震撼！FPGA实现负反馈控制纯数字锁相环！

2018-03-06 包春 / 胡建东QQ:1758931664

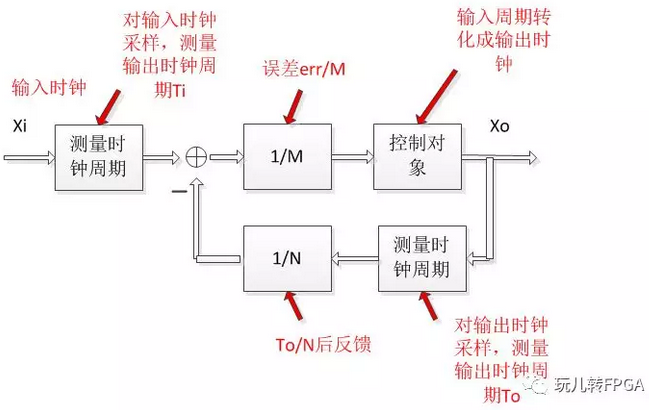
转载至微信公众号“[玩儿转FPGA](https://mp.weixin.qq.com/s?__biz=MzU0NDcxMDY2Mg==&mid=100000074&idx=1&sn=2342e460ec7fb80573e56772361820b6&chksm=7b7948234c0ec135d9a21f958fbd4f9590bb72ede0ea8035204affa895c9f8b57f6be16e6ba6&scene=18&key=d8ea7e7928bbcadda7e4ab4e16e674410e714750f8a945f7bb7d2e455d6be1e416112b6a0bf32369cf53f96892eaf0aaa72e53ca4ac83e2e34e14d1fafae5edb306db50fe54ae621401db2a4c3e89844&ascene=1&uin=MTI0MTAzODc2Mw%3D%3D&devicetype=Windows+7&version=6206014b&lang=zh_CN&pass_ticket=46OTQYTJS2pZvXmJ9n5dEK%2BUcGh6aek5G7KdApZOpKYCIYPC1RBn3L6EHkkitZ4X&winzoom=1)”原文链接：[震撼！负反馈控制实现纯数字锁相环！](http://mp.weixin.qq.com/s/f5gqryXwAqZChK-t2atLyw) 或者：http://mp.weixin.qq.com/s/f5gqryXwAqZChK-t2atLyw 很多小伙伴说控制原理真\*\*难学，小编深有同感。于是发布此文帮助在控制原理里面不停挣扎的小伙伴们，帮你们拨开迷雾。此文专门告诉你怎么实现数字负反馈控制，一切高深理论均不涉及！

首先我们做一个模块，输入信号T，代表输出时钟周期，输出时钟周期严格等于T，对于熟悉FPGA的小伙伴应该很容易。下面是小编的代码。简单的说就是建立一个状态机，对输入周期参数在产生时钟的每个周期，第一时刻更新值，然后进入下一个状态计数，计数到和输入周期一致的时候又回到初始状态。

1. module clk\_gen(
2. input               clk,
3. input     [15:0]    T  ,//it can not be 0!
4. output reg          clk\_gen
5. );
7. reg       [15:0]    cnt\_div = 0;
8. reg       [15:0]    period = 0;
9. reg       [ 1:0]    now\_s  = 0;
10. reg       [ 1:0]    next\_s = 0;
12. always@(posedge clk)
13. begin
14. now\_s <= next\_s;
15. end
17. always@(\*)
18. begin
19. **case**(now\_s)
20. 2'h0  : next\_s = (T > 0) ? 1 : 0;
21. 2'h1  : next\_s = (cnt\_div < (period>>1)) ? 1: 3;//跑前半周期
22. 2'h3  : next\_s = (cnt\_div < (period-1)) ? 3: 0; //跑后半周期
23. **default**: next\_s =0;
24. endcase
25. end
27. always@(posedge clk)
28. begin
29. **case**(next\_s)
30. 2'h0  : begin period  <= T;                         cnt\_div <= 0;           clk\_gen <= 1'b0; end
31. 2'h1  : begin period  <= (now\_s == 0) ? T : period; cnt\_div <= cnt\_div + 1; clk\_gen <= 1'b0; end//刷新T
32. 2'h3  : begin                                       cnt\_div <= cnt\_div + 1; clk\_gen <= 1'b1; end
33. **default**: begin period  <= 10;                       cnt\_div <= 0;           clk\_gen <= 1'b0; end
34. endcase
35. end
36. endmodule

