明德扬
科技.教育

点拨·FPGA之简易包文类型识别器练习设计思路

点透学习误区　拨出设计精髓

主 讲：潘文明

明德扬科教

明德扬
科技·教育

QQ群： 97925396

官　网：http://www.mdy-edu.com

淘　宝：http://mdy-edu.taobao.com

课程大纲

1. 功能要求

2. 设计思路(状态机设计)

3. 代码设计

一、功能要求

二、设计思路

1. 一个一个信号设计，逐个击破

2. 状态机的转移条件，要精确到1个时钟周期

三、设计思路—总体

数据包文格式

	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据

1. dout ： 输出数据
2. dout_sop：包文头指示信号。注意，此处sop是指pkt_type的首字节。
3. dout_eop：包文尾指示信号。此处eop是指fcs的最后一字节。
4. dout_vld：输出数据有效指示信号

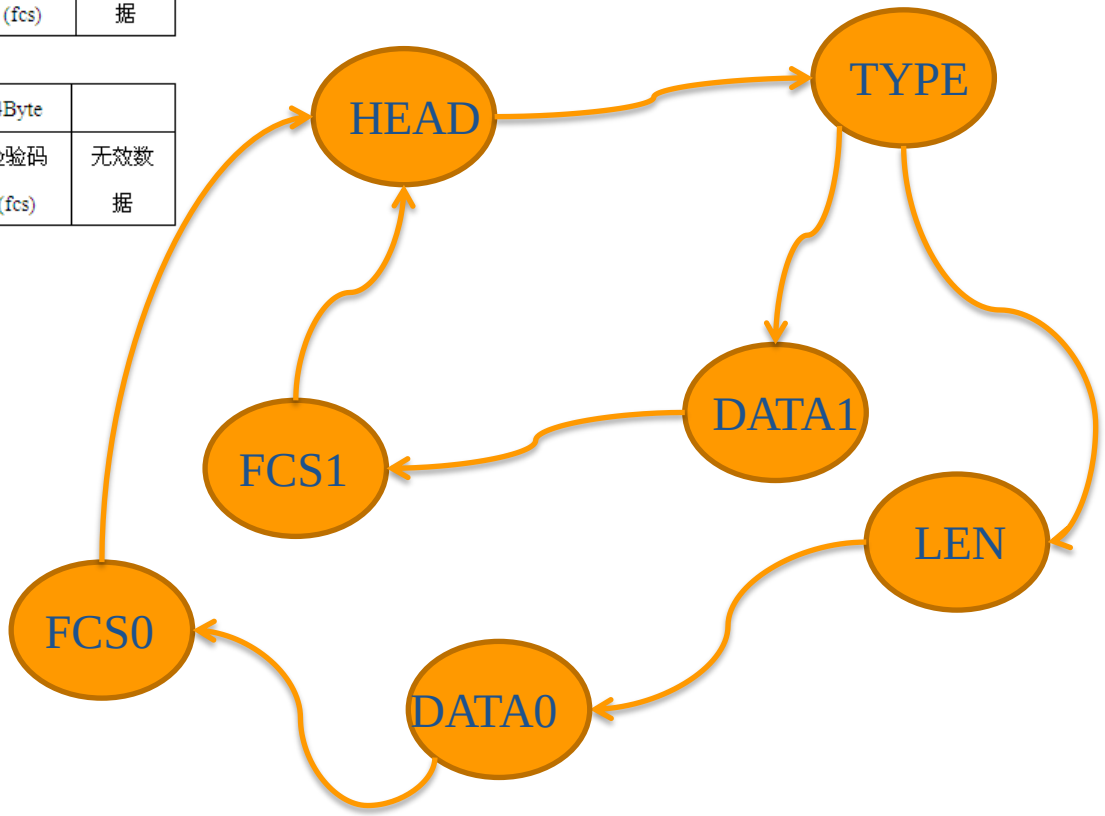
三、设计思路—总体

数据包文格式

	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据



LEN

三、设计思路—总体

数据包文格式

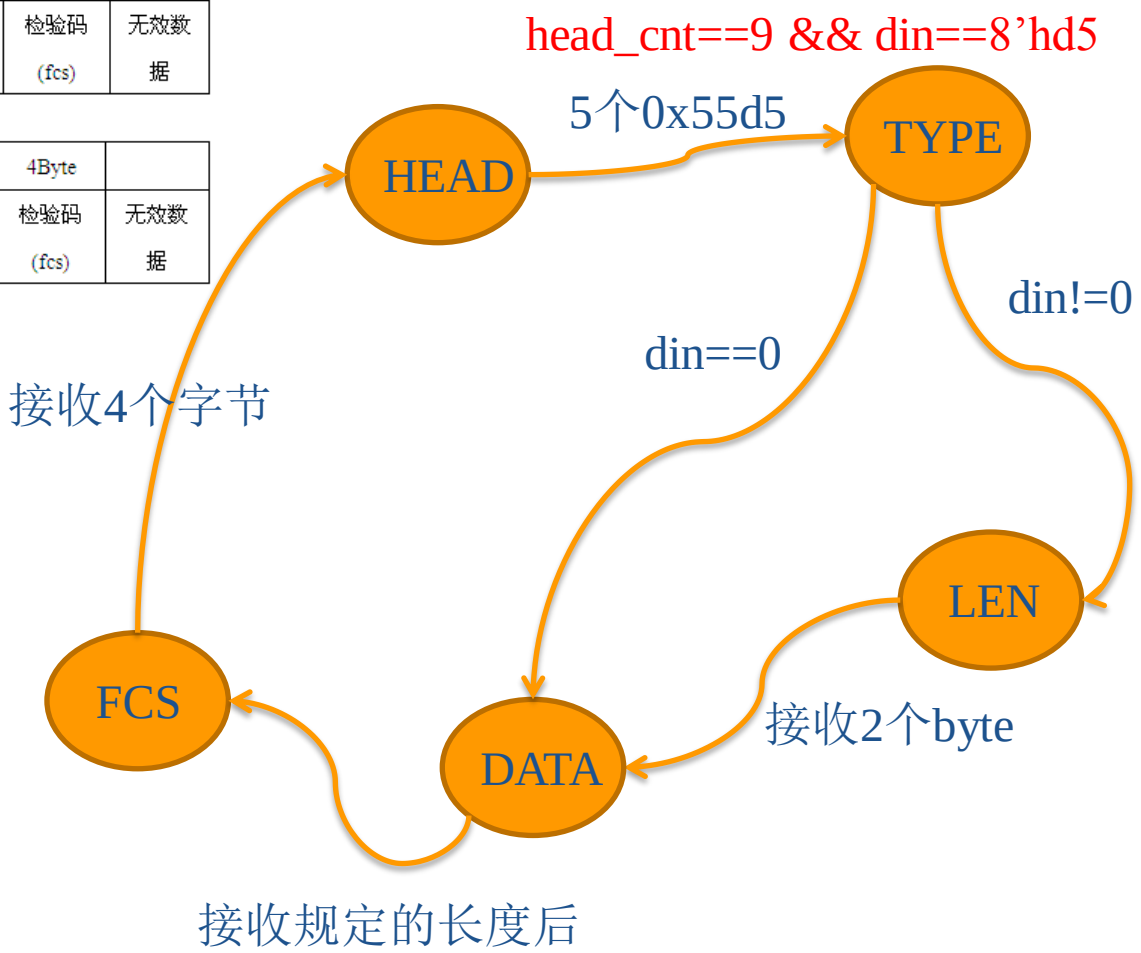
	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据

