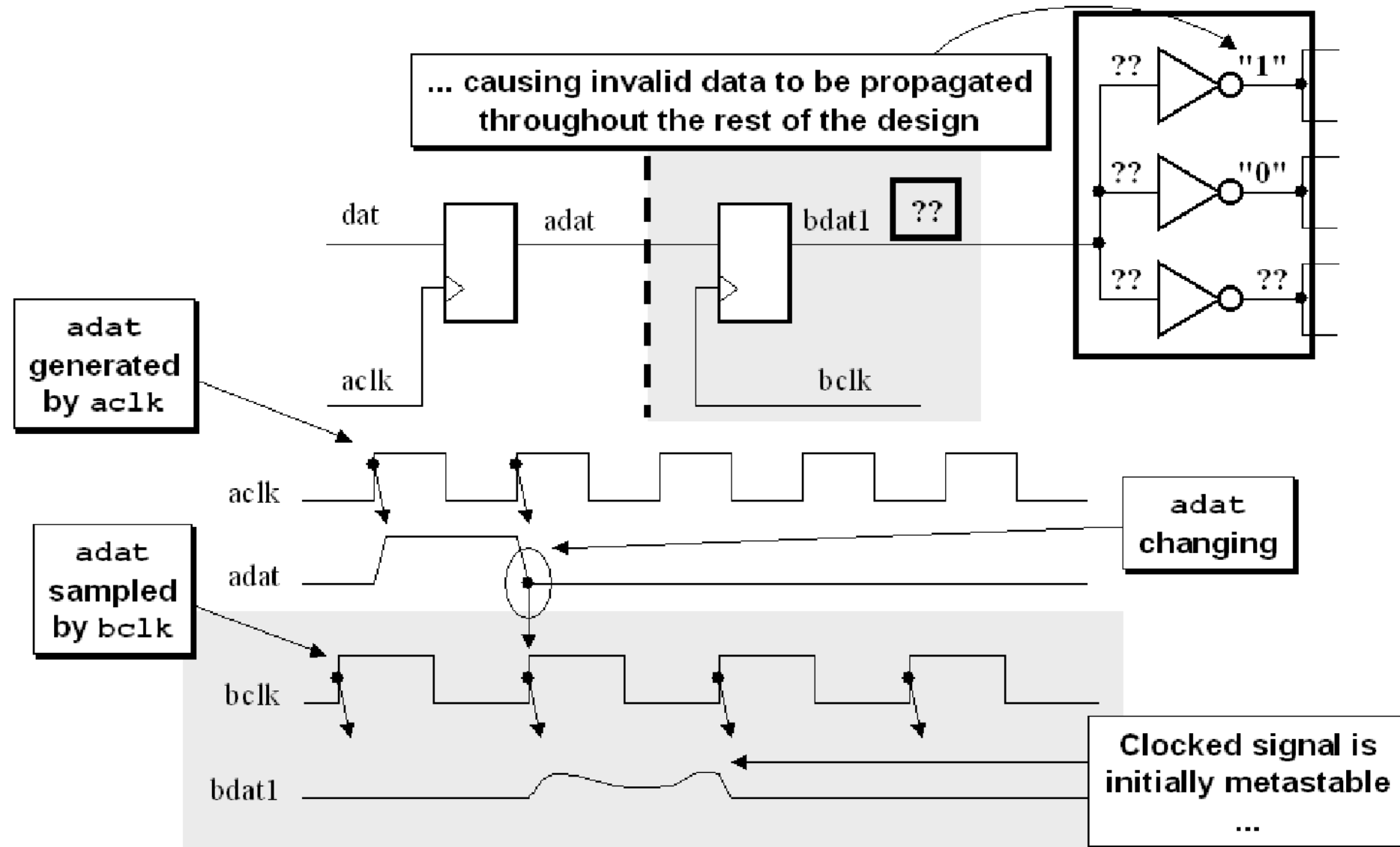


Clock domain crossing

[http://www.sunburst-design.com/papers/CummingsSNUG2008Boston\\_CDC.pdf](http://www.sunburst-design.com/papers/CummingsSNUG2008Boston_CDC.pdf)

# Почему метастабильность проблема



### 3.1 Two synchronization scenarios

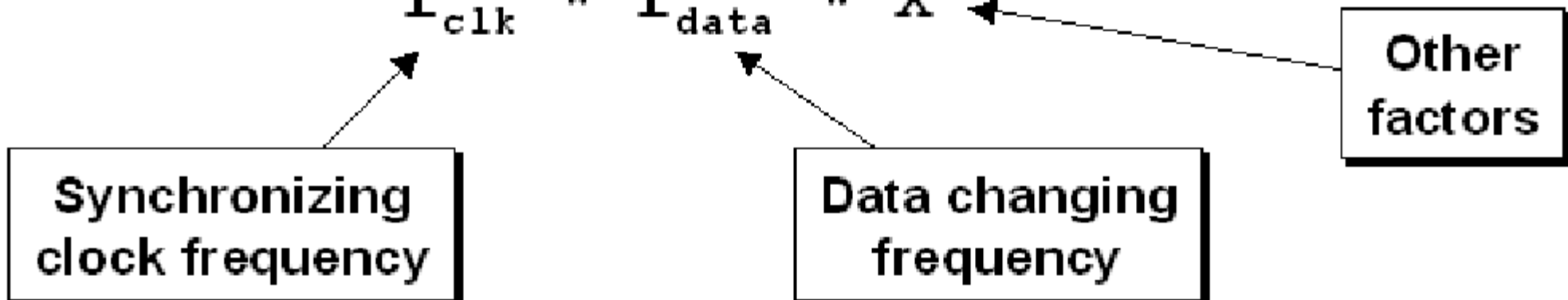
There are two scenarios that are possible when passing signals across CDC boundaries, and it is important to determine which scenario applies to your design:

- (1) It is permitted to miss samples that are passed between clock domains.
- (2) Every signal passed between clock domains must be sampled.

**First scenario:** sometimes it is not necessary to sample every value, but it is important that the sampled values are accurate. One example is the set of gray code counters used in a standard asynchronous FIFO design. In a properly designed asynchronous FIFO model, synchronized gray code counters do not need to capture every legal value from the opposite clock domain, but it is critical that sampled values be accurate to recognize when full and empty conditions have occurred.

**Second scenario:** a CDC signal must be properly recognized or recognized and acknowledged before a change is permitted on the CDC signal.

