Rotary Encoder Circuit Drive Based on STEP FPGA

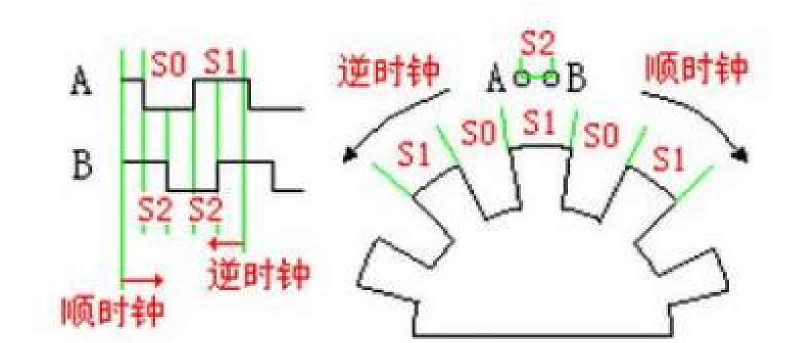
In this section, we will use FPGA to drive the principle and driving method of the EC11 rotary encoder on the backplane.

====Hardware description====

Rotary encoder is a device used to measure speed. Because of its humanized operation, it is used in more and more electronic devices. There are many categories of rotary encoders:

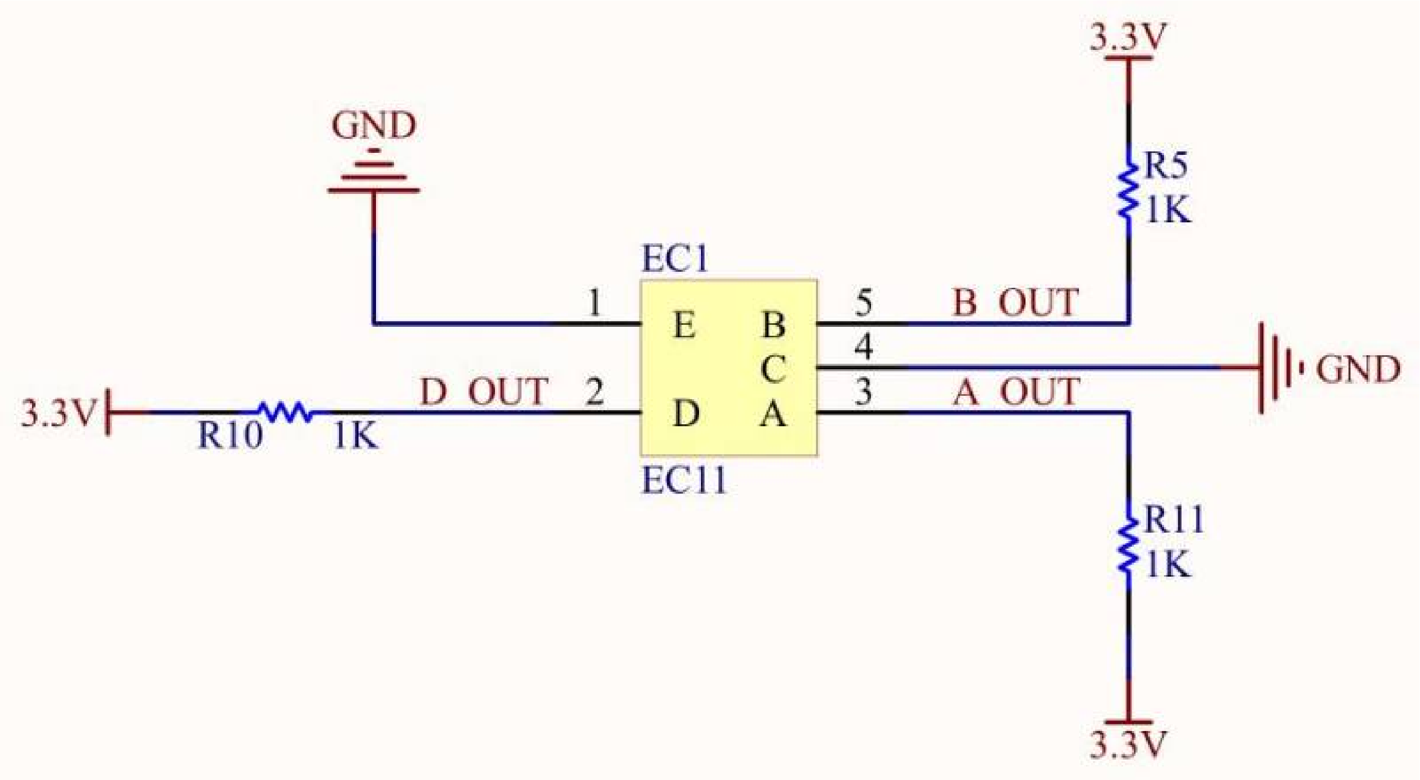
- According to the working principle of the encoder, it can be divided into: photoelectric type, magnetoelectric type and contact brush type.
- There are two types of code disc engraving methods: incremental and absolute.

