Project Report

32-Bit Priority Encoder

Group Members

1. Aman Imran (cs211249)
2. Muhammad Usman (cs211140)

Literature Review

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines. The output lines, as an aggregate, generate the binary code corresponding to the input value. The encoder can be implemented with OR gates whose inputs are determined directly from the truth table. The encoder has the limitation that only one input can be active at any given time. If two inputs are active simultaneously, the output produces an undefined combination. To resolve this ambiguity, encoder circuits must establish an input priority to ensure that only one input is encoded. If we establish a higher priority for inputs with higher subscript numbers will have high priority for generating output. Another ambiguity in the encoder is that an output with all 0's is generated when all the inputs are 0; but this output is the same as when input with subscript 0 is equal to 1. The discrepancy can be resolved by providing one more output to indicate whether at least one input is equal to 1 which is also known as high impedance.

A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. An enable bit is used. This is high priority encoder has active low inputs i.e. 0 is high and 1 is low. Logic circuit of 74x148 IC is implemented which is 8-input priority encoder. Cascading of 4 8-input priority encoders with active low input/outputs are implemented and the final output will be with normal binary output i.e. of I31 will be high then output will be 11111. Eo is enable output which is used in cascading. Its low will show to the next cascaded block that the respective high input is to be found and its high will show that the input is finding so next blocks will not be checked. GS is got something, its 0 will show that the input for encoder in find.

Truth table

Encoder module for 8bits

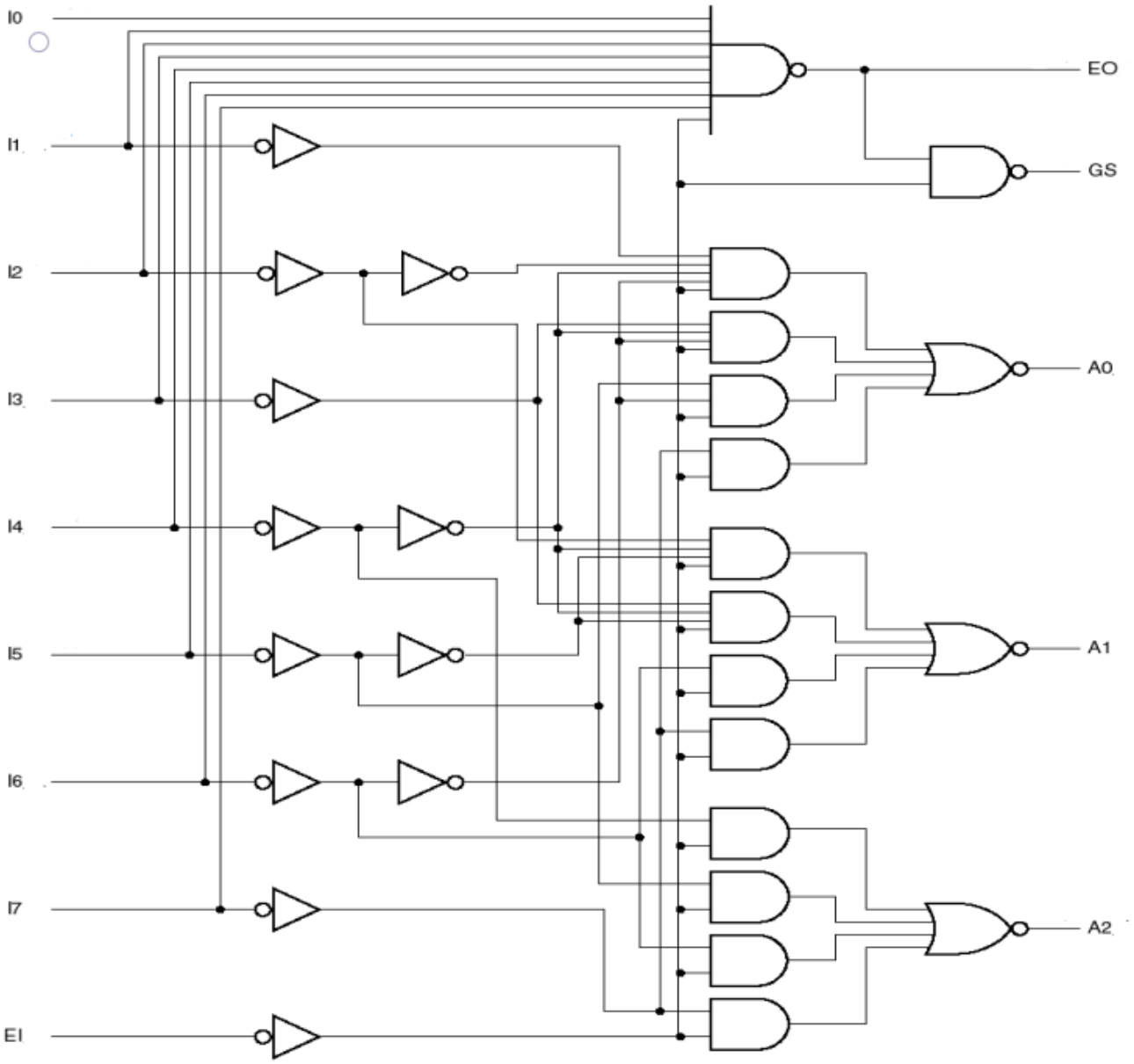
Inputs										Outputs				
EN	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**		**O2**	**O1**	**O0**	**GS**	**EO**
1	x	x	x	x	x	x	x	x		1	1	1	1	1
0	x	x	x	x	x	x	x	0		0	0	0	0	1
0	x	x	x	x	x	x	0	1		0	0	1	0	1

