

Quadrature Encoder Receiver Module

An Implementation on FPGA

Dr.Varodom Toochinda

<http://www.controlsystemslab.com>

July 2009

In this document we discuss the design and implementation of quadrature encoder receiver module on FPGA using Verilog. The unit receives the A and B pulses that are 90 degree out of phase and generates the up and down pulses depending on whether A leads B, or vice versa. The output pulses give 4 times resolution of the encoder lines. It has been verified by simulation and tested on Xilinx Spartan 3.

Servomotor and Encoder Fundamentals

There are many sources that explain servomotor and incremental encoder operations. Here we briefly describe only the essence. Servomotors come in many types, from some simplest brushed DC (Figure 1) preferred by hobbyists, to some sophisticated, high-performance AC brushless servos used in industrial CNCs and robots(Figure 2).



Figure 1: Brushed DC servomotors (Yasakawa Minertia Series)



Figure 2 AC Servomotors and drives (Panasonic)

Assuming the servo systems operated in velocity mode, when you send in analog signals within ± 10 V range, the motors rotate with speeds corresponding to the applied input voltages. The incremental encoders generate the quadrature A,B, and index Z signals with frequencies related to the RPMs of motors. As shown in Figure 3, the encoder disk has slots that pass the light beams from LEDs to receiving devices. This generates 3 main signals A, B, and Z. A and B are called quadrature because they have 90 degree phase differences. When the motor turns one direction, say, clockwise, the signal A leads B 90 degree. When it turns counter-clockwise, B leads A 90 degree. With this construction we could construct a circuit that detects the direction of rotation, in addition to the shaft position and speed. The overall encoder receiver unit is shown in Figure 4.

In real industrial environment things are not perfect. There are evils like noise that contaminates the signal. So the signals A, B, Z are sent with their complements A/, B/, and Z/ that are 180 degree out of phase. When noise gets into these signals, the noise component in each signal tends to be similar, so when passed through differential amps the noise are rejected. This is called RS-422 standards. This is the duty of the leftmost block in Figure 4.

After that, the signals pass through another noise rejection unit known as a digital filter, before going to the decoder unit that generates the up/down pulses. The last section is the counter that accumulates the pulses. The content in this counter represents the position of a machine that has long linear movement, for example, an axis of a CNC milling machine.

In this technical journal, we discuss only the digital filter and the x4 decoder in Figure 4.