上面这个模块就是我们的控制对象，也就是说，我们要想办法让他的时钟频率Fo = 系统给定时钟频率Fi\*N，也就是说: 输出时钟周期To = 系统时钟周期Ti/N。

小编采用如下环路实现。从中可以看出，小编的方法反馈控制的是周期，所以一切参数都转化到周期上，输入和反馈，输出都是通过周期控制。从图中可以得出结论输出时钟周期To=N/(M\*N+1)



输入时钟要求稳定，并且比系统时钟慢的越多越好，因为这样测量更准确。下面就是我们测量时钟周期的代码。反馈非常重要，因为他涉及到整个反馈环路的精度和运作效率。除以N的部分我们在另一个模块实现，因为测量时钟周期的模块是相对独立的输入和反馈都要用到，独立出来可以加强代码可重用性。这段代码就是在每个输入时钟上升沿电脑时候更新周期值。以保证周期的准确性。

1. module figure\_T(
2. input                clk,
3. input                clk\_in,
4. output reg[15:0]     T = 0
5. );
7. localparam           PE = 2'b01;
9. reg       [ 1:0]     clk\_buf = 0;
10. reg       [15:0]     cnt\_T = 0;
11. reg                  state = 0;
13. always@(posedge clk) begin clk\_buf <= {clk\_buf[0],clk\_in}; end
15. always@(posedge clk)
16. begin
17. **case**(state)
18. 0:begin state <= (clk\_buf == PE) ? 1 : 0; cnt\_T <= (clk\_buf == PE) ? 1 : 0; end
19. 1:begin state <= (clk\_buf == PE) ? 0 : 1; cnt\_T <= cnt\_T + 1; T <= (clk\_buf == PE) ? cnt\_T : T; end
20. **default**: state <= 0;
21. endcase
22. end
23. endmodule

控制对象和输入，反馈参数产生都已经好了，剩下的就是按照控制框图搭建反馈环路。代码如下图，代码中的Ti是在顶层产生的，输入时钟被测量后才把Ti给到环路输入。第34行是将反馈的时钟周期乘以反馈增益1/N，35行是将反馈误差乘以开环增益1/M。至此整个环路已经搭建完成！

1. module feedback(
2. input                clk,
3. input     [15:0]     Ti,
4. output               clk\_out
5. );
7. parameter            M  = 10;//参数
8. parameter            N  = 10;//参数
10. wire      [15:0]     T\_clk\_out;
11. reg       [15:0]     T;
12. reg       [15:0]     err;
13. //反馈误差
14. always@(posedge clk) begin err <= Ti - (T\_clk\_out/N); end
15. always@(posedge clk) begin T   <= err/M; end
16. //被控制对象
17. clk\_gen u\_clk\_gen(
18. .clk    ( clk     ),
19. .T      ( T       ),
20. .clk\_gen( clk\_out )
21. );
22. //输出时钟周期反馈
23. figure\_T uT(
24. .clk   ( clk       ),
25. .clk\_in( clk\_out   ),
26. .T     ( T\_clk\_out )
27. );
28. endmodule

