Digital Design and Verification

Lab Manual # 08 – Combinational Circuits using System Verilog

Release: 2.0

Date: 07-July-2024

NUST Chip Design Centre (NCDC), Islamabad, Pakistan

Revision History

Revision Number	Revision Date	Revision By	Nature of Revision	Approved By
1.0	22/01/2024	Hira Sohail, Muhammad Bilal	Complete manual	-
1.1	28/02/2024	Hira Sohail, M. Bilal	Revision in manual	-
2.0	07/07/2024	Saad Khan	Revision	-

File Submission Criteria.

All Current and future submissions should be done in this manner, anyone who fails to do so won't be able to get any marks.

General zip File hierarchy:

Fullname.zip/ fullnamefolder/ taskx/ taskx.sv, taskxtb.sv taskx.bit

Note:

1. "x": should be replaced by your current task no.
2. All small letters should be used while naming all the files and folders.
3. No spaces should be used while naming the files and folders.

Files Submission For lab-5:

Task	Source File Names	Testbench File Names	Constraint Files	bitstream File Names	Top module Prototypes	Testbench Prototypes
1	mux_gate.sv mux_behav.sv mux_case.sv	mux_gate_tb.sv mux_behav_tb.sv mux_case_tb.sv	mux_gate_const.xdc mux_gate_behav.xdc mux_gate_case.xdc	mux_gate_tb.bit mux_behav_tb.bit mux_case_tb.bit	mux_gate(a,b,c,d,s,o); mux_behav(a,b,c,d,s,o); mux_case(a,b,c,d,s,o);	mux_gate_tb(a,b,c,d,s,o); mux_behav_tb (a,b,c,d,s,o); mux_case_tb (a,b,c,d,s,o);
2	bcd_ssd.sv	bcd_ssd_tb.sv		bcd_ssd.bit	bcd_ssd([3:0]sw, [7:0] ssd, [7:0] an);	bcd_ssd_tb([3:0] sw, [7:0] ssd, [7:0] an);
3	dec_bcd.sv	dec_bcd_tb.sv	dec_bcd.xdc	dec_bcd.bit	dec_bcd (logic [9:0] decimal_in, logic [3:0] bcd_out);	dec_bcd_tb (logic [9:0] decimal_in, logic [3:0] bcd_out);
4	barrel_shifter.sv	barrel_shifter_tb.sv	barrel_shifter_const.xdc	barrel_shifter.bit	barrel_shifter(logic [3:0] in, logic [1:0] shift, logic dir, logic [3:0] out);	barrel_shifter_tb(logic [3:0] in, logic [1:0] shift, logic dir, logic [3:0] out);
5	cla_4bit.sv	cla_4bit_tb.sv	cla_4bit_const.xdc	cla_4bit.bit	cla_4bit (logic [3:0] A, logic [3:0] B, logic Cin, logic [3:0] Sum, logic cout);	cla_4bit_tb([3:0] A, logic [3:0] B, logic Cin, logic [3:0] Sum, logic Cout);

Contents

Objective

The objective of this lab is to:

- Understand and practice basic concepts of System Verilog for combinational circuits.
- Understand 4x1 Mux, BCD to 7-segment decoder, decimal to BCD priority encoder, barrel shifter and implement carry look ahead adder.
- Practice writing good RTL using SV with comprehensive testbench.

Tools

- Xilinx Vivado
- Cadence Xcelium

Introduction

4x1 MUX

Consider a circuit in which the output "m" has to be selected from four inputs "u," "v," "w," and "x." Figure 1 shows how we can build the required 4-to-1 multiplexer by using three 2-to-1 multiplexers. The circuit uses a 2-bit select input "s1" and "s0" and implements the truth table shown in Figure 2, as well as the circuit symbol.

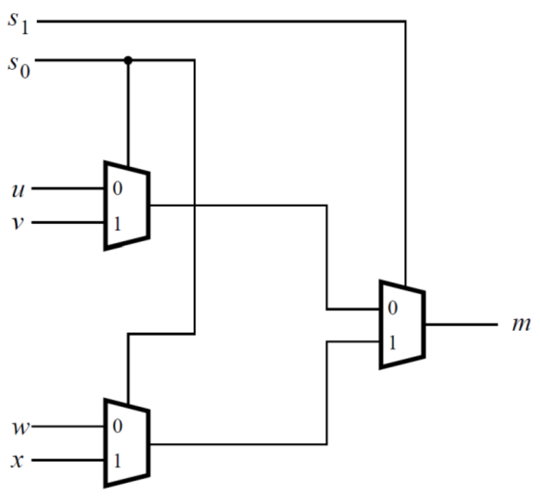

Figure 1. Circuit

s_1 s_0	m
0 0	u
0 1	v
1 0	w
1 1	x

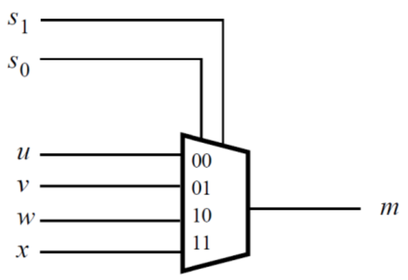

Figure 2. Truth Table and Symbol

LAB TASK 1.1
Create a System Verilog module to implement one-bit wide 4-to-1 multiplexer using Gate Level Programing.