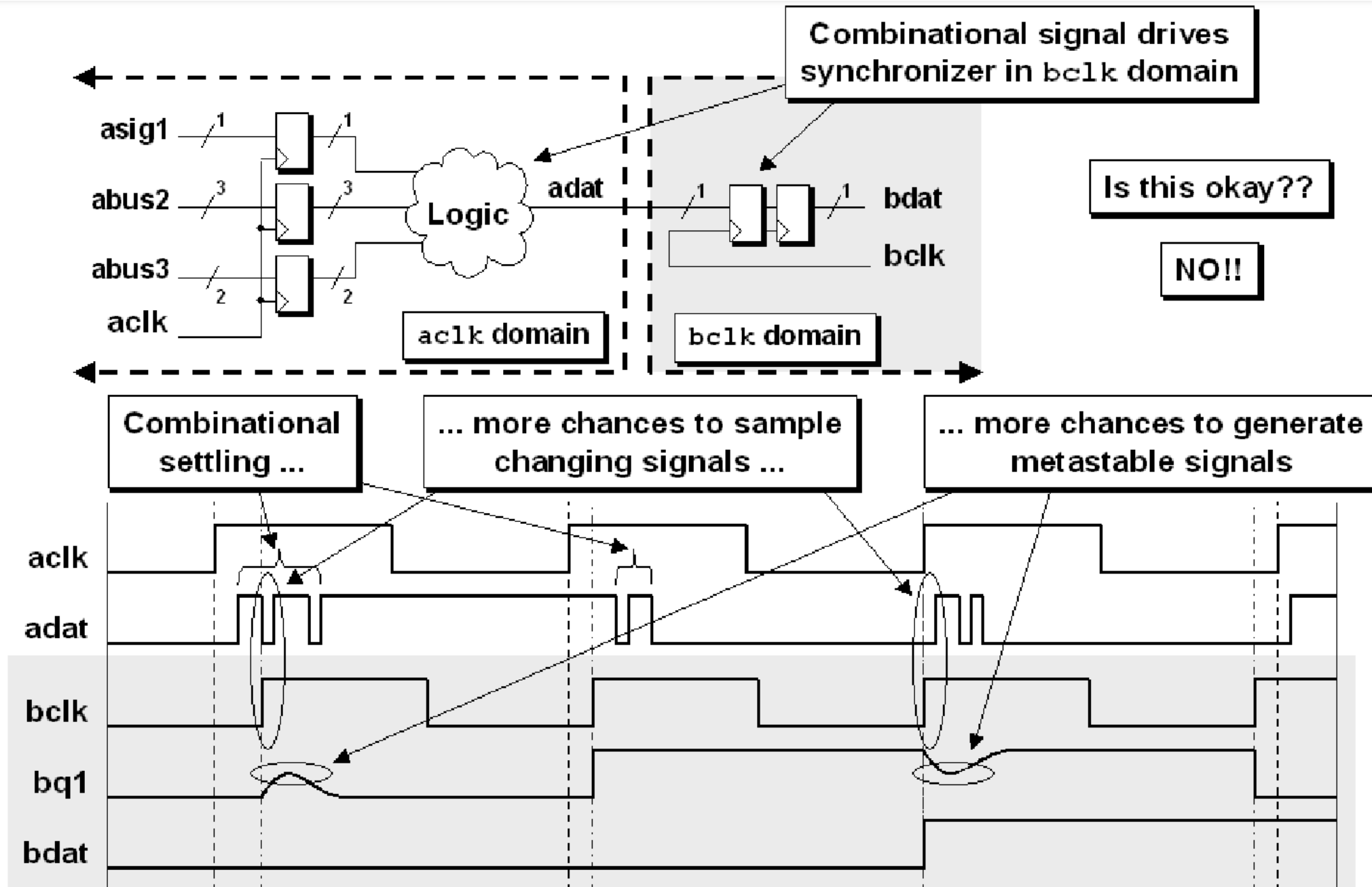
# Mean time between failures

$$\text{MTBF} = \frac{1}{f_{\text{clk}} * f_{\text{data}} * X}$$


The diagram illustrates the factors contributing to MTBF. Three boxes at the bottom point to the variables in the equation: 'Synchronizing clock frequency' points to  $f_{\text{clk}}$ , 'Data changing frequency' points to  $f_{\text{data}}$ , and 'Other factors' points to  $X$ .

**Figure 4 - Primary contributing factors to short MTBF values**

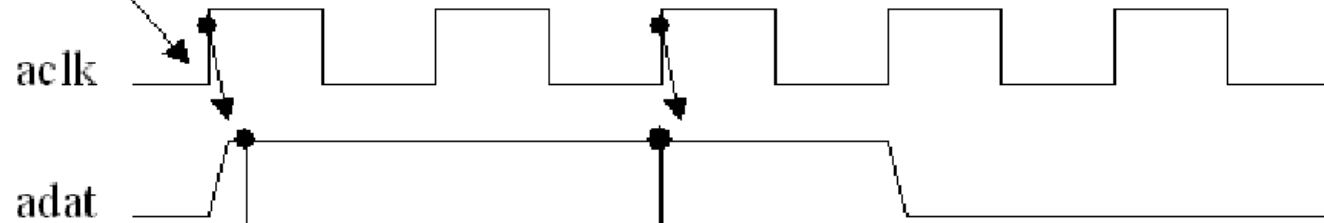
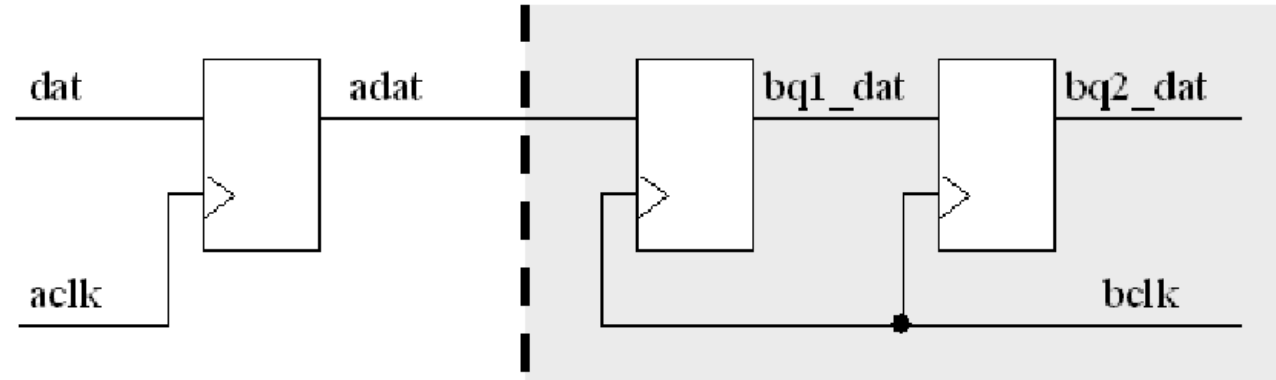
# Синхронизация в исходном домене



