Experiment 05

Objectives

(i) Design and simulate a 4 × 16 decoder using 2 × 4 decoders (with Enable)
(ii) Design, simulate, and implement a Priority Encoder
(iii) Design, simulate, and implement a Binary-to-Gray Code Converter

(i) Decoder 4 x 16

A 4 x 16 decoder can be made using five 2 x 4 decoders as shown in Fig. 1.

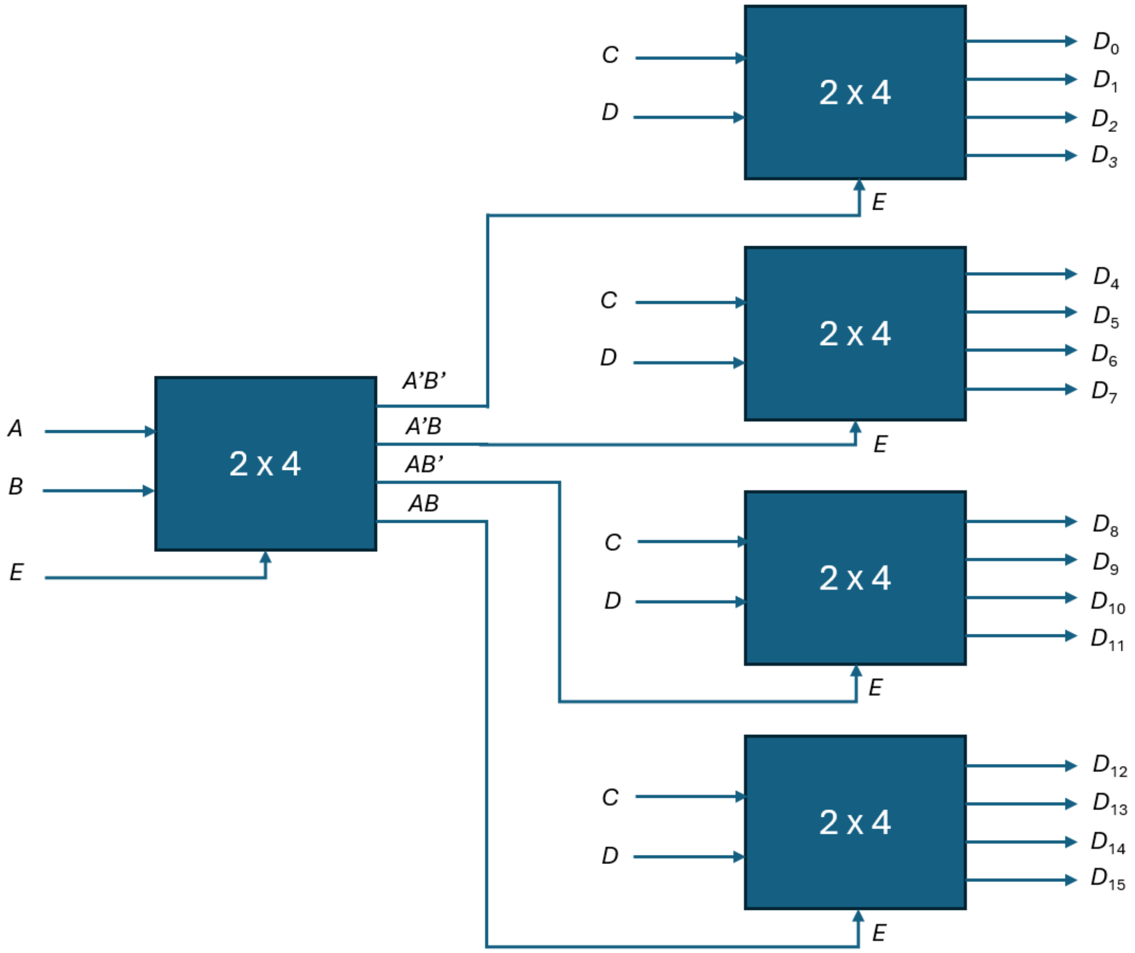


Figure 1 Creating a 4 x 16 decoder using 2 x 4 decoders with enable

Procedure

Step 1: Write a Verilog module for a 2 x 4 decoder, and use it to create a 4 x 16 decoder as shown in Fig. 1.