0	x	x	x	x	x	0	1	1		0	1	0	0	1
0	x	x	x	x	0	1	1	1		0	1	1	0	1
0	x	x	x	0	1	1	1	1		1	0	0	0	1
0	x	x	0	1	1	1	1	1		1	0	1	0	1
0	x	0	1	1	1	1	1	1		1	1	0	0	1
0	0	1	1	1	1	1	1	1		1	1	1	0	1
0	0	1	1	1	1	1	1	1		1	1	1	1	0

32bit Priority Encoder

Inputs																																		Outputs				
EN	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**	**I11**	**I12**	**I13**	**I14**	**I15**	**I16**	**I17**	**I18**	**I19**	**I20**	**I21**	**I22**	**I23**	**I24**	**I25**	**I26**	**I27**	**I28**	**I29**	**I30**	**I31**		**O4**	**O3**	**O2**	**O1**	**O0**
0	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	0	0	0
0	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	0	0	1
0	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	0	1	0
0	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	0	1	1
0	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	1	0	0
0	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	1	0	1
0	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	1	1	0
0	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	0	1	1	1
0	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	0	0	0
0	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	0	0	1
0	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	0	1	0
0	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	0	1	1
0	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	1	0	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	1	0	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		0	1	1	1	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	0	1	1	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	0	0	0	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	0	0	0	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1		1	0	0	1	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1	1		1	0	0	1	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1	1		1	0	1	0	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1	1		1	0	1	0	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1	1		1	0	1	1	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1	1		1	0	1	1	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1	1		1	1	0	0	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1	1		1	1	0	0	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1	1		1	1	0	1	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1	1		1	1	0	1	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1	1		1	1	1	0	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1	1		1	1	1	0	1
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**	1		1	1	1	1	0
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0**		1	1	1	1	1