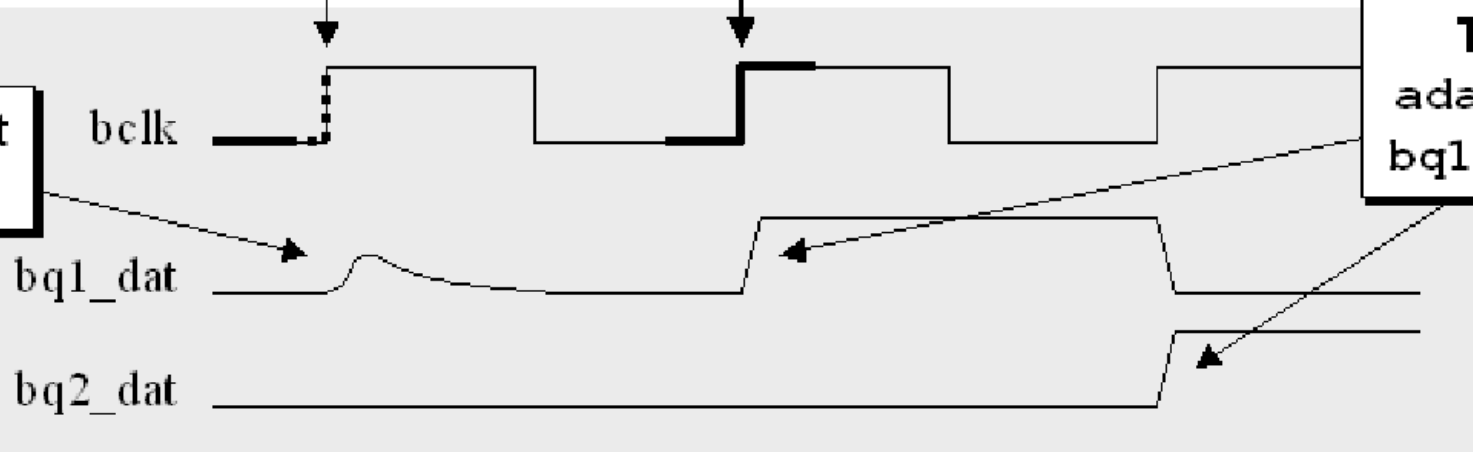
# Из быстрого в медленный, один сигнал

## "Open-Loop" solution

adat pulse should  
be 1-1/2 bclk  
periods in width