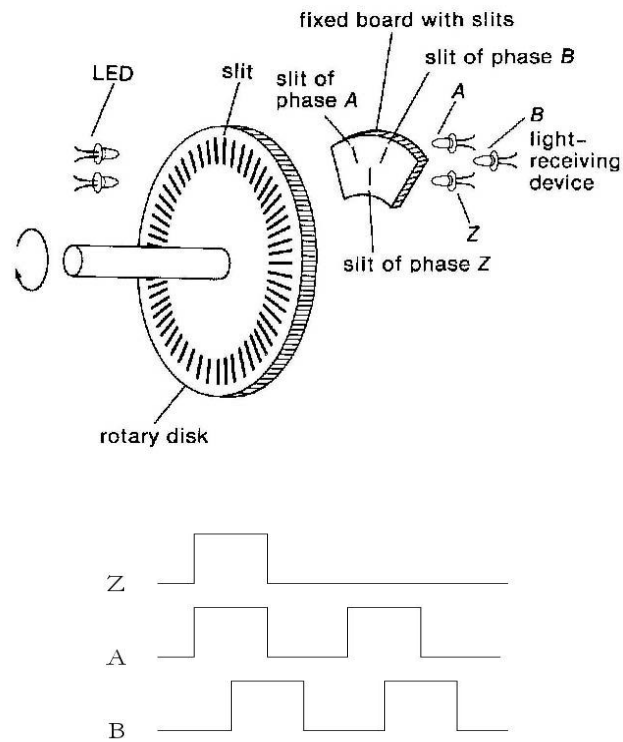


Figure 3: incremental encoder and quadrature signals

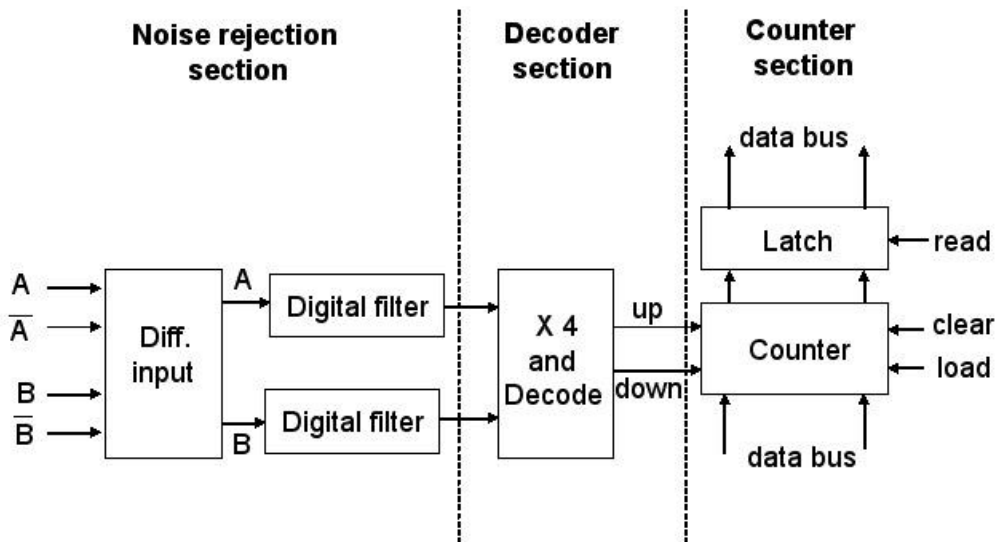


Figure 4: Overall encoder receiver unit

Digital Filter Unit

In case one would like to implement the digital filter using discrete ICs or schematic file, the circuit in Figure 5 can be used. Its operation is simple. At some moment in time, the encoder signal, either high or low, has to pass through 4 stages of D FF before it could be considered valid. When the signal is contaminated, the noise component is more like a spike. It could make the signal change logic abruptly. But that intermittent would disappear at the output, since it couldn't survive 4 clock cycles. Of course, the clock frequency has to be properly considered, relative to the highest frequency of the encoder signal. The Verilog version of this circuit is provided in Listing 1.

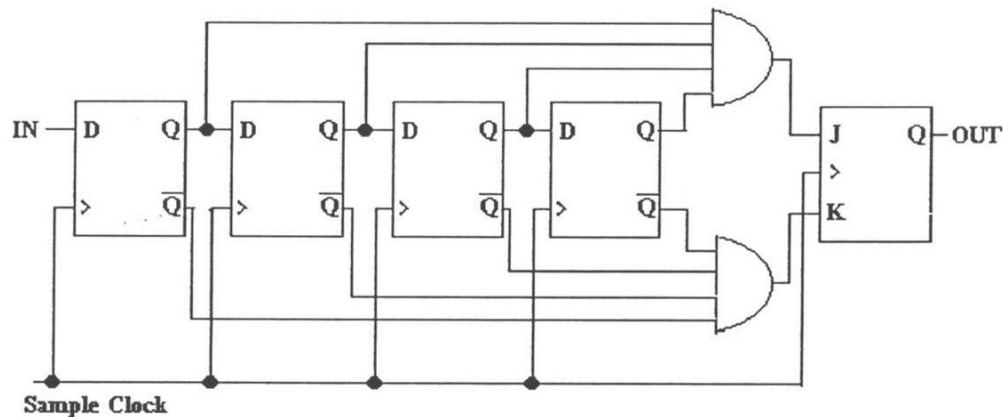


Figure 5: Digital Filter Circuit

```
module dfilter(clk, D, Q);  
    input clk;  
    input D;  
    output reg Q;  
    reg [3:0] dfilter4;  
    parameter valid0 = 4'b0000, valid1 = 4'b1111;  
    always @ (posedge clk)  
    begin
```

```

dfilter4 <= {dfilter4[2:0], D};

case(dfilter4)

    valid0: Q <= 0;

    valid1: Q <= 1;

    default: Q <= Q;           // hold value

endcase

end

endmodule