LAB TASK 1.2
Create a System Verilog module to implement one -bit wide 4-to-1 multiplexer using Behavioral Modeling.

LAB TASK 1.3
Create a System Verilog module to implement one -bit wide 4-to-1 multiplexer using Case Statements.

Note: FPGA Prototyping of all 3 subtasks Required.

BCD to 7 Segment Decoder

In Binary Coded Decimal (BCD) encoding scheme each of the decimal number (0-9) is represented by its equivalent binary pattern (which is generally of 4-bits).

Whereas seven segment display is an electronic device which consists of seven Light Emitting Diodes (LEDs) arranged in some definite pattern (common cathode or common anode type), which is used to display Hexadecimal numerals (in this case decimal numbers, as input is BCD i.e., 0-9).

Two types of seven segment LED display:

1. **Common Cathode Type:** In this type of display all cathodes of the seven LEDs are connected together to the ground or -Vcc (hence, common cathode) and LED displays digits when some 'HIGH' signal is supplied to the individual anodes.
2. **Common Anode Type:** In this type of display all the anodes of the seven LEDs are connected to battery or +Vcc and LED displays digits when some 'LOW' signal is supplied to the individual cathodes.

But, seven segment display does not work by directly supplying voltage to different segments of LEDs. First, our decimal number is changed to its BCD equivalent signal then BCD to seven segment decoder converts that signal to the form which is fed to seven segment display.

This BCD to seven segment decoders has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g), this output is given to seven segment LED display which displays the decimal number depending upon inputs.

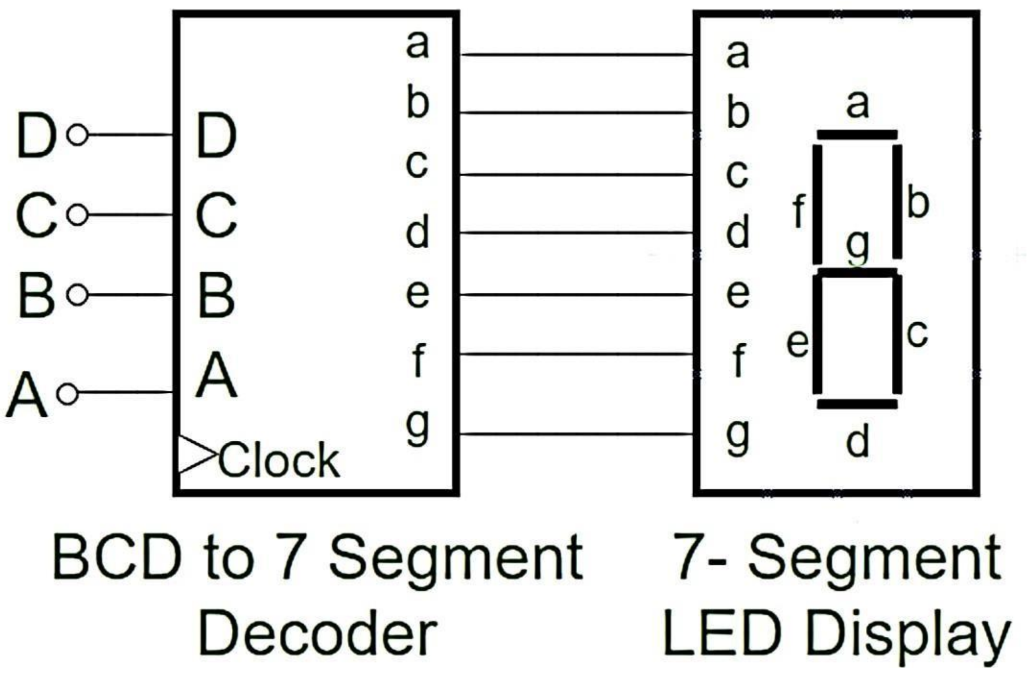

Figure 3. BCD to 7-segment decoder

A	B	C	D	a	b	c	d	e	f	g	H
0	0	0	0	1	1	1	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	1
0	0	1	1	1	1	1	1	0	0	1	1
0	1	0	0	0	1	1	0	0	1	1	1
0	1	0	1	1	0	1	1	0	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1

Table 1. Truth table – Common Cathode

Note

- For Common Anode type seven segment LED display, we only have to interchange all '0s' and '1s' in the output side i.e., (for a, b, c, d, e, f, and g replace all '1' by '0' and vice versa) and solve using K-map.
- Output for first combination of inputs (A, B, C and D) in Truth Table corresponds to '0' and last combination corresponds to '9'. Similarly rest corresponds from 2 to 8 from top to bottom.
- BCD numbers only range from 0 to 9, thus rest inputs from 10-F are invalid inputs.