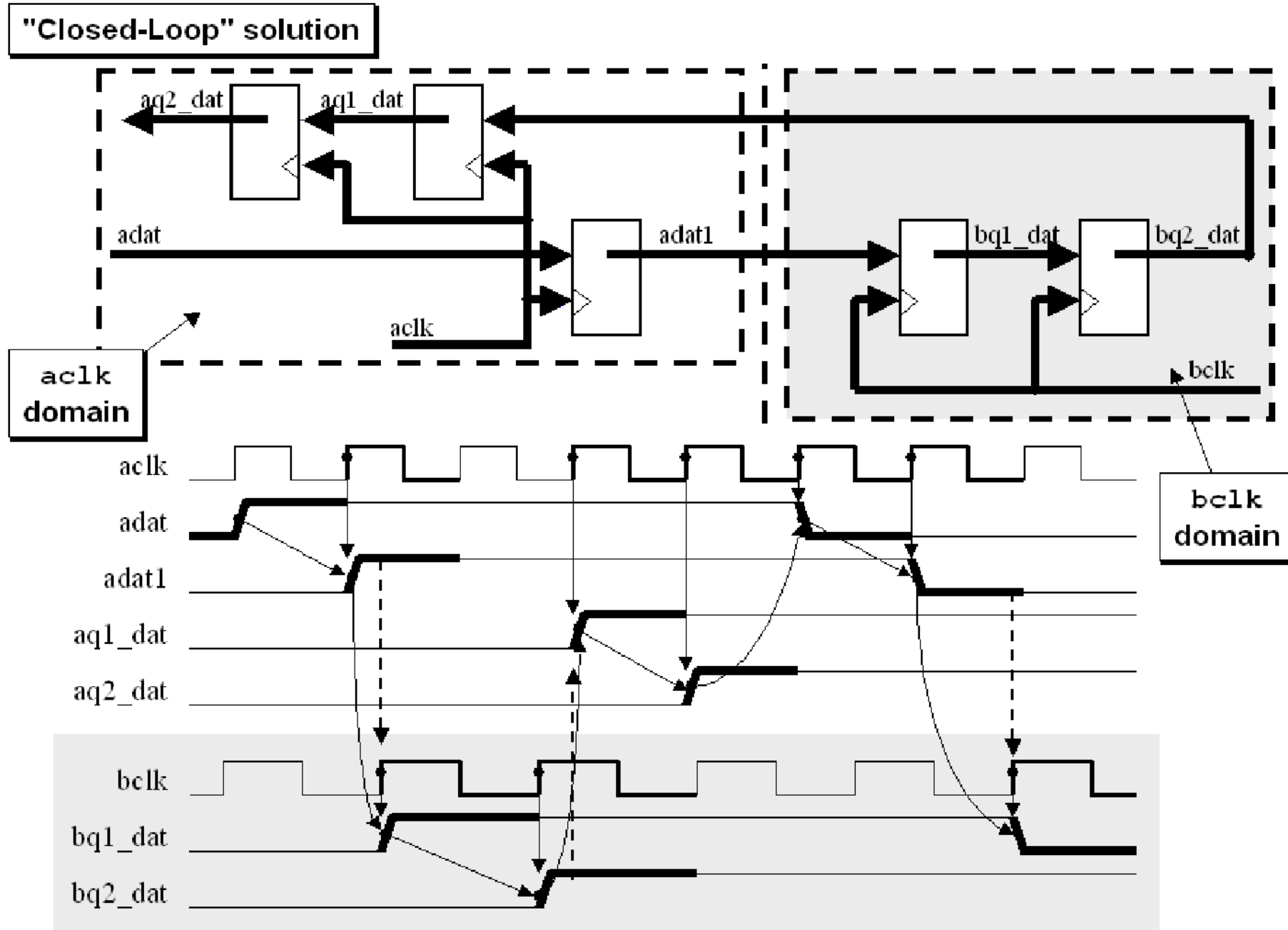


Pulse not  
formed



This insures that  
adat is propagated to  
bq1\_dat and bq2\_dat

# Из быстрого в медленный, один сигнал