Regarding the difference between the above types of encoders, please check the information yourself, so I won't introduce them here.
The EC11 rotary encoder integrated on our STEP-BaseBoard backplane is an incremental electric shock brush encoder. Its working principle is as follows: As shown in the figure above, when rotating clockwise, the A signal advances the B signal by 90 degrees. When the clock hand rotates, the B signal advances the A signal by 90 degrees. When the FPGA receives the A and B signals of the rotary encoder, it can determine the rotation direction of the encoder based on the combination of the states of A and B. In the program design, we can detect the A and B signals, detect the edge of the A signal and the state of the B signal,



- When the A signal is at the rising edge, the B signal is low, or when the A signal is at the falling edge, the B signal is high, which proves that the current encoder is rotating clockwise
- When the A signal is at the rising edge, the B signal is high, or when the A signal is at the falling edge, the B signal is low, which proves that the current encoder is rotating counterclockwise

The actual circuit connection of this design is as follows:



====Verilog code====

```
// ---------------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ---------------------------------------------
// Module: Encoder
/ /
// Author: Step
//
// Description: Driver for rotary encoder
//
// Web: www.stepfapga.com
//
// ------------------- --------------------------------------------------
/ / Code Revision History:
// -------------------------------------------- -------------------------
// Version: |Mod. Date: |Changes Made:
// V1.0 |2016/04/20 |Initial ver
// ---------------------------------------------------- --------------------
module Encoder
 (
input                                   clk_in ,                        //System clock
input                                   rst_n_in ,              //System reset, low effective
input                                   key_a ,                         //Rotary encoder A tub
e Pin
input                                   key_b ,                         //Rotary encoder B pin
input                                   key_ok ,                        //Rotary encoder D pin
output  reg                             Left_pulse ,            //Left rotation pulse output
output  reg                             Right_pulse ,   //Right rotation pulse output
output                                  OK_pulse                 //Press pulse Output
) ;

localparam                              NUM_500US       =      6 _000 ;

reg                             [ 12 : 0 ]      cnt ;
//The counter period is 500us, and the control key value sampling frequency is
always @ ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in ) cnt <=  0 ;
        else  if ( cnt >= NUM_500US - 1 ) cnt <=  1'b0 ;
        else cnt <= cnt +  1'b1 ;
end

reg                             [ 5 : 0 ]       cnt_20ms ;
reg                                             key_a_r , key_a_r1 ;
reg                                             key_b_r , key_b_r1 ;
reg                                             key_ok_r ;

//Do simple debounce operations for the A, B, and D pins.
//If the requirements for the rotary encoder are relatively high, it is recommended to perform
strict debounce processing on the output of the rotary encoder before doing the rotary encoder
Drive
//For the input buffer of the rotary encoder, eliminate the metastability and delay the latch
always @ ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! Rst_n_in )  begin
```

```
                key_a_r         <=      1'b1 ;
                key_a_r1        <=      1'b1 ;
                key_b_r         <=      1'b1 ;
                key_b_r1        <=      1'b1 ;
                cnt_20ms        <=      1'b1 ;
                key_ok_r        <=      1'b1 ;
        end else  if ( cnt == NUM_500US - 1 )  begin
                key_a_r         <=      key_a ;
                key_a_r1        <=      key_a_r ;
                key_b_r         <=      key_b ;
                key_b_r1        <=      key_b_r ;
                if ( cnt_20ms >=  6'd40 )  begin        //Use 20ms for the button D signal Per
iodic sampling method, 40*500us = 20ms
                        cnt_20ms <=  6'd0 ;
                        key_ok_r <= key_ok ;
                end  else  begin
                        cnt_20ms<= cnt_20ms +  1'b1 ;
                        key_ok_r <=     key_ok_r ;
                end
        end
end

reg                                             key_ok_r1 ;
//Latch the button D signal with delay
always @ ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in ) key_ok_r1 <=  1'b1 ;
        else key_ok_r1 <= key_ok_r ;
end

wire    A_state         = key_a_r1 && key_a_r && key_a ;       //Rotary encoder A signal high
level state detection
wire    B_state         = key_b_r1 && key_b_r && key_b ;       //Rotary encoder B signal high
level state detection
assign  OK_pulse        = key_ok_r1 &&  ( ! key_ok_r ) ;               //D signal falling edg
e detection of rotary encoder

reg                                             A_state_reg ;
//delay latch
always @ ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in ) A_state_reg <=  1'b1 ;
        else A_state_reg <= A_state ;
end

//Rising and falling edge detection of the rotary encoder A signal
wire    A_pos    =  ( ! A_state_reg )  && A_state ;
wire    A_neg    = A_state_reg &&  ( ! A_state ) ;

//Judging the operation of the rotary encoder by the combination of the edge of the rotary enc
oder A signal and the level status of the B signal, and output the corresponding pulse signal
always @ ( posedge clk_in or  negedge rst_n_in ) begin
        if ( ! Rst_n_in ) begin
                Right_pulse < =  1'b0 ;
                Left_pulse <=  1'b0 ;
```

```
        end  else  begin
                if ( A_pos && B_state ) Left_pulse <=  1'b1 ;
                else  if ( A_neg && B_state ) Right_pulse <= 1'b1 ;
                else  begin
                        Right_pulse <=  1'b0 ;
                        Left_pulse <=  1'b0 ;
                end
        end
end

endmodule
```

====Summary====

This section mainly explains the working principle and software design of the rotary encoder. You need to create a project by yourself while mastering it, and generate the FPGA configuration file loading test through the entire design process.
If you are not familiar with the use of Diamond software, please refer to here: Use of Diamond .

====Related Information====

Use STEP-MXO2 second generation rotary encoder applications: download link will be updated, the subsequent use of STEP-MAX10 rotary encoder applications: subsequent download link will be updated