5个0x55d5
1. 要用计数器
2. 要区分55和d5

1. head_cnt
2. 统计正确的din个数



三、设计思路—总体

数据包文格式

	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

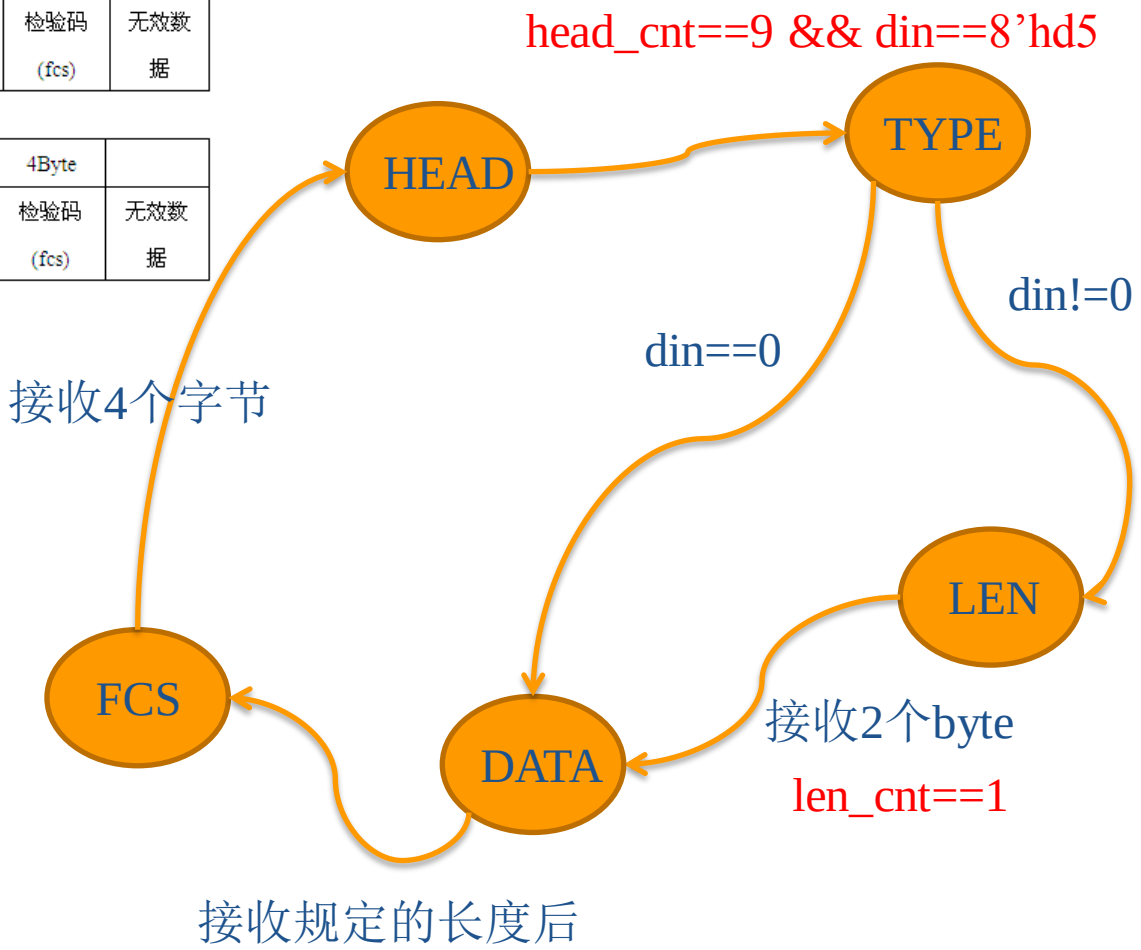
控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据

