<u>111-2 Digital System Design Homework 0</u>

0. After logging into the workstation, you should source two files to ensure that you can use nc-verilog and nWave successfully.

```
source /usr/cad/cadence/cshrc
source /usr/spring_soft/CIC/verdi.cshrc
```

1. 8-bit Carry Ripple Adder (40%)

In problem 1, you need to model an <u>8-bit carry ripple adder</u> (CRA). The input and output ports are illustrated in Figure 1. This is similar to the lab in *Switching Circuit and Logic Design* course.

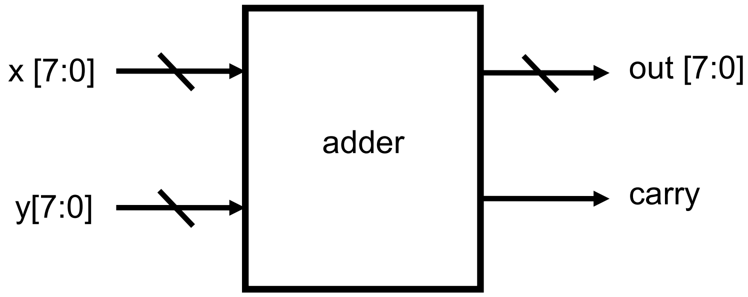

Figure 1

(1) (20%)

Modify "adder_gate.v" which contains the module name and input/output ports. Design at **gate level**. We suggest you design a *full adder* first and then instantiate eight full adders in the higher hierarchy. Use the given test bench, "adder_gate_test.v" to verify your design. To simulate, use the following command:

```
ncverilog adder_gate_test.v adder_gate.v +access+r
```

A waveform file "adder.fsdb" will be dumped. You can use waveform viewer to help your debugging:

```
nWave &
```

When the window appears, you need to open the dumped fsdb file and then add signals to the waveform viewer. Show your waveform result in the report.

(2) (10%)

Use continuous assignment (use "**assign**") to describe the transfer between input signals and output signals of an 8-bit carry ripple adder. You can use arithmetic operators. Start with the "adder.v" file, which contains the module name and input/output ports. To simulate, use the following command:

```
ncverilog adder_test.v adder.v +access+r
```

(3) (10%)

The **critical path** of a combinational circuit is defined as follows: on every path from input to output, the longest (largest sum of delay) one is the critical path. Specify 1ns delay for each Verilog primitive in (1) (*and*, *or*, ...). Illistrate the critical path of your design in the report and calculate the sum of the delay. Try to modify the 2nd line in "adder_gate_test.v". Use different cycle time and report the correctness.

```
`define CYCLE 20.0
```

Verify the timing of your design with waveform viewer.

```
ncverilog adder_gate_test.v adder_gate.v +access+r
```

2. 8-bit Barrel-shifter (40%)

Barrel-shifter has a simple and regular structure and is usually used in different microprocessors. Problem 2 asks you to design an <u>8-bit logical shift-left barrel-shifter</u> (shift to MSB with zero padding from LSB).

Figure 2 shows an operating example of barrel shifter. The control input (shl4, shl2, and shl1) determines the amount of shifting. The "shl4" performs a shift-left by 4 bits, the "shl2" performs a shift-left by 2 bits, and the "shl1" performs a shift-left by 1 bit. Thus, the 8-bit barrel shifter in Figure 2 shifts the input data left by 5 bits. The output data are left-shifted input data and 5 zeros.

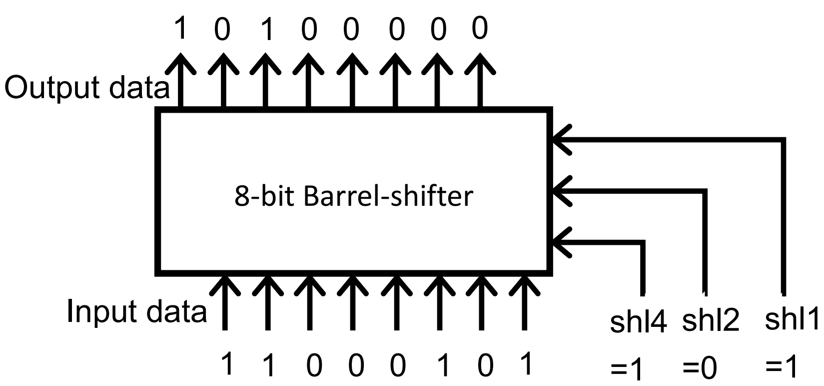

Figure 2

Figure 3 shows the regular structure of barrel shifter (shift from MSB to LSB). For an 8-bit barrel shifter, there are three levels of multiplexers. Each level contains eight 2-to-1 multiplexers. To efficiently describe the structure, we suggest you partition the design into three levels: *mux*, *level*, and *a barrel shifter*.

