Stanford University
Department of Electrical Engineering

EE 121
Winter 2001

Lab 3
Programmable Priority Encoder

Prelab Due Date: Tuesday, January 29, 2001 at 9:30am

This is a challenging Lab--Start Early!!

1 Overview

This week's laboratory assignment is to design and build a combinational circuit called a *programmable priority encoder* (PPE). The PPE performs priority encoding of eight input signals, where the input priorities are specified by control inputs and therefore can change. You will use the Xilinx Foundation software to implement your design and demonstrate it using the XC2S100 FPGA in the lab. The block diagram of the system that you will build and test is shown in Figure 2. The logic symbol for the PPE is shown in the following figure.

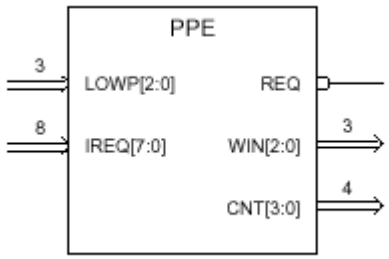

Figure 1: Prgrammable priority encoder logic symbol

The inputs to the PPE are eight request signals **IREQ[7:0]** and three control bits **LOWP[2:0]**. The control bits, **LOWP[2:0]**, encode in binary the value in the range {0,…,7} that corresponds to the input that currently has *lowest* priority. Input $i + 1$ has second lowest priority, while input $i + 7$ has highest priority. The expressions $i + j$ are calculated modulo 8; that is, only the low order three bits of the sum are used. For example, when **LOW[2:0]** is 011 = 3, then the input with highest priority is (3 + 7) mod 8 = 10 mod 8 = 2. Input priorities as a function of the inputs LOWP[2:0] are shown in the following table.

LOWP	Input priorities lowest							highest
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

The output **WIN[2:0]** encodes the active input of highest priority, according to the current setting of **LOWP[2:0]**. The **WIN[2:0]** value is valid only when there is at least one active request, and the active low output **/REQ** indicates whether any request input is active. The 4-bit **CNT[3:0]** output provides the number of active request inputs, in the range {0,…,8}.

The combinational logic that you will design for the PPE will be incorporated into the system shown in Figure 2. (This is a simple sequential circuit.)

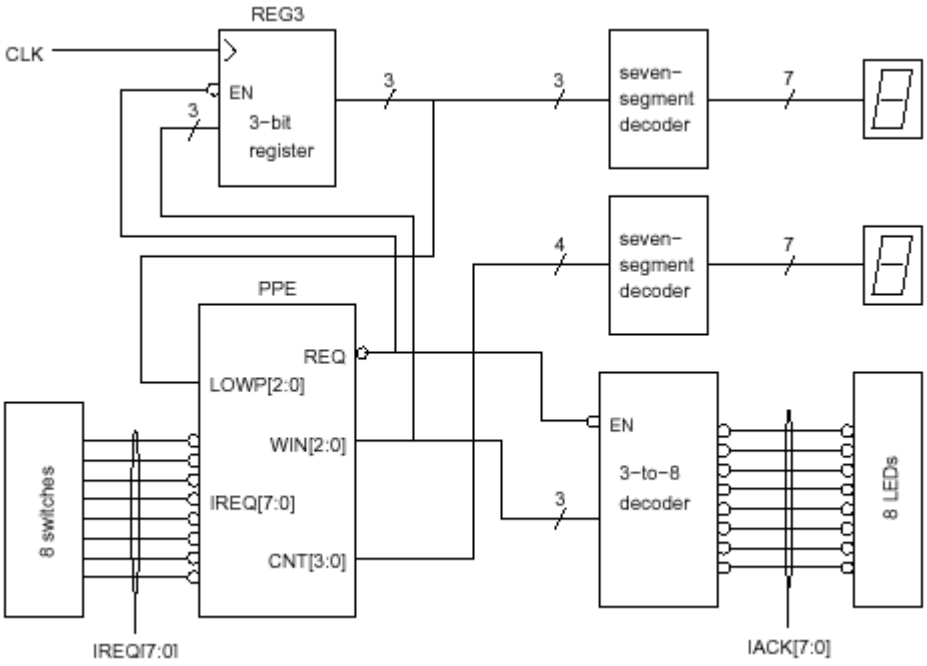

Figure 2: Programmable Priority Encoder demonstration circuit

The circuit shown in Figure 2 operates as follows. The 8-position DIP switch on the XStend prototyping extender board produce the eight interrupt request signals **IREQ[7:0]**. The number of active request signals is the 4-bit output **CNT[3:0]**, which is supplied to a seven-segment decoder that drives a seven-segment display on the XStend board.