Logic circuit



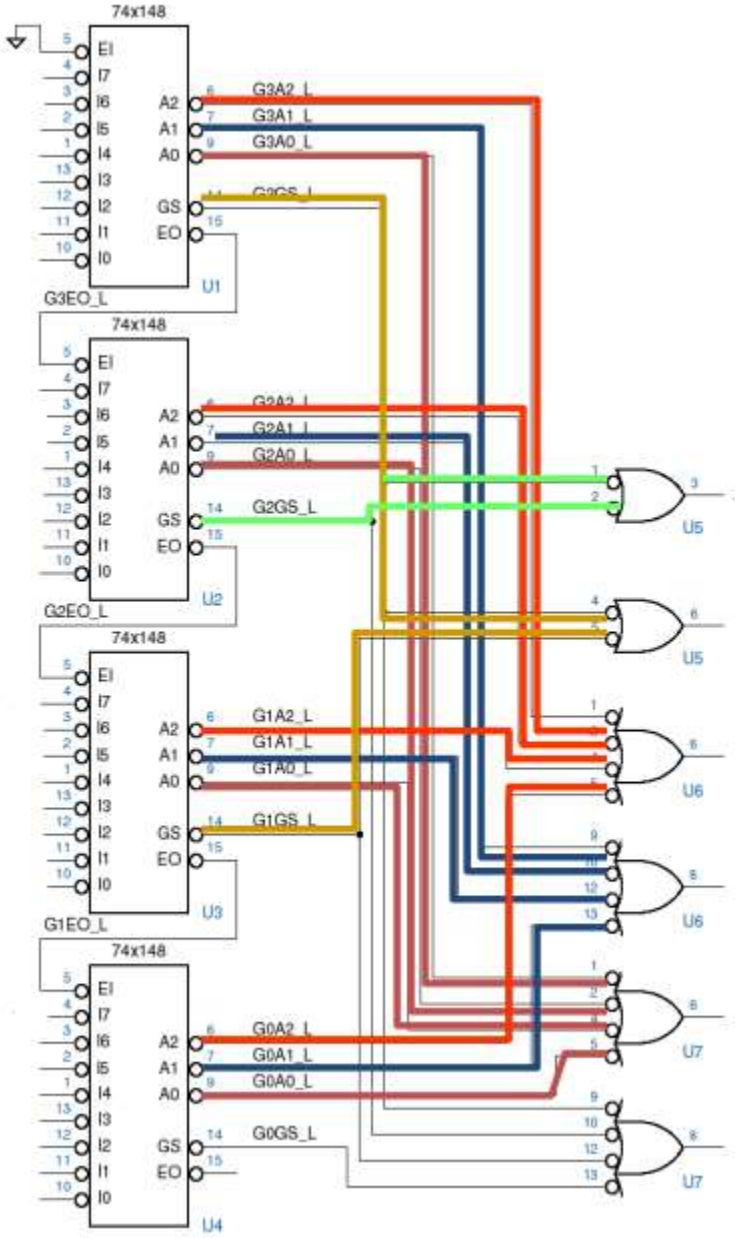
Active-low I/O

EN: Enable Input

GS: Got Something

EO: Enable Output

Block Diagram



Boolean Equations

For Encoder Module

$$\text{Out}[2] = \overline{((\overline{EN} \,.\, \overline{I[7]}) + (\overline{EN} \,.\, \overline{I[6]}) + (\overline{EN} \,.\, \overline{I[5]}) + (\overline{EN} \,.\, \overline{I[4]}))}$$

$$\text{Out}[1] = \overline{((\overline{EN} \,.\, \overline{I[7]}) + (\overline{EN} \,.\, \overline{I[6]}) + (\overline{EN} \,.\, I[5] \,.\, I[4] \,.\, \overline{I[3]}) + (\overline{EN} \,.\, I[5] \,.\, I[4] \,.\, \overline{I[2]}))}$$

$$\text{Out}[0] = \overline{((\overline{EN} \,.\, \overline{I[7]}) + (\overline{EN} \,.\, I[6] \,.\, \overline{I[5]}) + (\overline{EN} \,.\, I[6] \,.\, I[4] \,.\, \overline{I[3]}) + (\overline{EN} \,.\, I[6] \,.\, I[4] \,.\, I[2] \,.\, \overline{I[1]}))}$$