```

Listing 1: Verilog code for digital filter

X 4 Decoder (Design A)

The decoder unit is designed such that when signal A leads B, and A lags B, pulses are generated at the U (up) , and D (dn) outputs, respectively. We can use standard state-transition graph (STG) technique to realize a circuit, as shown in Figure 6. This is a Mealy machine STG. It consists of 4 states and shows how state/output changes with input. We can use 2 bits to represent the states. Table 1 shows the relationship between states, next states, and outputs. From Table 1 we can then use Karnaugh map to reduce the Boolean equations as shown in Figure 7. The results are the equations (1) – (4) below.

$$q_0^+ = A \oplus B \quad (1)$$

$$q_1^+ = B \quad (2)$$

$$U = A'Bq_1' + A'q_1'q_0 + Aq_1q_0 + AB'q_1 \quad (3)$$

$$D = A'B'q_1 + A'q_1q_0' + Aq_1'q_0' + ABq_1' \quad (4)$$

Figure 8 shows the circuit, which can be implemented as schematics, or in Verilog as in Listing 2. Figure 9 shows the simulation on ModelSim.

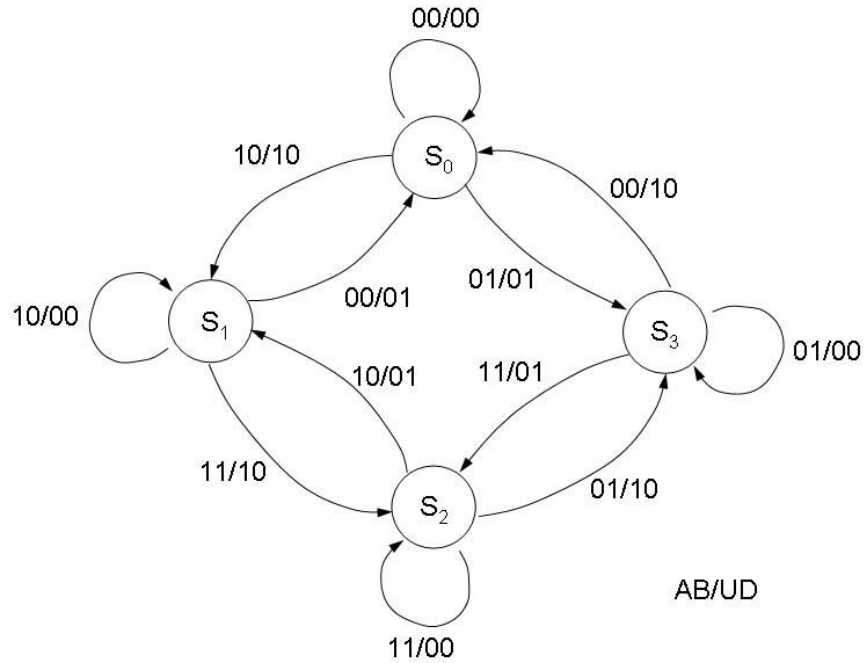


Figure 6: STG for the decoder unit

State	Next State				Output UD				
	q_1q_0	AB=00	AB=01	AB=10	AB=11	AB=00	AB=01	AB=10	AB=11
S0	00	00 (S0)	11 (S3)	01 (S1)	XX	00	01	10	11
S1	01	00 (S0)	XX	01 (S1)	10 (S2)	01	XX	00	10
S2	10	XX	11 (S3)	01 (S1)	10 (S2)	XX	10	01	00
S3	11	00 (S0)	11 (S3)	XX	10 (S2)	10	00	XX	01