这里附上小编顶层代码，这里从30行开始的模块就是产生上文说的Ti用的，输入时钟直接测量后，变成是种病周期T输入到feedback模块。

1. module dll(
2. input          clk,
3. input          clk\_in,
4. output         clk\_out
5. );
7. wire    [15:0] T;
9. figure\_T uT(
10. .clk   ( clk ),
11. .clk\_in( clk\_in ),
12. .T     ( T )
13. );
14. //
15. feedback #(
16. .M      ( 9  ),
17. .N      ( 1  )
18. ) u\_fb(
19. .clk    ( clk ),
20. .Ti     ( T   ),
21. .clk\_out( clk\_out )
22. );
24. endmodule

那么实际效果咋样呢，能工作吗？工作起来是啥样子？小编接下来就测试给大家看。测试代码如下，输出时钟周期在反馈的代码里面找，输入时钟周期是给定的，这两个找出来对比看看与理论差多远就知道了，这里M = 10,N=10。

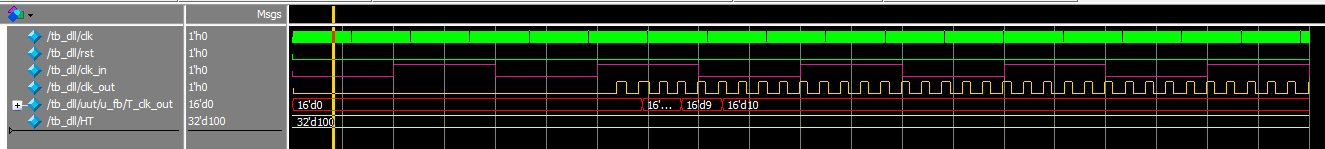
1. module tb\_dll;
2. parameter     HT = 100;
3. // Inputs
4. reg clk;
5. reg rst;
6. reg clk\_in;
8. // Outputs
9. wire clk\_out;
10. wire [15:0] Tclk\_out;
11. wire [15:0] Tclk\_in;

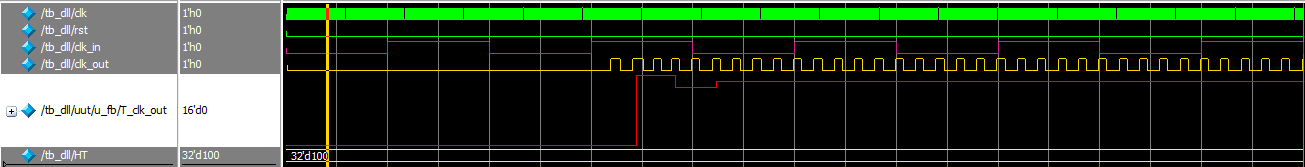
14. initial begin
15. // Initialize Inputs
16. clk = 0;
17. rst = 0;
18. clk\_in = 0;
20. // Wait 100 ns for global reset to finish
21. #100;
23. // Add stimulus here
25. end
27. always begin clk    = 0;#1   clk    = 1;  #1;  end//主时钟产生
28. always begin clk\_in = 0;#HT  clk\_in = 1;  #HT; end//输入时钟产生
29. // Instantiate the Unit Under Test (UUT)
30. dll uut (
31. .clk    ( clk    ),
32. .clk\_in ( clk\_in ),
33. .clk\_out( clk\_out)
34. );

37. endmodule

仿真结果如下，看图1可知输出时钟周期稳定后为To = 10\*2ns，输入时钟为Ti=100\*2ns，所以To/Ti = 0.1。而理论值为N/(M\*N+1)=10/(10\*10+1)=0.099。

误差err = |0.1-0.099|/0.1=1% !





我们看看下图的波形，输出时钟周期的震荡是不是很像连续系统的二阶震荡环节的阶跃响应？只不过超调不大，震荡周期比较少。和控制原理书上很近似（看看离散控制部分会更觉得近似）。小编今天讲到这里，有任何疑问请对公众号发消息“提问”，就可以获得小编联系方式并提问。

**下一篇小编会精心讲解反馈控制离散化的一些知识，从理论上破解大家的疑惑，让大家知其然也知其所以然！！**