$$EO = \overline{(I[0] \,.\, I[1] \,.\, I[2] \,.\, I[3] \,.\, I[4] \,.\, I[5] \,.\, I[6] \,.\, I[7] \,.\, \overline{EN})}$$

$$GS = \overline{(EO \,.\, \overline{EN})}$$

For Main Module

$$\text{Out}[4] = \overline{GS2} + \overline{GS1}$$

$$\text{Out}[3] = \overline{GS1} + \overline{GS3}$$

$$\text{Out}[2] = \overline{Out1[2]} + \overline{Out2[2]} + \overline{Out3[2]} + \overline{Out4[2]}$$

$$\text{Out}[1] = \overline{Out1[1]} + \overline{Out2[1]} + \overline{Out3[1]} + \overline{Out4[1]}$$

$$\text{Out}[0] = \overline{Out1[0]} + \overline{Out2[0]} + \overline{Out3[0]} + \overline{Out4[0]}$$

Verilog Code:

design.v

```
`include "encoder_8bit.v"

module encoder(i,out,gs);

    input [31:0] i;
    output [4:0] out;
    output gs;

    wire eo1, eo2, eo3, gs1, gs2, gs3, gs4;
    wire [2:0] out1, out2, out3, out4;

    encoder_8bit encoder1(.i(i[31:24]),.out(out1),.en(1'b0),.gs(gs1),.eo(eo1));

    encoder_8bit encoder2(.i(i[23:16]),.out(out2),.en(eo1),.gs(gs2),.eo(eo2));

    encoder_8bit encoder3(.i(i[15:8]),.out(out3),.en(eo2),.gs(gs3),.eo(eo3));

    encoder_8bit encoder4(.i(i[7:0]),.out(out4),.en(eo3),.gs(gs4),.eo());
```

```
    assign out[4] = (~gs2|~gs1);
    assign out[3] = (~gs1|~gs3);
    assign out[2] = (~out1[2]|~out2[2]|~out3[2]|~out4[2]);
    assign out[1] = (~out1[1]|~out2[1]|~out3[1]|~out4[1]);
    assign out[0] = (~out1[0]|~out2[0]|~out3[0]|~out4[0]);
    assign gs = ~gs1|~gs2|~gs3|~gs4;


endmodule
```