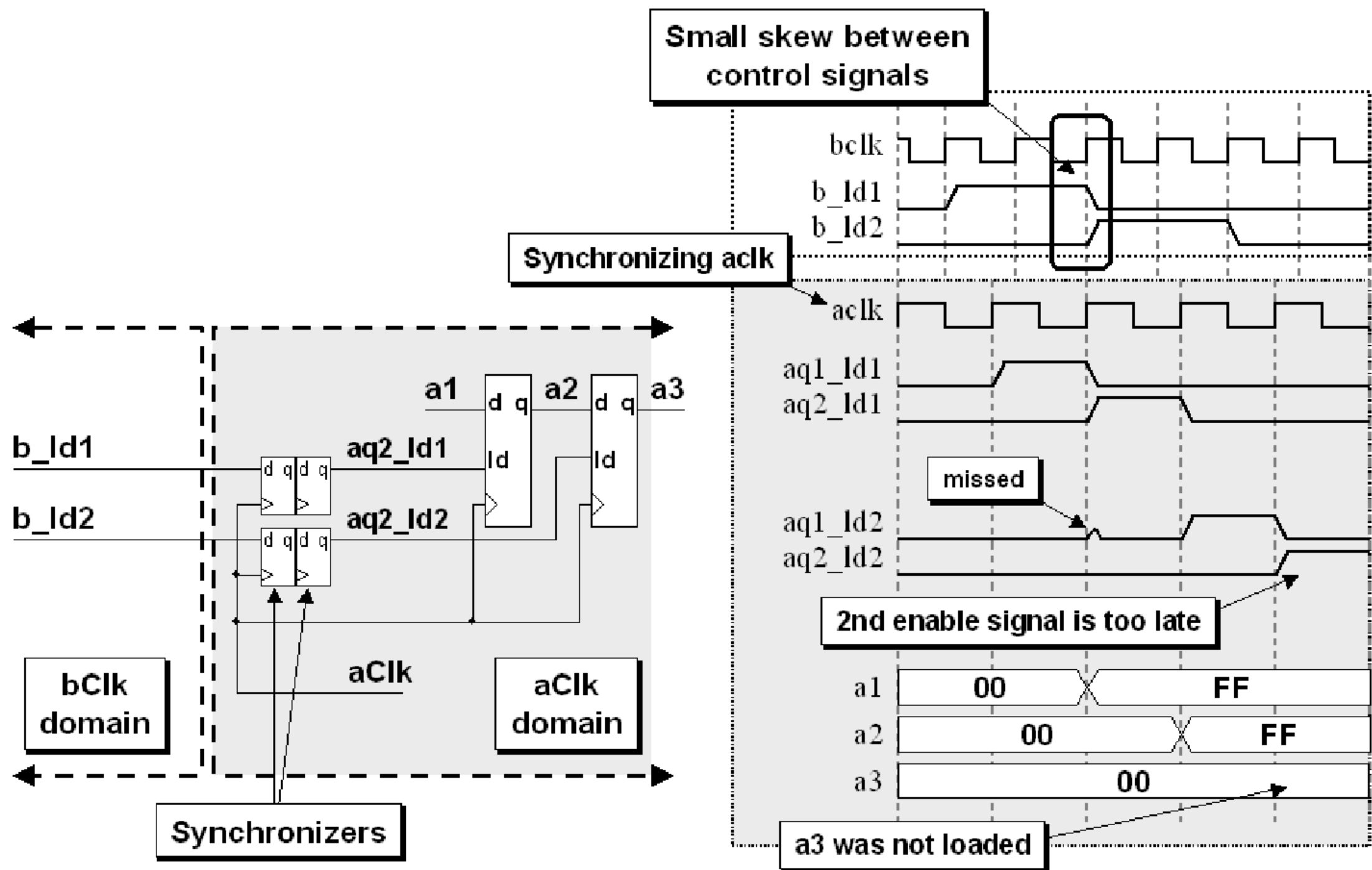


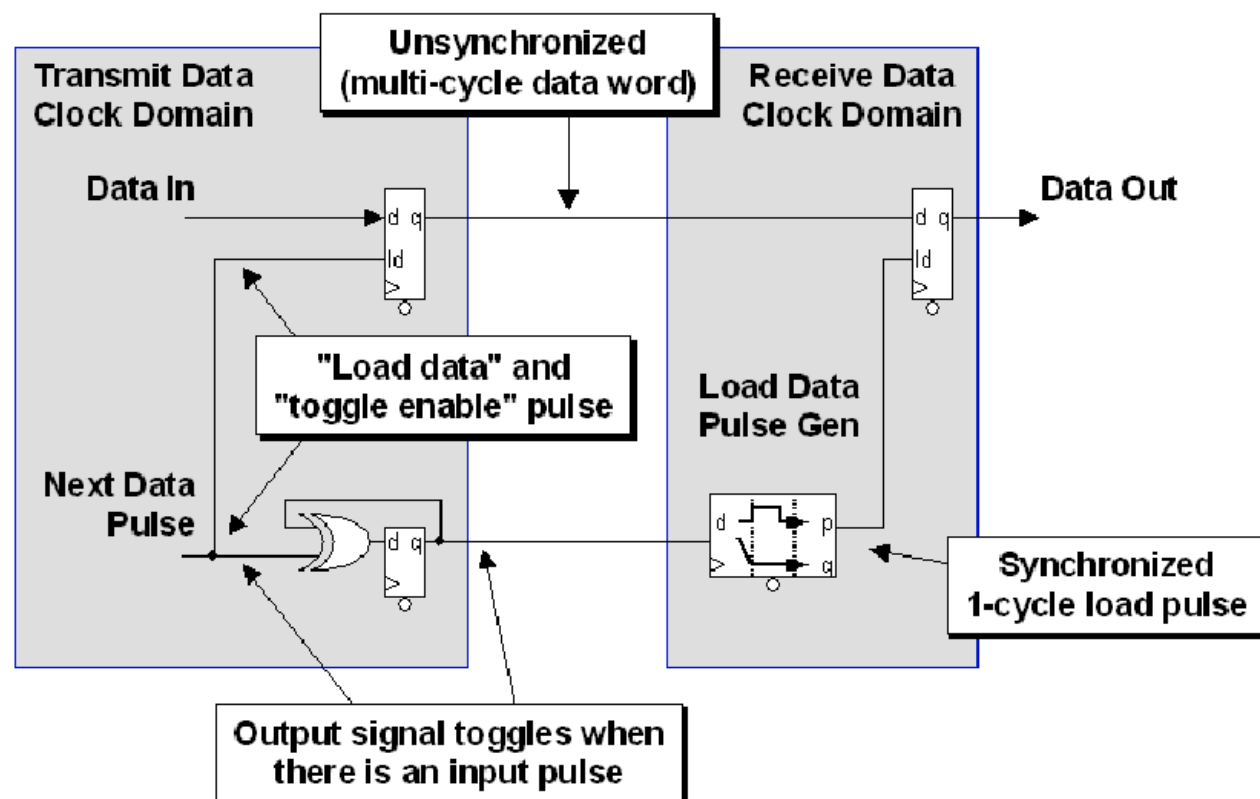
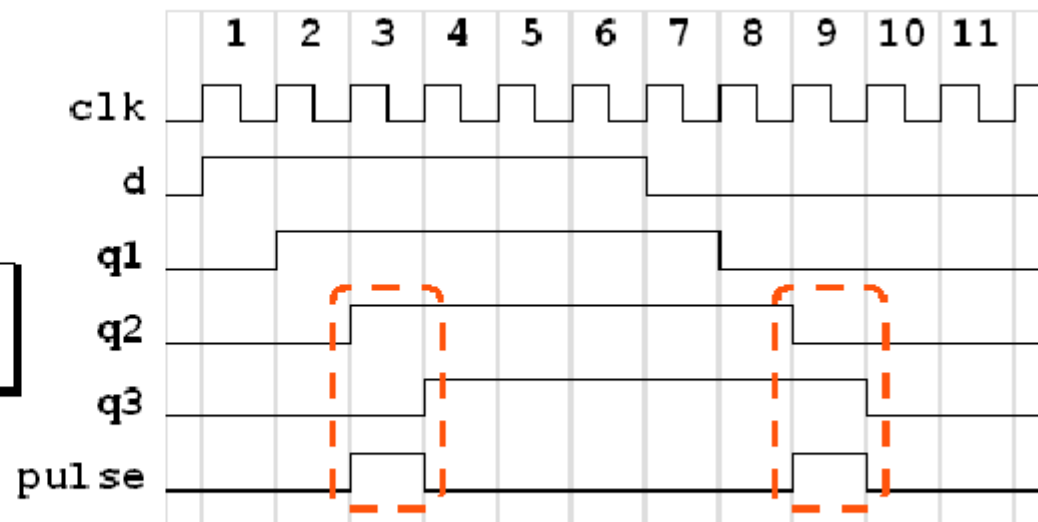
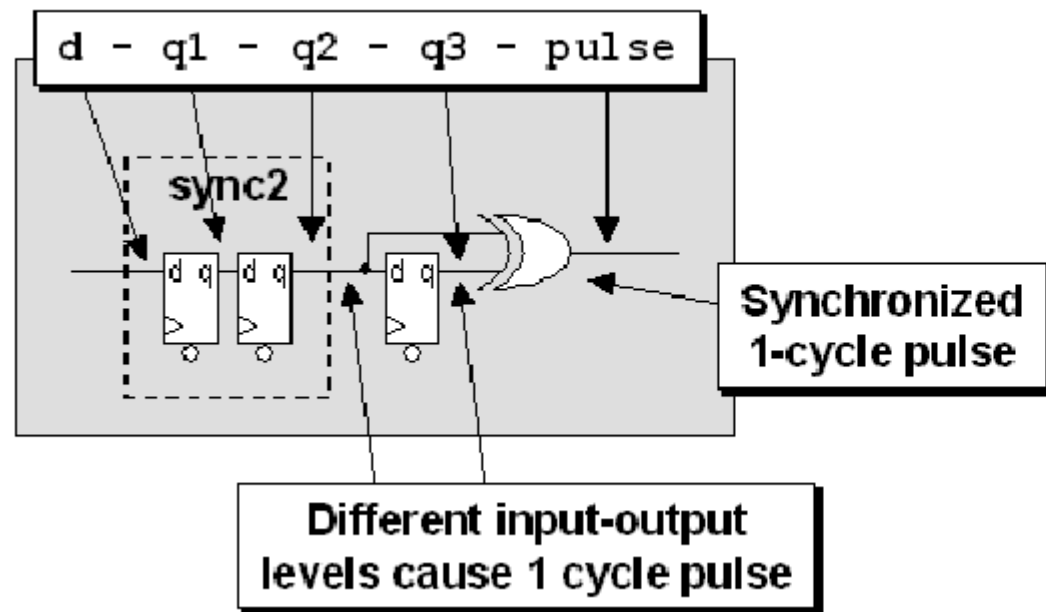