LEN:接收2个byte

1. 要用计数器

1. len_cnt



三、设计思路—总体

数据包文格式

	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

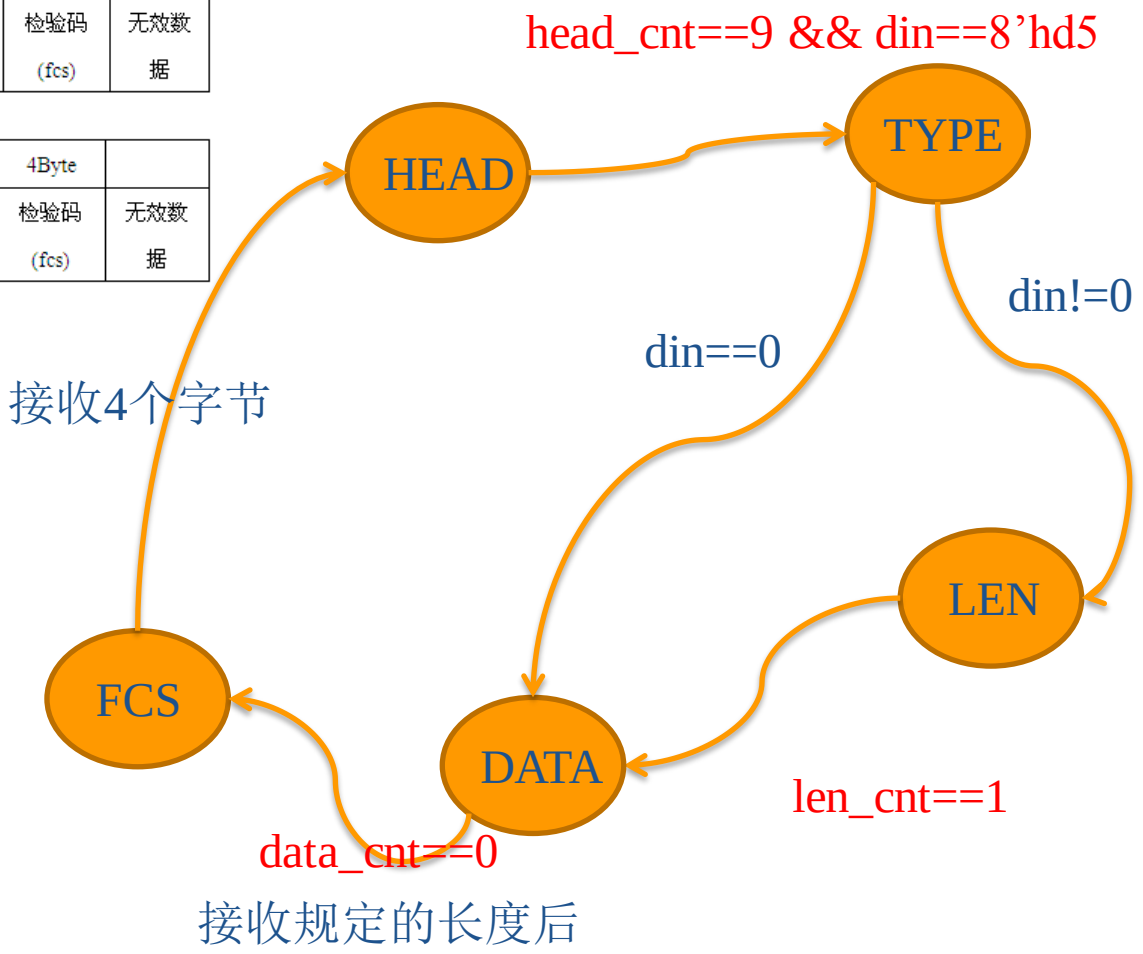
控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据