Step 2: Write a testbench to simulate the results for at least eight inputs combinations. Verify the timing diagram with the truth table, **and save the screenshot for the report**.

Step 3: Use the four slide switches on the RPI module as inputs (eg. Inputs (I_0 I_1 I_2 I_3) on Switches (*H G F E*). To display the sixteen distinct decoder outputs, use 06 LEDs on the RPI module (eg. $D_0 – D_5$), 04 on the PYNQ-Z2 board (eg. $D_6 – D_9$). The remaining six can be shown as R-G-B combinations on the two RGB LEDs or by connecting six external LEDs to the PMod. **Update the same on the RPI and PYNQ Z2 XDC files (and include the updates in your report).**

Step 2: Generate bitstream and program the device. Verify the working of all combinations of inputs.

Record a video showing all input/output combinations.

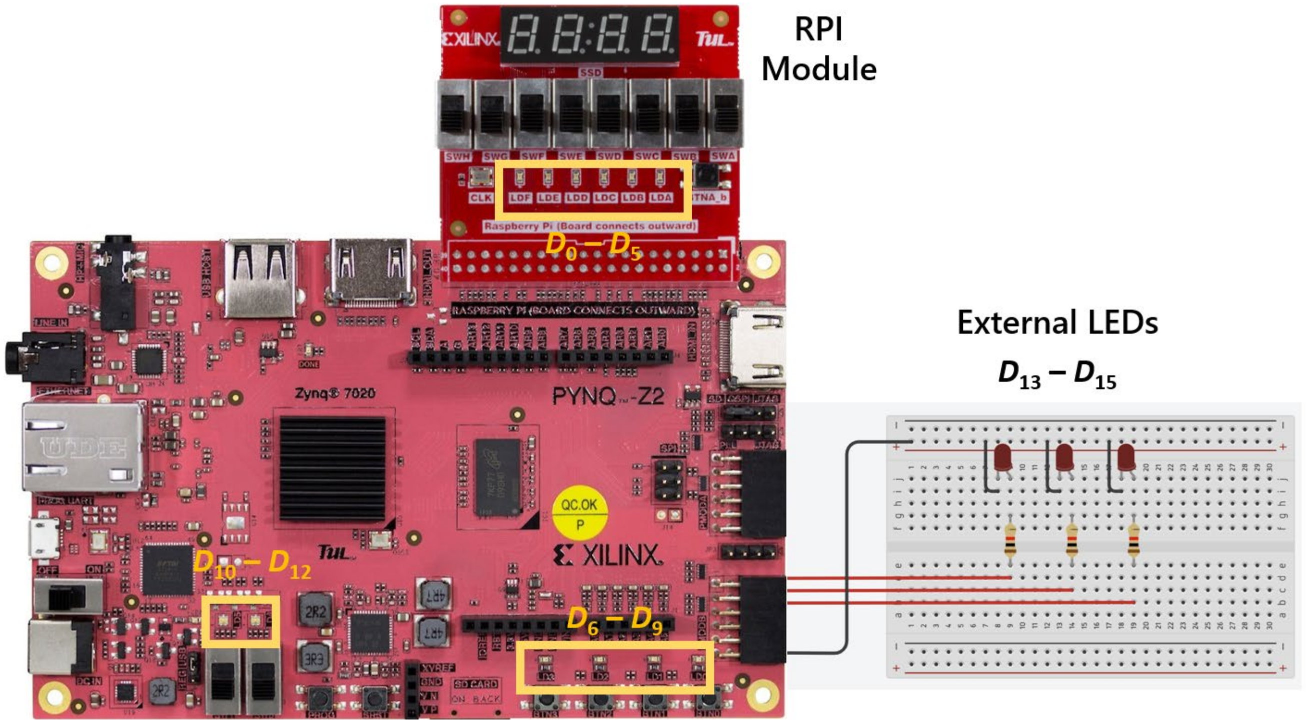


Figure 2 Circuit Schematic of the connections

(ii) 4-bit Priority Encoder

A Priority Encoder implements a priority function: If more than one inputs are 1, the input having highest priority takes precedence. Here, the input having higher index will take priority.
(*V* shows Valid priority; *V* = 1 only when at least one of the priority inputs are high). The Truth Table is shown in Table 1.