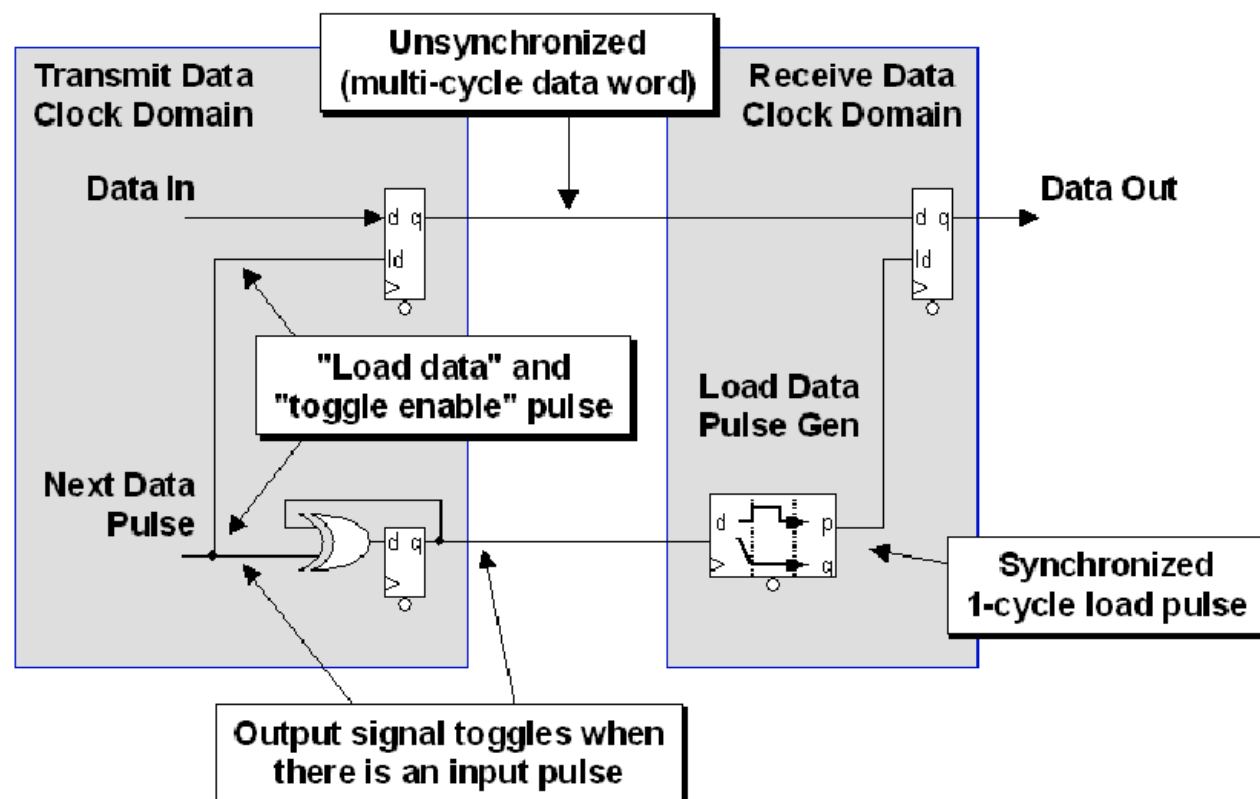
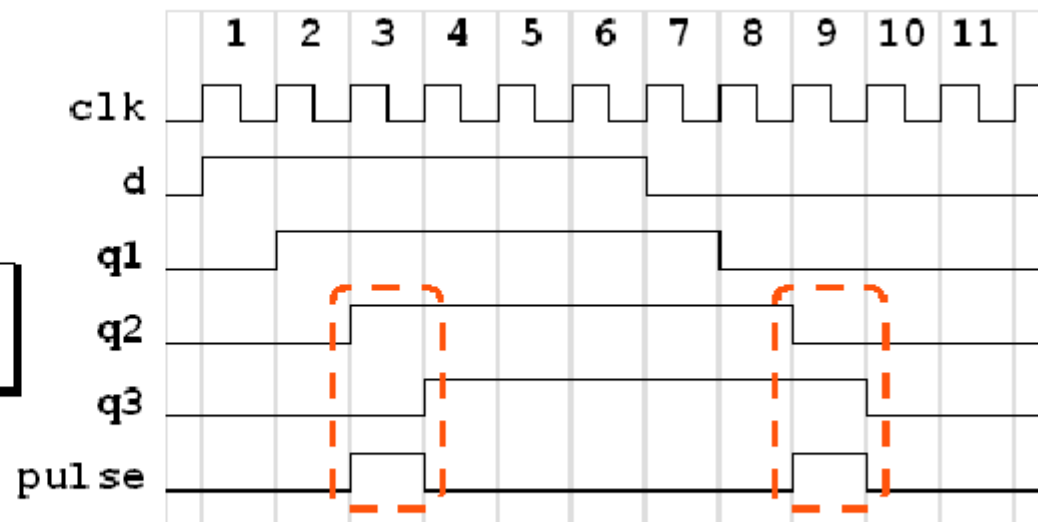
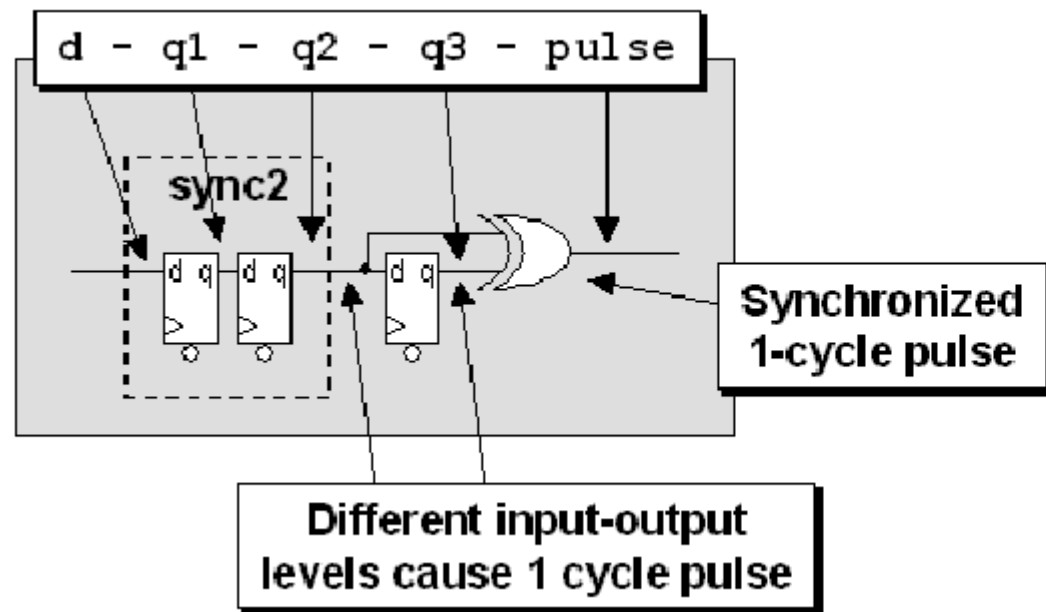
## **5.1 Multi-bit CDC strategies**