DATA:接收规定的长度

1. 要用计数器
2. 长度不定

1. data_cnt
2. 减一计数器



三、设计思路—总体

数据包文格式

	10Byte	1Byte	2Byte	0~65535Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	长度（length）	数据（playload）	检验码（fcs）	无效数据

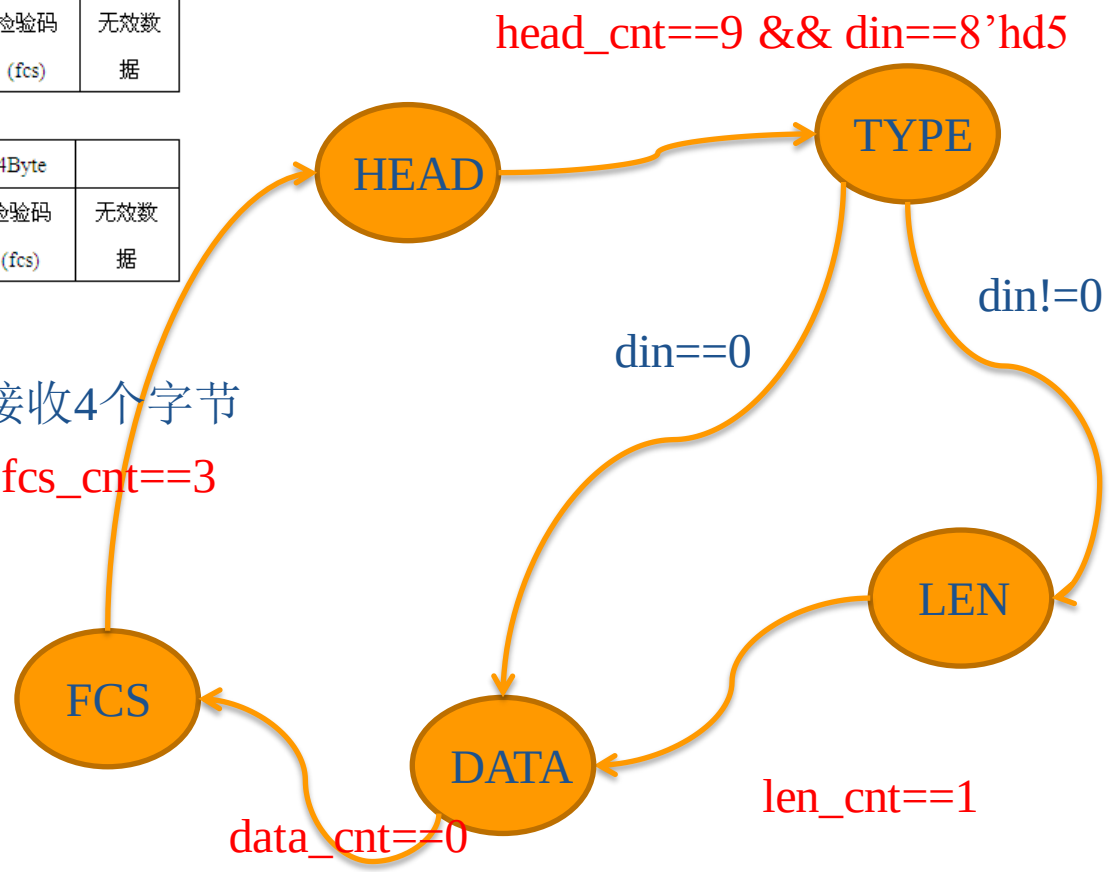
控制包文格式

	10Byte	1Byte	64 Byte	4Byte	
无效数据	包文头（head）	包文类型（pkt_type）	数据（playload）	检验码（fcs）	无效数据

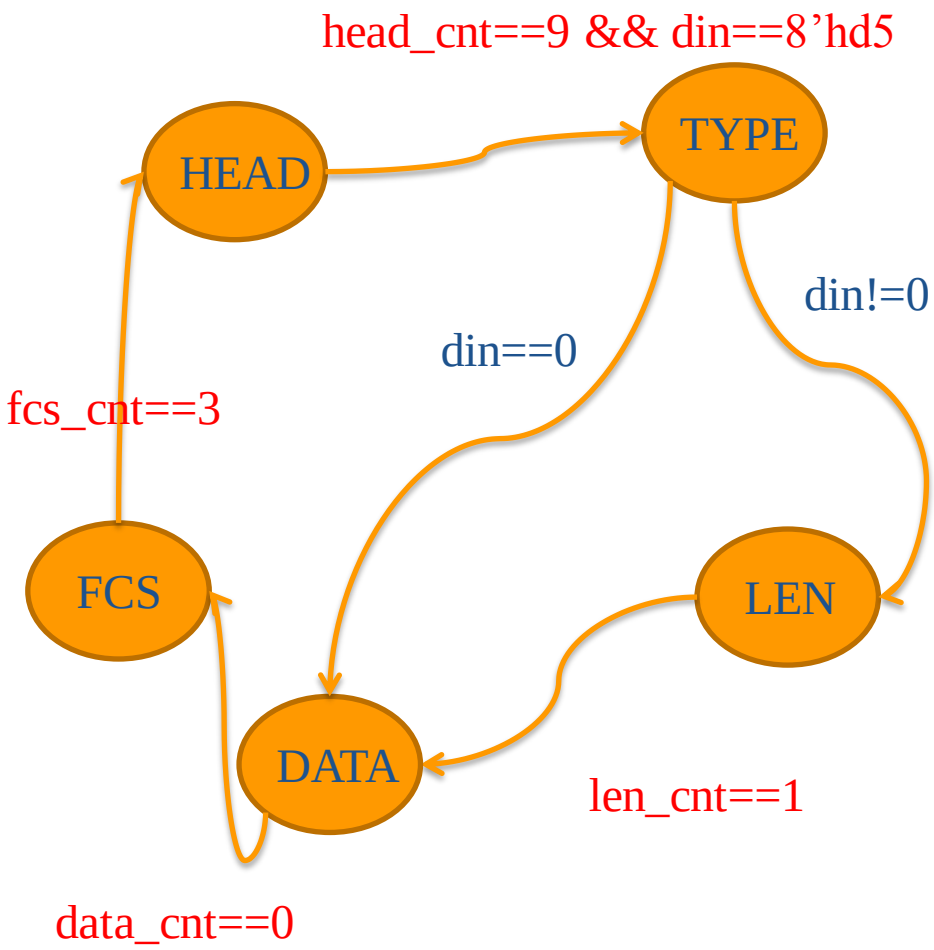
FCS:接收4个字节

1. 要用计数器

1. fcs_cnt



三、状态机设计




```
always
    ca
                DATA   :begin
                            if(data_cnt==0)begin
                                state_n = FCS;                    in
                            end
                            else begin
                                state_n = DATA;
                            end
                       end
                FCS    :begin
                            if(fcs_cnt==3)begin
                                state_n = HEAD;
                            end
                            else begin
                                state_n = FCS;
                            end
                       end
            endcase
        end
                      if(len_cnt==1)begin
                          state_n = DATA;
                      end
                      else begin
                          state_n = LEN;
                      end
                end
```