Table 1: State Table

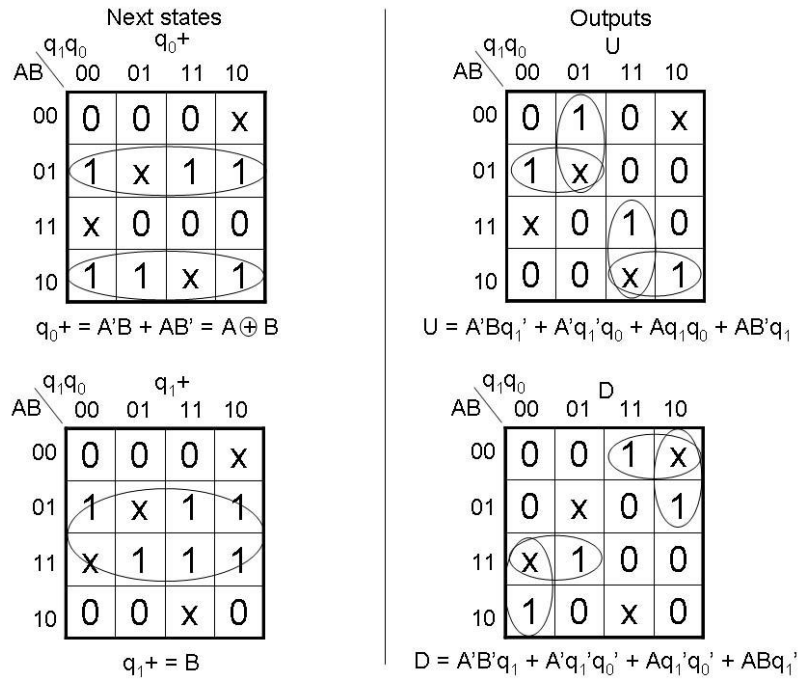


Figure 7: Reduction by Karnough Map

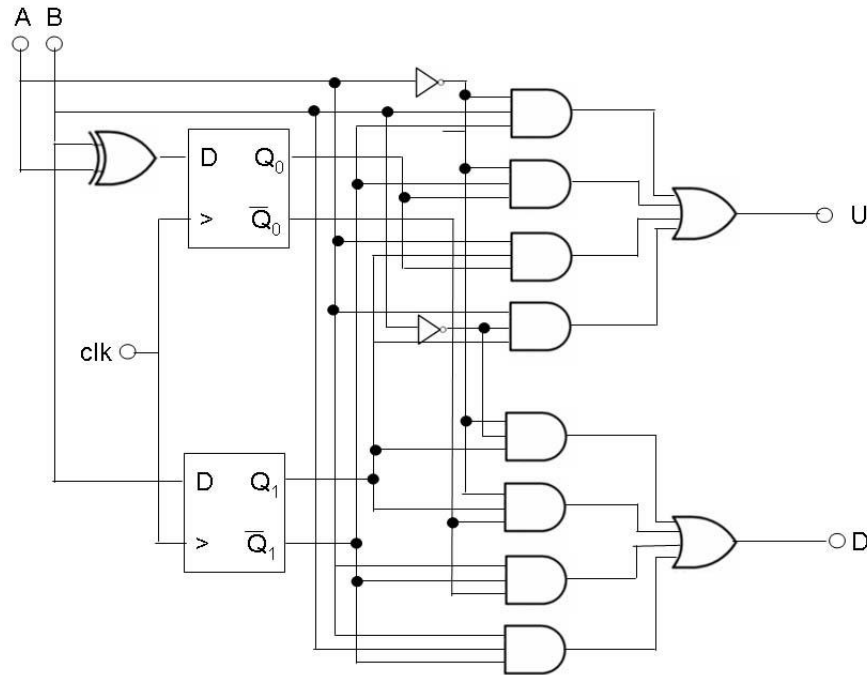


Figure 8: The decoder schematics

```

module x4enc(
    input A,
    input B,
    input Clk,
    output U,
    output D
);
    reg q0, q1, Q0, Q1; // next states are capitalized
    // Output equations
    assign U = (~A&B&~q1) | (~A&~q1&q0)|(A&q1&q0)|(A&~B&q1);
    assign D = (~A&~B&q1)|(~A&q1&~q0)|(A&~q1&~q0)|(A&B&~q1);
    always @ (A, B, q0, q1)
        begin
            // next states
            Q0 = A^B;
            Q1 = B;
        end
    always @ (posedge Clk)
        begin
            q0 <= Q0;
            q1 <= Q1;
        end
endmodule