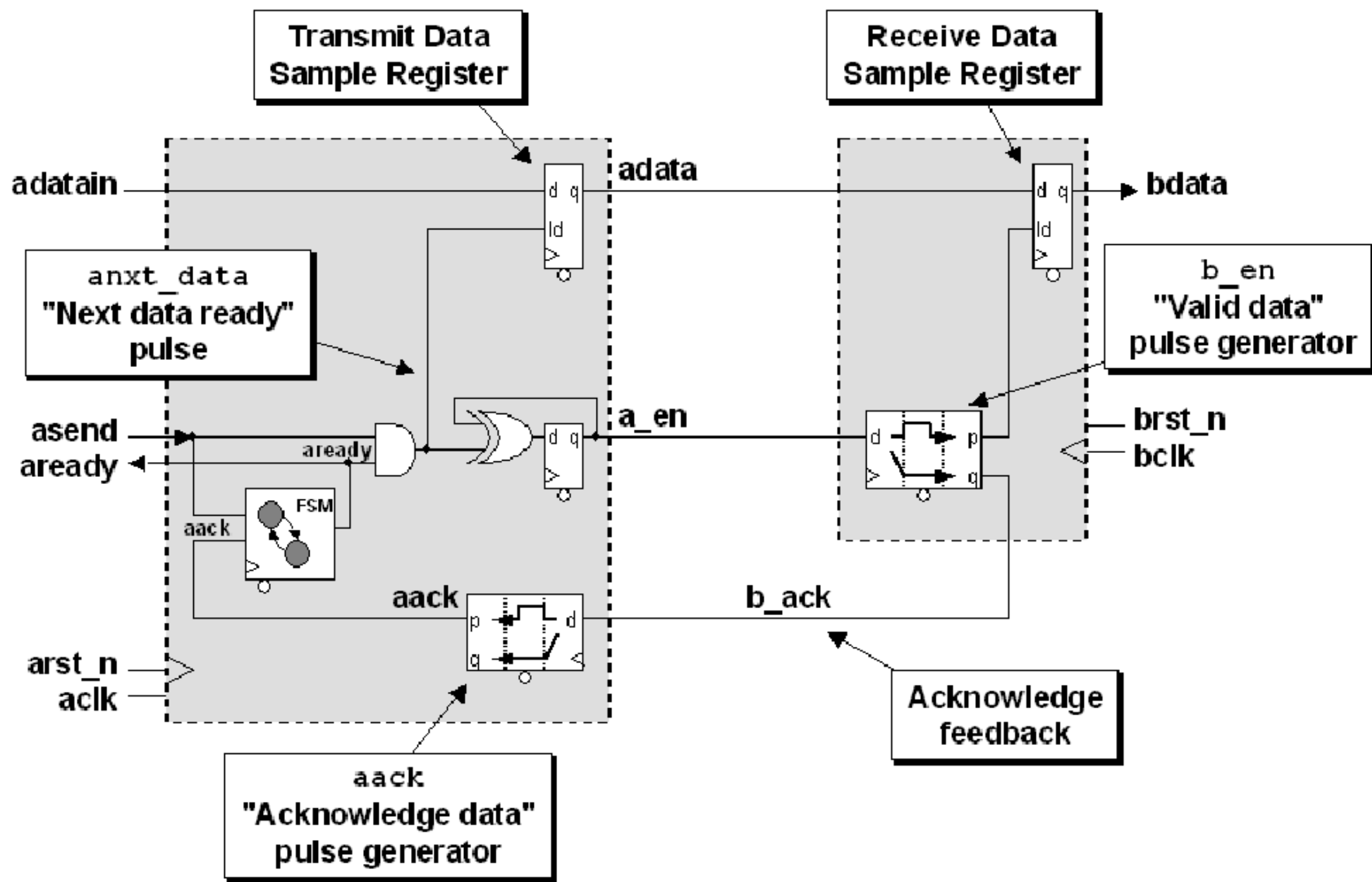
To avoid multi-bit CDC skewed sampling scenarios, I have classified multi-bit CDC strategies into three main categories:

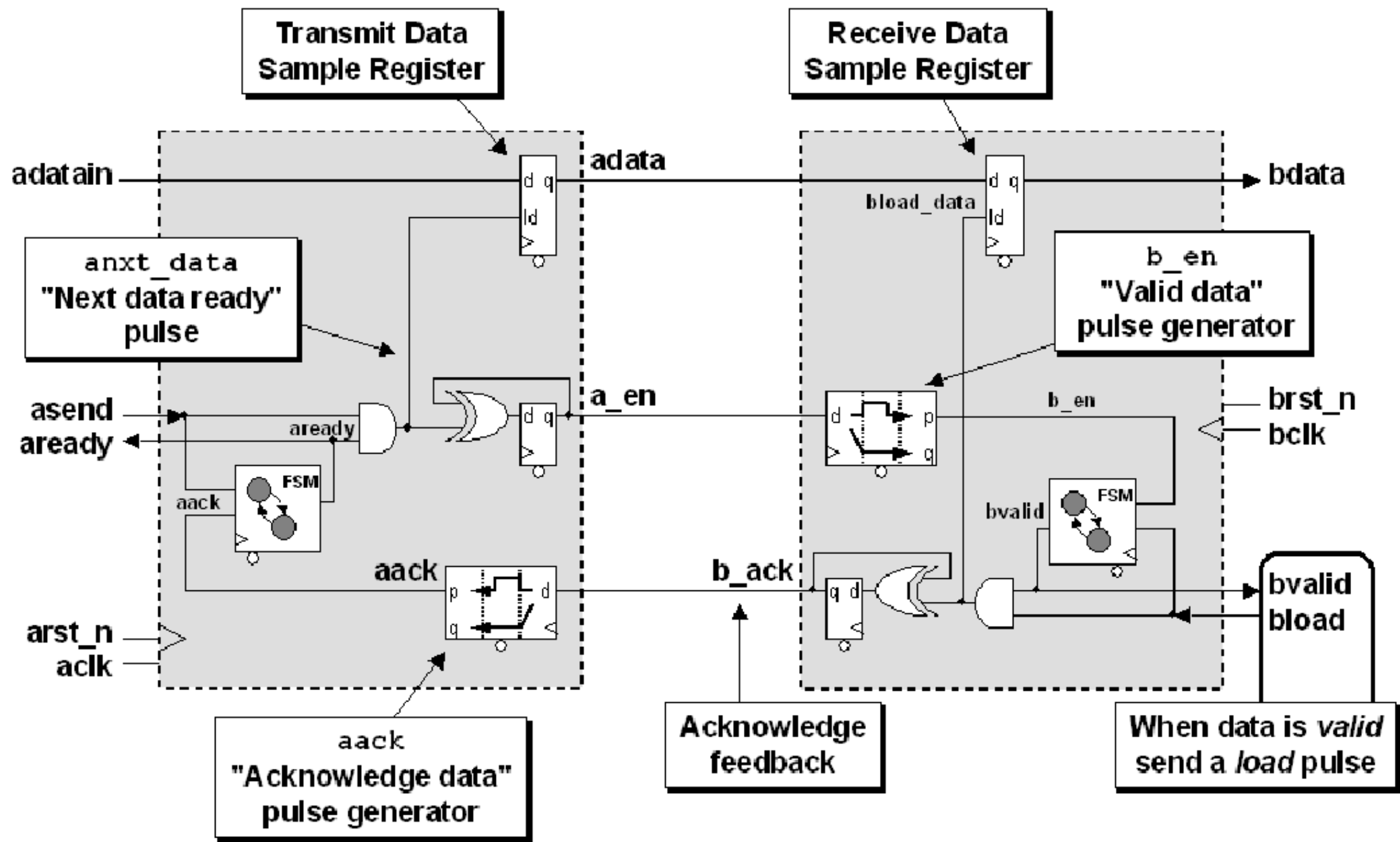
- (1) Multi-bit signal consolidation. Where possible, consolidate multiple CDC bits into 1bit CDC signals.
- (2) Multi-cycle path formulations. Use a synchronized load signal to safely pass multiple CDC bits.
- (3) Pass multiple CDC bits using gray codes.