三、信号设计—head_cnt

1. 在HEAD状态时，用于统计5个0x55d5的个数
2. 由于一次输入1字节，且55！=d5，head_cnt定义为统计正确字节数。
3. 如何知道现在希望是55还是d5呢？假设head_flag信号，head_flag=0，希望是55；head_flag=1，希望是d5。

55	d5	55	d5	55	d5	55	d5
55	d5	55	d5	55	01	55	d5
55	d5	55	d5	55	55	d5	55
55	d5	55	d5	01	55	d5	55
55	d5	55	d5	d5	55	d5	55

三、信号设计—head_cnt

1. 在HEAD状态时，用于统计5个0x55d5的个数
2. 由于55！=d5，因此head_cnt定义为统计正确数据个数。
3. 如何知道现在希望是55还是d5呢？假设head_flag信号，head_flag=0，希望是55；head_flag=1，希望是d5。
4. 初值：0；加条件：HEAD状态、希望正确；正常结束条件：9且希望正确。
5. 希望不正确，如是55，则为1；如果非55，则为0。

三、信号设计—head_cnt

1. 初值：0；加条件：HEAD状态、希望正确；正常结束条件：9且希望正确。
2. 希望不正确，如是55，则为1；如果非55，则为0。

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        head_cnt <= 0;
    end
    else if(state_c==HEAD) begin
        if(head_flag==0) begin
            if(din==8'h55)begin
                head_cnt <= head_cnt + 1;
            end
            else begin
                head_cnt <= 0;
            end
        end
        else if(head_flag==1)begin
            if(din==8'hd5)begin
                if(head_cnt==9)begin
                    head_cnt <= 0;
                end
                else begin
                    head_cnt <= head_cnt + 1;
                end
            end
            else if(din==8'h55) begin
                head_cnt <= 1;
            end
            else begin
                head_cnt <= 0;
            end
        end
    end
    else begin
            head_cnt <= 0;
    end
```

三、信号设计—head_flag

1. 假设head_flag信号，head_flag=0，希望是55；head_flag=1，希望是d5。
2. head_flag=0时，如果是55，则head_flag=1；否则，head_flag=0；
3. head_flag=1时，如果是d5，则为0；如果是55，则为1；其他为0。

55	d5	55	d5	55	d5	55	d5
55	d5	55	d5	55	01	55	d5
55	d5	55	d5	55	55	d5	55
55	d5	55	d5	01	55	d5	55
55	d5	55	d5	d5	55	d5	55

三、信号设计—head_flag

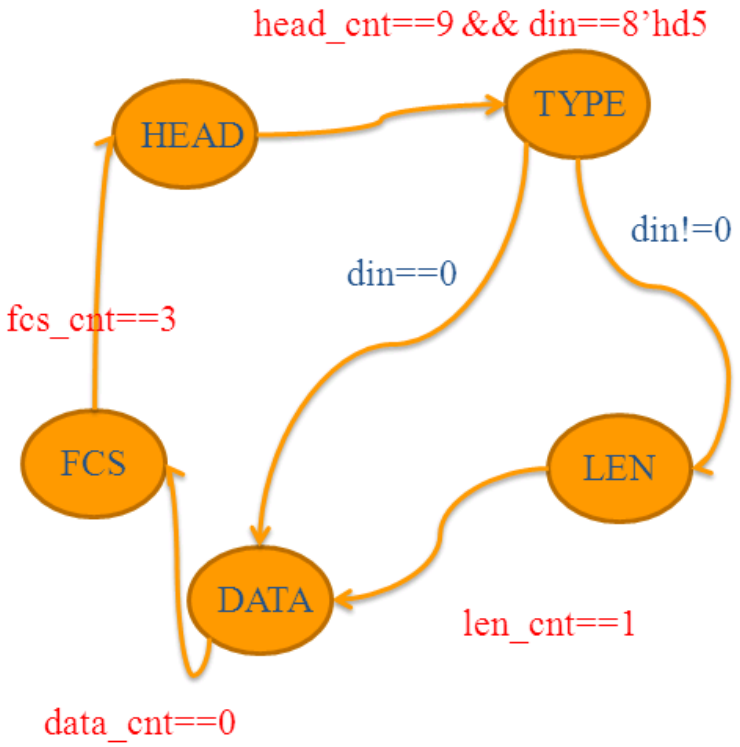
1. 假设head_flag信号，head_flag =0，希望是55；head_flag =1，希望是d5。
2. head_flag=0时，如果是55，则head_flag=1；否则，head_flag =0；
3. head_flag=1时，如果是d5，则为0；如果是55，则为1；其他为0。

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        head_flag <= 1'b0;
    end
    else if(state_c==HEAD) begin
        if(head_flag==1'b0)begin
            if(din==8'h55)
                head_flag <= 1'b1;
        end
        else begin
            if(din==8'h55)begin
                head_flag <= 1'b1;
            end
            else begin
                head_flag <= 1'b0;
            end
        end
    end
    else begin
        head_flag <= 1'b0;
    end
end
```