If any request input is active, the input with the highest priority, as determined by the current value of **LOWP[2:0]**, will be "acknowledged". The binary encoding of that input will be output on the signals **WIN[2:0]**, and this encoding will be decoded by 3-to-8 decoder, enabled by PPE output **/REQ**, to activate exactly one of the acknowledge signals **IACK[7:0]**. The active **IACK*i*** turns on the corresponding LED of the LED bar graph.

At discrete time instants, specified by the rising edge of a clock signal, the encoded value of the winning interrupt request is stored in a 3-bit register, so that it becomes the lowest priority request for the next interrupt cycle. The clock signal will be produced either using a signal generator set to 1 Hz or slower or by a pushbutton on the XStend board. The PPE output **/REQ** enables storing new values in the 3-bit latch, so the output of the latch, which is supplied to the PPE inputs **LOWP[2:0]**, are unchanged until a request input is active. The register's current value is stored on the second seven-segment display.

Laboratory Requirements

For this laboratory assignment, partners should work together.

A top-level schematic will be provided on the course web page that includes signal names and pin assignments. Please make sure that all signals defined in this file are still in your design. (Hint: Make your project look like Figure 2).

NOTE: For all your logic blocks, you are only allowed to use macros you build yourself and the following basic gates:

```
AND2, AND3, AND4, NAND2, NAND3, NAND4, NOR2, NOR3, NOR4,
INV, OR2, OR3, OR4, XOR2, XNOR2, M2_1, GND, VCC
```

What to turn in for this week's Prelab:

1. Complete Foundation Project with schematics of all logic blocks in your design (submitted electronically)
2. Test script demonstrating that your design works (submitted electronically)
3. A report file that specifies resources used in the FPGA for the design—ie, the number of CLBs/IOBs/etc used. Look under the "reports" tab in Xilinx foundation. Summarize and discuss the results in your README in one paragraph (submitted electronically)
4. Extract timing information for the design as demonstrated in lecture and submit the file. The "Analyze against Auto Generated Design constraints…" button in the Xilinx Timing Analyzer tool will provide the values needed. Summarize and discuss the results in your README in one paragraph (submitted electronically)
5. Look at your design in the Xilinx FPGA Editor—be impressed it did all that wiring and not you! ☺ (show your TA in lab section)
6. Readme file (submitted electronically—see below)

Lab Exercise

During lab time, you will get a chance to compile your design, download it onto your Xilinx chip, and test it with real signals. There is no lab write-up for this lab. However, you do need to demo your circuit to your TA, using any test values your TA desires.

Submission

As penance for all the paper we wasted on Lab 1, we will use electronic submission for this and all future labs in this course. You do not need to print anything out. To submit your lab electronically:

(0) Write a README file that contains your name, your partner's name, the name of the project, and any information that the TAs should know when evaluating your design.

You should say whether or not you think your design completely meets the assignment's requirements. If it does, tell us which script file(s) you used to prove to yourself that your design worked.

If your design doesn't work, explain your approach and what problems you've encountered. List each major component, whether or not you think it works, and what script file(s) you used to verify them. Identify what component(s) or what stage of integration does not work.

Designs without any README file will lose points. But if your design passes both your tests and ours, then we will not expect any analysis of your design in the README file. A simple file with the basic information (names, project name, whether you think it works, and the name of a script file) is all that is necessary if your design is fully functional. If it is not, however, the README file is very important. Non-functional designs without README files will almost certainly receive very poor grades; non-functional designs that clearly identify what works and what does not may receive substantial partial credit. (depending, of course, on exactly how much is working)

(1) Choose Project->Clear Implementation Data from the Project Manager to remove all the implementation-related files that we don't need to see.
(2) Choose File->Archive Project. Set the Compression Factor to Max. Do not give your archive a password. Then click Next.
(3) Accept the default choice of archiving the entire project. Click Next.
(4) Add any extra files outside your project directory that you want to add to your archive. Usually, you won't need to add any. Then click Finish. A Zip file with your project's name will appear in the directory one level above your project directory. This is your archived project.
(5) Submit your archived project in one of the following ways:

(a) Drop it in the "submit" directory of your Z drive on the ee121-server. You will need to use one of the computers in Packard 128 to access your submit directory.

You will be able to write to the submit directory but not modify or delete any of the files in it. If you need to re-submit your project, you must copy a new archive into the submit directory.

(b) E-mail your archived project as an attachment to the TA who heads your lab section: Joel Coburn (jcoburn@stanford.edu) for Tuesday night, Frederic Sarrat (fsarrat@stanford.edu) for Wednesday afternoon, James Nielsen (jfn@stanford.edu) for Wednesday night, or Lee Kenyon (lakenyon@stanford.edu) for Thursday night.

Remember, you must submit your project electronically, whether or not it works correctly, before 9:30 AM on the due date or you will receive no credit.