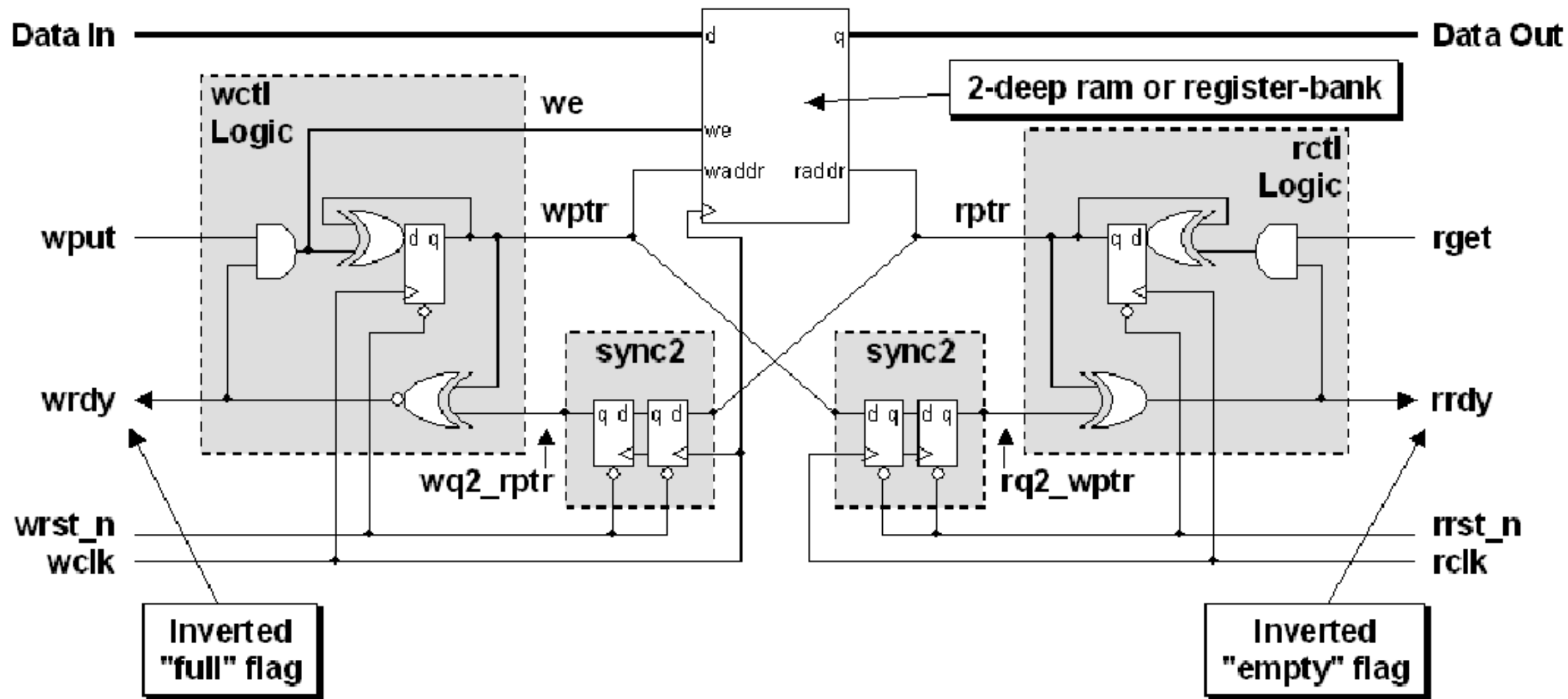












```

bin[0] = gray[3] ^ gray[2] ^ gray[1] ^ gray[0] ; // gray>>0
bin[1] = 1'b0 ^ gray[3] ^ gray[2] ^ gray[1] ; // gray>>1
bin[2] = 1'b0 ^ 1'b0 ^ gray[3] ^ gray[2] ; // gray>>2
bin[3] = 1'b0 ^ 1'b0 ^ 1'b0 ^ gray[3] ; // gray>>3

```

```

module bin2gray #(parameter SIZE = 4)
(output logic [SIZE-1:0] gray,
input logic [SIZE-1:0] bin);

assign gray = (bin>>1) ^ bin;
endmodule

```

```

gray[0] = bin[0] ^ bin[1];
gray[1] = bin[1] ^ bin[2];
gray[2] = bin[2] ^ bin[3];
gray[3] = bin[3] ^ 1'b0 ;

```

```

module gray2bin #(parameter SIZE = 4)
(output logic [SIZE-1:0] bin,
input logic [SIZE-1:0] gray);

always_comb
for (int i=0; i<SIZE; i++)
bin[i] = ^(gray>>i);
endmodule

```

Число	Бинарный код	Код Грея
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000