LAB TASK 2

1. Create a System Verilog module for the 7-segment decoder.
2. Create a BCD decoder to handle values ranging from 0 to 9.
3. Use 4 switches to get a binary input from the FPGA board, decode it, and display it on the FPGA 7-Segment display.
4. Verify the results using a test bench.
5. Prototype it on FPGA.

The Decimal-to-BCD Priority Encoder

A Decimal to Binary-Coded Decimal (BCD) Priority Encoder is a digital circuit that takes a decimal input and produces a binary output in BCD form. BCD is a binary representation of decimal numbers where each decimal digit is represented by a 4-bit binary code.

The LS147 encodes nine data lines to four-line (8-4-2-1) BCD. The implied decimal zero condition does not require an input condition because zero is encoded when all nine data lines are at a high logic level. It has nine active-low inputs representing decimal numbers 1 through 9. The encoder produces the inverted BCD code corresponding to which of the nine inputs is activated.

Inputs									Outputs			
$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$\bar{A}_8$	$\bar{A}_9$	$\bar{O}_3$	$\bar{O}_2$	$\bar{O}_1$	$\bar{O}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

Table 2: Truth table for a Decimal-to-BCD Priority Encoder

Note the "don't-cares" (Xs) in the truth table. They imply that, if two inputs are activated simultaneously, only the highest data line is encoded. For example, if lines A1 and A5 are activated at the same time, A5 will be encoded producing the output 1010 (which is 0101 inverted, or BCD 5). That's why it's called a "priority" encoder. Moreover, the implied decimal zero condition requires no inputs since zero is encoded when all nine data lines are at HIGH.

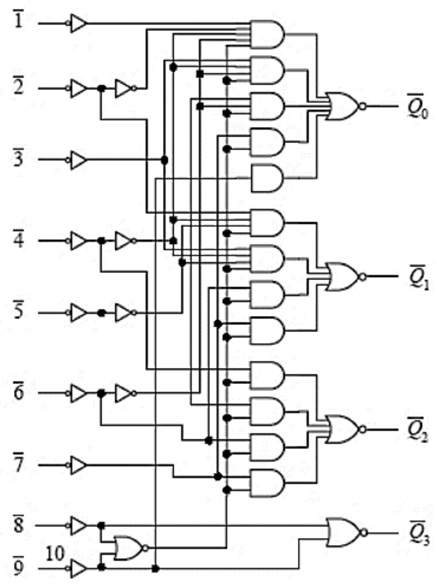

Fig 4: Circuit Diagram for a Decimal-to-BCD Priority Encoder

LAB TASK 3

Design a Decimal-to-BCD Priority Encoder using System Verilog. To test your RTL code, write a comprehensive testbench. Record your results and show the simulation in your report.

Barrel Shifter

A Barrel Shifter is a combinational block that allows for the shifting of binary data in a more flexible and efficient manner than traditional shifters. Unlike a conventional shifter that shifts data by a fixed amount, a barrel shifter can shift data by any specified number of positions. It gets its name from the visual representation of the circuit, where the shifters are arranged in a circular or barrel-like fashion.

Each shifter is responsible for shifting the data by a specific number of positions. The selection of the appropriate shifter is determined by the control inputs.

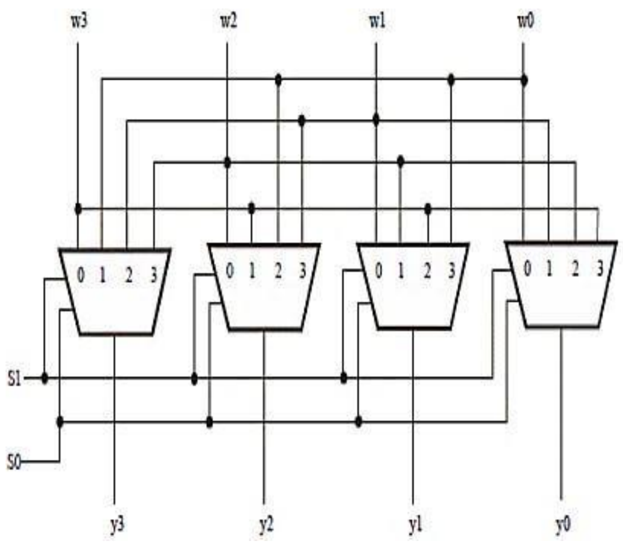

Fig 5: Design of 4-bit Barrel shifter using MUX

LAB TASK 4

Design a 4-bit Barrel Shifter using System Verilog. Write System Verilog code for the Barrel Shifter. Include modules for shifters, multiplexers, and the main Barrel Shifter. Simulate the Barrel Shifter using a testbench to verify its functionality. Create a report documenting the System Verilog code, simulation results, and explanations of the design. Create test cases to validate different shifting scenarios (e.g., left shift, right shift)