三、信号设计—len_cnt

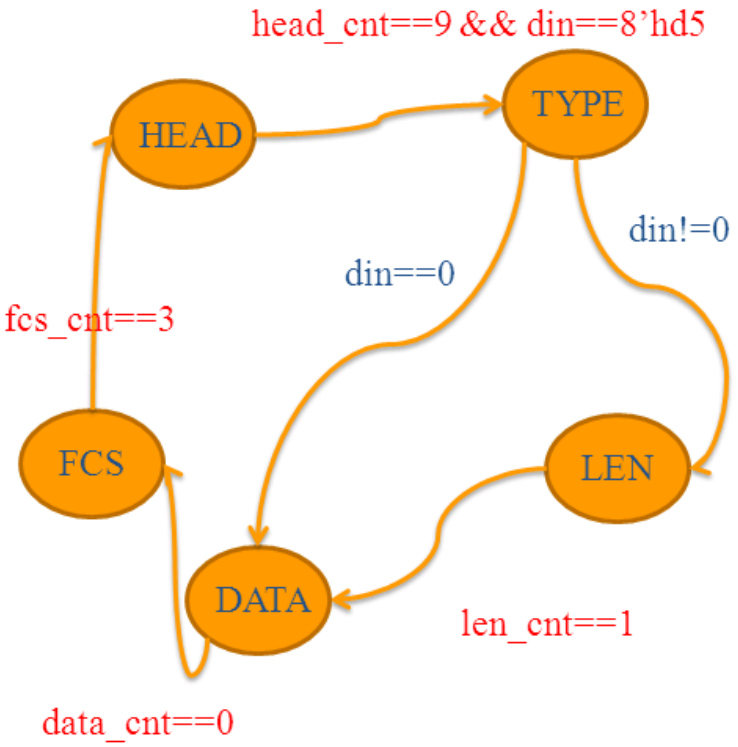
1. len_cnt，用于计算在LEN状态时间
2. 由于接收连个字节，因此长度=2
3. 初值：0；加1条件：LEN状态；结束：1