```

Listing 2: Verilog code for the x4 decoder (design A)

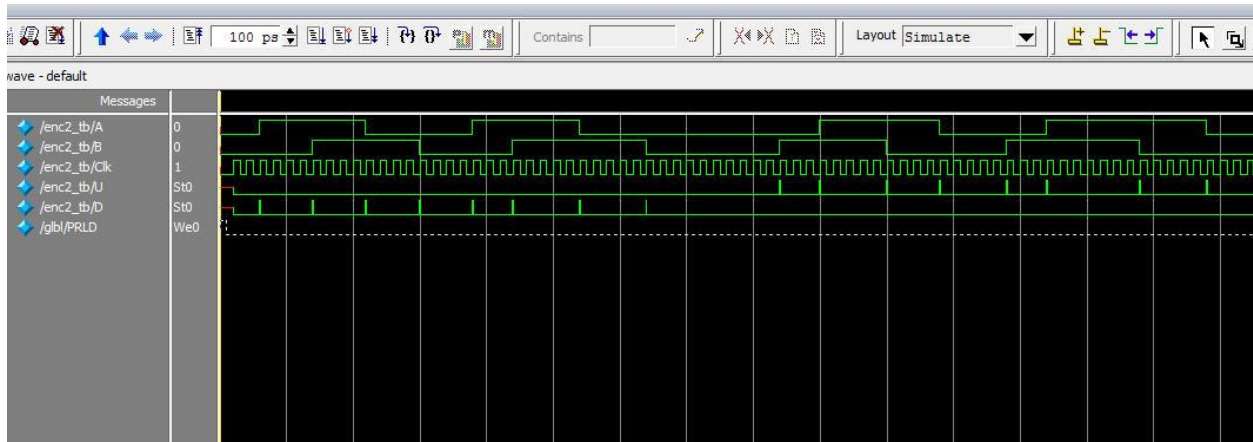


Figure 9: Simulation result from ModelSim

X4 Decoder (Design B)

One drawback of the X4 decoder above is the designer has to do the Karnaugh map manually. This approach is susceptible to human errors. We would rather want to let the synthesis software do the hard job for us. Another issue is this decoder has to be used with a counter that has separate pins for up/down pulses. In Xilinx ISE Verilog coding example, they provide up/down counters with count and direction input pins. So in this section we will consider a behavioral approach to implement a decoder that gives pulse/direction outputs.

One could use a systematic tool to tackle this design, such as an Algorithmic State Machine (ASM) chart. This problem is not too complicated to start coding right away, if you choose to do so. The idea is, a count output pulse is generated on each rising and falling edges of A and B. This gives us 4 times resolution as before. Now, for the direction output, during the change of one signal, we simply check the logic of other signal to determine which signal is leading. For example, suppose at the instant A changes from 0 to 1, we check B and find it has logic 1. This means B is leading A and hence the direction must be up.

So the trick of this approach is, how could we detect the rising (positive) and falling (negative) edges of A and B? The Verilog module `x4enc2` in Listing 3 illustrates a simple method. Just create a 2-bit registers `Ar` and `Br` to keep the logic of A and B, respectively, at the previous clock state. Then, rising edge of A implies that the value in `Ar` must equal "01," while falling edge gives a value "10." This way we could detect all the positive and negative edges of A and B. The code to determine the logic of other signal at edges of one is straightforward.

Figure 10 shows a simulation result of `x4enc2` from ModelSim. The module gives count and direction outputs as desired.

```

module x4enc2(
    input A,
    input B,
    input clk,
    output reg p,
    output reg dir
);
reg [1:0] Ar, Br;
wire A_posedge = (Ar==2'b01); // now we can detect A rising edges
wire A_negedge = (Ar==2'b10); // and falling edges
wire B_posedge = (Br==2'b01); // now we can detect B rising edges
wire B_negedge = (Br==2'b10); // and falling edges

always @ (posedge clk)
begin
    Ar <= {Ar[0], A};
    Br <= {Br[0], B};
    if (A_posedge)
    begin
        p <= 1;
        if (~Br[1]) dir<=1;    // A leads B
        else dir<=0;
    end
    else if (A_negedge)