Procedure

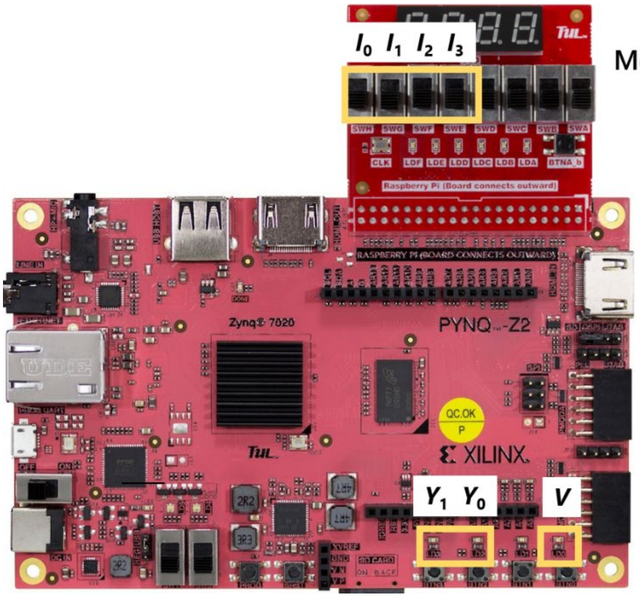
Step 1: Write the Verilog module for the priority encoder based on the Truth Table (Table 1). Simulate the circuit with a testbench showing at least 8 input combinations. (Insert the timing diagram in you report).

Step 2: Refer to Figure 3 for connections and input/output configurations. Update the XDC files and insert the updates in your report.

Step 3: Generate the bitstream and program the device. Verify all input combinations and record a video for your demonstration.

Table 1 Truth Table of a 4-bit Priority Encoder

Inputs				Outputs		
I_0	I_1	I_2	I_3	Y		V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1



Inputs (I_0 I_1 I_2 I_3) on
RPI Module Switches (H G F E)

Outputs
V : LD0
Y_1 : LD3
Y_0 : LD2

Figure 3 Connections and input/output pin configurations

(iii) 4-bit Binary-to-Gray Converter

A 4-bit Binary-to-Gray converter takes a 4-bit binary number and converts it to corresponding Gray code. Refer to the truthtable and the logic circuit shown in Table 2

Table 2 Truth Table for Binary-to-Gray Decoder

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

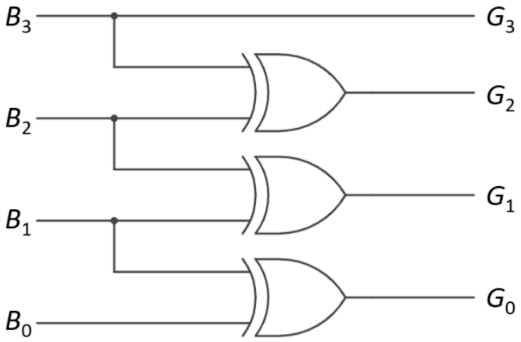


Figure 4 Logic circuit of Binary-to-Gray Converter

Procedure

Step 1: Write the Verilog module for the Binary-to-Gray Converter.

Step 2: Create a testbench and simulate at least eight combinations. Save the timing diagram for your report.

Step 3: Update the XDC files of RPI module and PYNQ Z2 boards as follows:

Input (B_3 B_2 B_1 B_0) – RPI Switches (*H, G, F, E*) respectively

Output (G_3 G_2 G_1 G_0) – PYNQ Board LEDs (LD3, LD2, LD1, LD0) respectively.

Step 4: Generate the bitstream, program the device. Verify all input-output combinations and record a video showing at least ten combinations.

Online Reference

R. Chouhan, *Experiment 05, EEL2020 Digital Design* (Lab Notes)