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        len_cnt <= 1'b0;
    end
    else if(state_c==LEN) begin
        len_cnt <= ~ len_cnt;
    end
    else begin
        len_cnt <= 1'b0;
    end
end
```



三、信号设计—data_cnt

1. data_cnt，用于计算DATA状态时要接收的字节数
2. 由于长度不固定，因此用减一计数器
3. 在DATA状态中减一，直至减为0。
4. 控制包文长度为64字节；
5. 数据包文长度从LEN状态中获取；

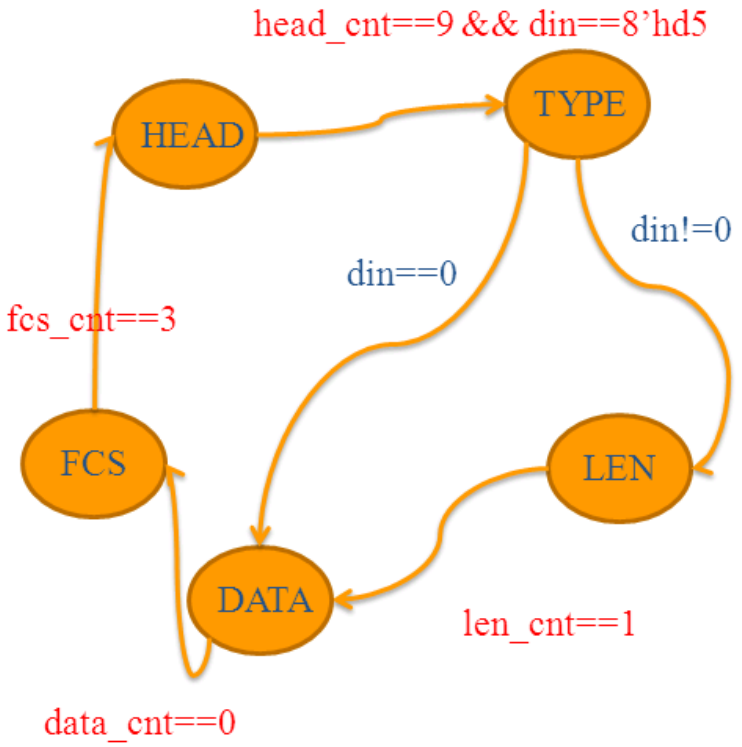


1. 在DATA状态，从N-1开始减一
2. TYPE跳到DATA时，获得（64-1）长度
3. 在LEN状态，获得（两次数据-1）

三、信号设计—data_cnt

1. 在DATA状态，从N-1开始减一
2. TYPE跳到DATA时，获得（64-1）长度
3. 在LEN状态，获得（两次数据-1）

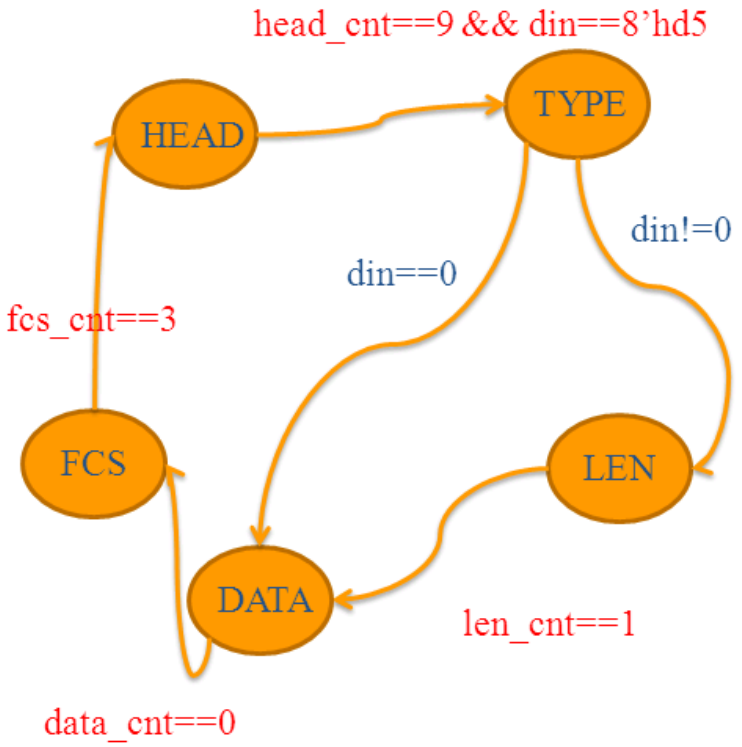
```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        data_cnt <= 0;
    end
    else if(state_c==TYPE && din==0) begin
        data_cnt <= CTRL_PKT_LEN-1;
    end
    else if(state_c==LEN)begin
        if(len_cnt==0)begin
            data_cnt <= {data_cnt[7:0],din};
        end
        else begin
            data_cnt <= {data_cnt[7:0],din}-1;
        end
    end
    else if(data_cnt!=0)begin
        data_cnt <= data_cnt - 1;
    end
end
```



三、信号设计—fcs_cnt

1. fcs_cnt用来计算在FCS状态时间
2. FCS固定为4
3. 初值：0；加1条件：FCS状态；结束：3

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        fcs_cnt <= 0;
    end
    else if(state_c==FCS)begin
        if(fcs_cnt==3)
            fcs_cnt <= 0;
        else
            fcs_cnt <= fcs_cnt + 1;
    end
end
```



三、信号设计—dout,dout_vld

1. pkt_type到fcs有效

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        dout <= 0;
    end
    else begin
        dout <= din;
    end
end

always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        dout_vld <= 1'b0;
    end
    else if(state_c != HEAD) begin
        dout_vld <= 1'b1;
    end
    else begin
        dout_vld <= 1'b0;
    end
end
```

三、信号设计—dout_eop,dout_sop

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        dout_sop <= 1'b0;
    end
    else if(state_c==TYPE) begin
        dout_sop <= 1'b1;
    end
    else begin
        dout_sop <= 1'b0;
    end
end
```

```
always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        dout_eop <= 1'b0;
    end
    else if(state_c==FCS && fcs_cnt==3) begin
        dout_eop <= 1'b1;
    end
    else begin
        dout_eop <= 1'b0;
    end
end
```

明德扬科教

明德扬
科技·教育

QQ群： 97925396

官　网：http://www.mdy-edu.com

淘　宝：http://mdy-edu.taobao.com

Thank You !