```

```
begin
    p <= 1;
    if (Br[1]) dir<=1;
    else dir<=0;
end
else if (B_posedge)
begin
    p <= 1;
    if (Ar[1]) dir<=1;
    else dir<=0;
end
else if (B_negedge)
begin
    p <= 1;
    if (~Ar[1]) dir<=1;
    else dir<=0;
end
else p<=0;
end
endmodule
```

Listing 3: Verilog code for the x4 decoder (Design B)

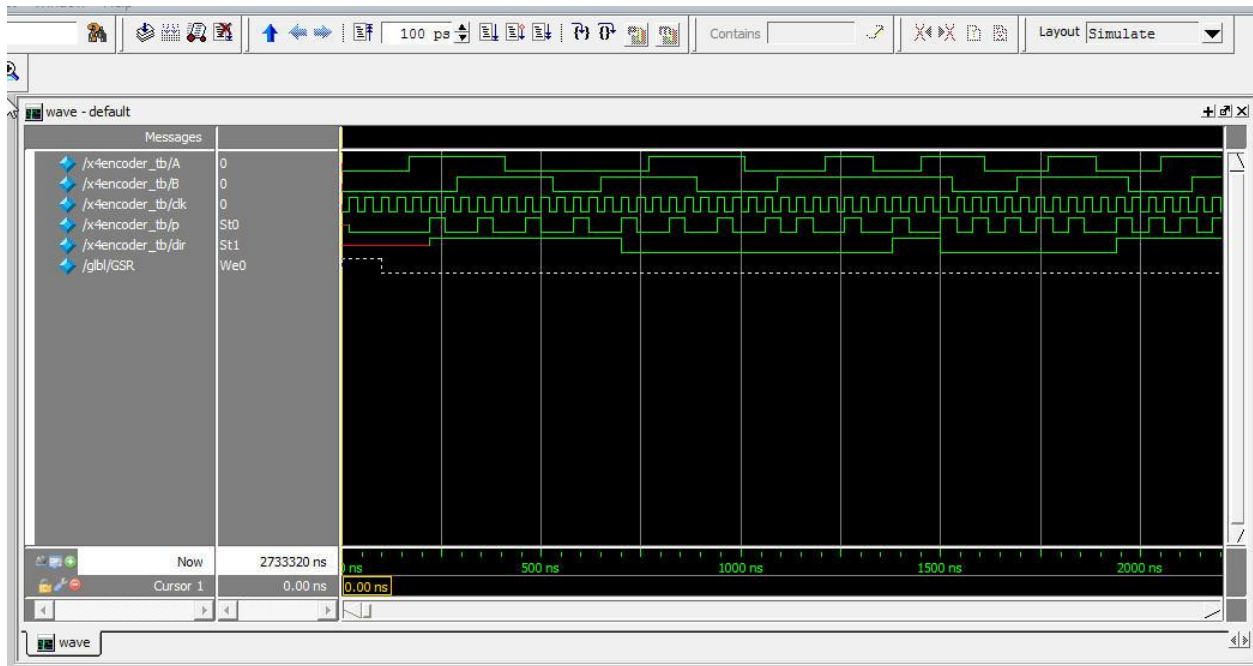


Figure 10: Simulation result of x4enc2 module

Summary

This document provides FPGA solutions to motor encoder interface. A modern microcontroller might have some special pins for encoder signals, it usually supports only 1 channel. Using interrupt pins to tackle the edges of A and B simultaneously for more than one channel is problematic, especially for motors running at high speeds. So It is best to implement a hardware circuitry for this task.

In addition to a digital filter scheme, we give two designs for the decoder module. Both have their pro and con. A counter with count and direction inputs is more common, so design B might be preferred by many developers. We have to be careful on one issue though. In design B, the direction output changes state at the same time with the positive edge of a count pulse. So metastability problem could occur if the counter changes at the positive edge of count. A simple way to circumvent this is to increment or decrement the count at negative edge of each count pulse. If the decoder and counter are synchronized by a common clock, the direction signal will have one clock cycle to settle. For a very high system clock rate, this might not suffice. In such system design A might be a better choice.