Carry Look Ahead Adder

The adder produces carry propagation delay while performing other arithmetic operations like multiplication and divisions as it uses several additions or subtraction steps. This is a major problem for the adder and hence improving the speed of addition will improve the speed of all other arithmetic operations. Hence reducing the carry propagation delay of adders is of great importance. There are different logic design approaches that have been employed to overcome the carry propagation problem. One widely used approach is to employ a carry look-ahead which solves this problem by calculating the carry signals in advance, based on the input signals. This type of adder circuit is called a carry look-ahead adder.

Carry Look Ahead Adder is digital circuit which performs n-bit addition operation. Here we only use generator and propagator to computes carry out. Boolean expression for carry out of a full adder is given by:

$$C_{out} = AB + (A \oplus B)C_{in}$$

A	B	Cin	AB	A⊕B	Cout
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	1

Table 3. Truth table of a full-adder

Truth table of a full-adder is given in above table. When A⊕B is 1, carry is propagated and when AB is 1, carry is generated. Let's replace A AND B with G (generator) and A XOR B with P (propagator). Then the carry out equation becomes.

$$C_{out} = G + PCin$$

We can also replace XOR operation with OR operation. It can also compute the carry out

$$C_{out} = AB + (A + B)C_{in}$$

Carry Out Equation

Now we can write the carry out equation in terms of generator and propagator

$$C1 = g0 + p0C_{in}$$

$$C2 = g1 + p1C1$$

$$C3 = g2 + p2C_2$$

$$C4 = g3 + p3C3$$

Let's expand these equations

$$\boxed{C1 = g0 + p0C_{in}}$$

$$C2 = g1 + p1C1$$

$$C2 = g1 + p1(g0 + p0C_{in})$$

$$\boxed{C2 = g1 + g0p1 + p1p0C_{in}}$$

$$C3 = g2 + p2C_2$$

$$C3 = g2 + p2(g1 + g0p1 + p1p0C_{in})$$

$$\boxed{C3 = g2 + g1p2 + g0p2p1 + p2p1p0C_{in}}$$

$$C4 = g3 + p3C3$$

$$C4 = g3 + p3(g2 + g1p2 + g0p2p1 + p2p1p0C_{in})$$

$$\boxed{C4 = g3 + g2p3 + g1p3p2 + g0p3p2p1 + p3p2p1p0C_{in}}$$

If we observe the carry out equation, there is no dependency on previous carry output for performing addition operation. This can perform fast and parallel addition operation.

Circuit Diagram

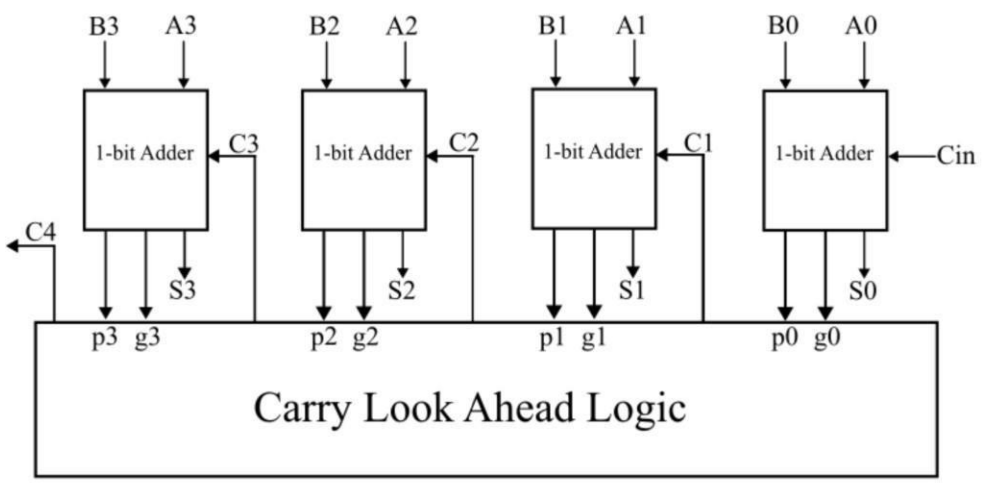

Figure 6. 4-bit carry look ahead adder

LAB TASK 5

Write a System Verilog code to implement 4-bit Carry Look Ahead Adder. To test your RTL code, write a comprehensive testbench.