encoder.v

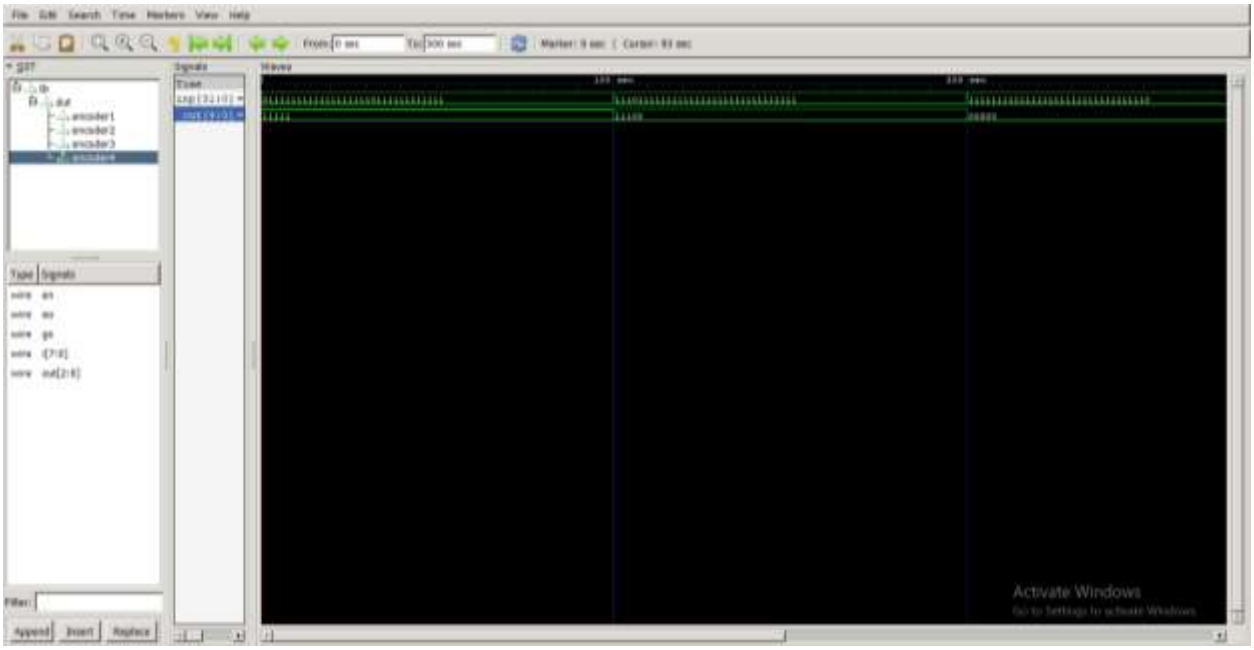
```
module encoder_8bit(i,out,en,gs,eo);

    input [7:0]i;
    input en;
    output gs,eo;
    output [2:0]out;

    assign out[2] = ~((~en & ~i[7]) | (~en & ~i[6]) | (~en & ~i[5]) | (~en & ~i[4]));
    assign out[1] = ~((~en & ~i[7]) | (~en & ~i[6]) | (~en & i[5] & i[4] & ~i[3]) | (~en &
~i[2] & i[5] & i[4]));
    assign out[0] = ~((~en & ~i[7]) | (~en & ~i[5] & i[6]) | (~en & i[6] & i[4] & ~i[3]) |
(~en & i[4] & i[6] & i[2] & ~i[1]));
    assign eo = ~(i[0] & i[1] & i[2] & i[3] & i[4] & i[5] & i[6] & i[7] & ~en);
    assign gs = ~(eo & ~en);


endmodule
```

testbench.v

```
module tb();

    reg [31:0]inp;
    wire [4:0]out;

    encoder dut(.i(inp),.out(out));

    initial begin
      $dumpfile("dump.vcd");
      $dumpvars(0);
    end

    initial begin
        inp <= 32'b01111111111111111110111111111111;

        #100;
```

```
        inp <= 32'b11101111111111111111111111111111;

        #100;
        inp <= 32'b11111111111111111111111111111110;

        #100;
        inp <= 32'b11111111111111111111111111111101;

        #100;
        inp <= 32'b11111111111111111111111111111111;

        #100;

        inp <= 32'b11111111111111111111111101111111;

        #100;
        $finish;
    end

endmodule
```

Waveforms



Applications of Priority Encoder

Keyboard Encoder

Priority encoders can be used to reduce the number of wires needed in a particular circuits or application that has multiple inputs. For example, assume that a microcomputer needs to read the 104 keys of a standard QWERTY keyboard where only one key would be pressed either “HIGH” or “LOW” at any one time.

One way would be to connect all 104 wires from the individual keys on the keyboard directly to the computers input but this would be impractical for a small home PC. Another alternative and better way would be to interface the keyboard to the PC using a priority encoder.

The 104 individual buttons or keys could be encoded into a standard ASCII code of only 7-bits (0 to 127 decimal) to represent each key or character of the keyboard and then input as a much smaller 7-bit B.C.D code directly to the computer. Keypad encoders such as the 74C923 20-key encoder are available to do just that.

Interrupt Requests

Other applications especially for **Priority Encoders** may include detecting interrupts in microprocessor applications. Here the microprocessor uses interrupts to allow peripheral devices such as the disk drive, scanner, mouse, or printer etc, to communicate with it, but the microprocessor can only “talk” to one peripheral device at a time so needs some way of knowing when a particular peripheral device wants to communicate with it.

The processor does this by using “Interrupt Requests” or “IRQ” signals to assign priority to all the peripheral devices to ensure that the most important peripheral device is serviced first. The order of importance of the devices will depend upon their connection to the priority encoder.

IRQ Number	Typical Use	Description
IRQ 0	System timer	Internal System Timer.
IRQ 1	Keyboard	Keyboard Controller.
IRQ 3	COM2 & COM4	Second and Fourth Serial Port.

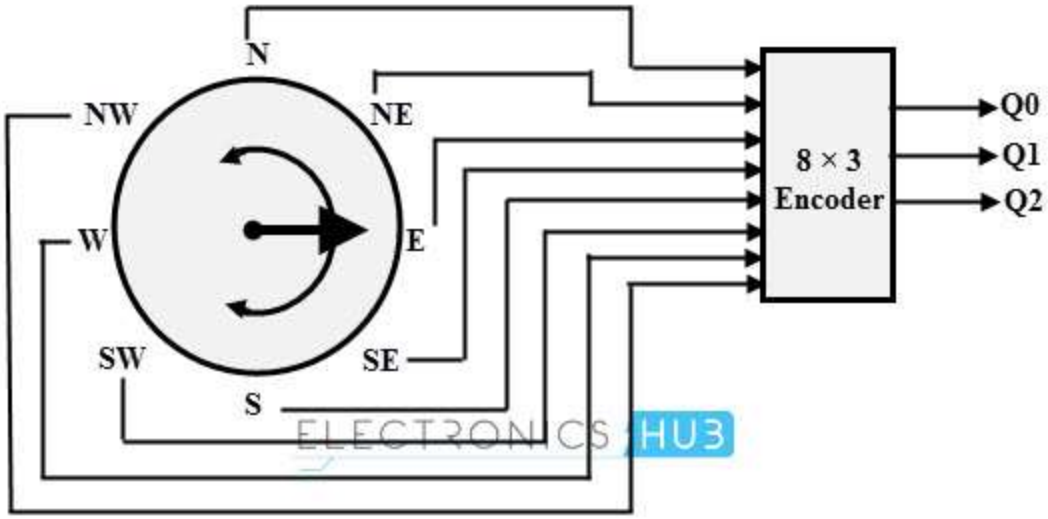
IRQ 4	COM1 & COM3	First and Third Serial Port.
IRQ 5	Sound	Sound Card.
IRQ 6	Floppy disk	Floppy Disk Controller.
IRQ 7	Parallel port	Parallel Printer.
IRQ 12	Mouse	PS/2 Mouse.
IRQ 14	Primary IDE	Primary Hard Disk Controller.
IRQ 15	Secondary IDE	Secondary Hard Disk Controller.

Because implementing such a system using priority encoders such as the standard 74LS148 priority encoder IC involves additional logic circuits, purpose built integrated circuits such as the 8259 Programmable Priority Interrupt Controller is available.

Positional Encoders

A magnetic positional control is another common application of priority encoders. Such control is used in robotic arm positioning and ship navigations. In such cases, encoder converts the rotary or angular position of a compass to a digital code. Then this code is input to the computer so that the navigational data is provided.

Below figure shows the simple compass encoder that converts the 8 positions to 3 bit output. For this type of input –output configuration, a 74LS148 IC is used which is an 8-to-3 line priority encoder. For indicating the compasses angular position, generally reed switches and magnets are used.



Compass Direction	Binary Output		
	Q_0	Q_1	Q_2
North	0	0	0
North - East	0	0	1
East	0	1	0
South - East	0	1	1
South	1	0	0
South - West	1	0	1
West	1	1	0
North - West	1	1	1