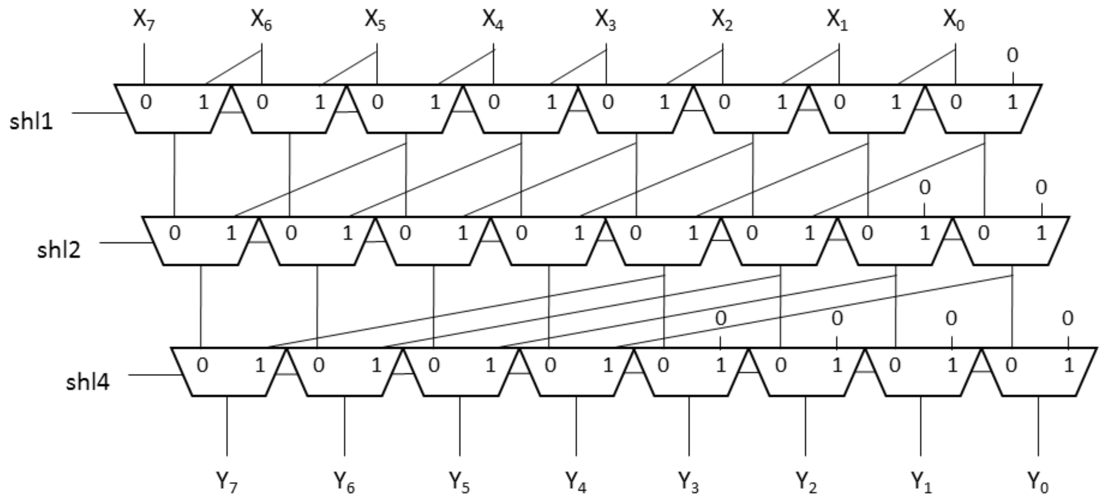

Figure 3

The input and output signals are illustrated in Figure 4.

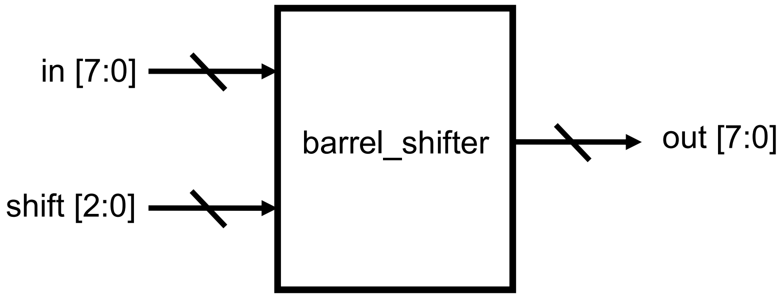

Figure 4

(1) (20%)

Modify the “barrel_shifter_gate.v” file. Use **gate level** description to design an 8-bit barrel shifter. Properly partition your design for clarity. Verify your design with “barrel_gate_test.v” by using the following command:

```
ncverilog barrel_gate_test.v barrel_shifter_gate.v +access+r
```

Show your waveform result in the report.

(2) (10%)

Design an 8-bit barrel-shifter with continuous assignment (use “**assign**”). Start with the “barrel_shifter.v” file, which contains the module name and input/output ports. Verify your design with the given test bench, “barrel_test.v”. Use the following command:

```
ncverilog barrel_test.v barrel_shifter.v +access+r
```

(3) (10%)

Illistrate the critical path of your design in the report and calculate the sum of delay. Specify 1ns delay for each Verilog primitive in (1). Try to modify the 2nd line in “barrel_gate_test.v”. Use different cycle time and report the correctness.

```
`define CYCLE 20.0
```

Verify the timing of your design with the following command:

```
ncverilog barrel_gate_test.v barrel_shifter_gate.v +access+r
```

3. Adder-Shifter Unit (20%)

Combine the previous two designs into an adder-shifter unit (Figure 5). The control signal “mode” is used to select the output from adder or barrel shifter. When mode = 1’b0, the out[7:0] is from barrel-shifter. When mode = 1’b1, out[7:0] is from adder. Note the “shift” input signal of the barrel-shifter is connected to the input Y[2:0]. At barrel-shifter mode, the output signal, “carry”, should keep 1’b0.

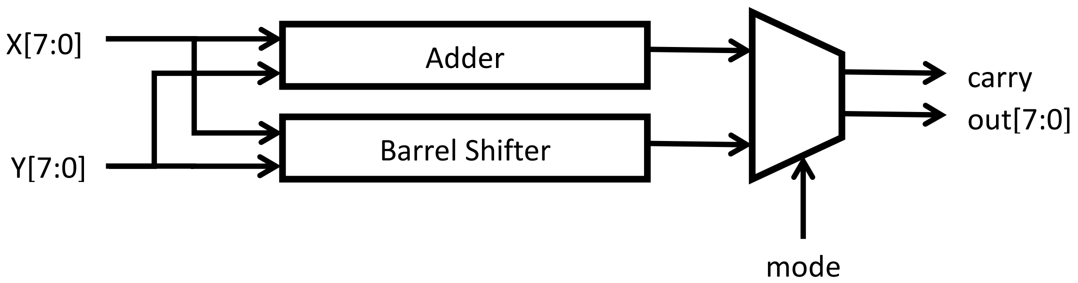

Figure 5

(1) (5%)

Instantiate the previously designed two transfer models (continuous assignment) in “asu.v” to implement the adder-shifter unit. Describe the mode multiplexer by using continuous assignment in “asu.v.” To run a simulation, **you need to put the related files in the same folder** and use the following command:

```
ncverilog asu_test.v asu.v adder.v barrel_shifter.v +access+r
```

(2) (5%)

Specify 2.5ns delay on the mode multiplexer in "asu.v" and save the file as "asu_gate.v". Use the gate-level design with specified delays in 1-(3) and 2-(3) to run the simulation again. Calculate the sum of delay on the critical path and verify the timing of your design by this command:

```
ncverilog asu_gate_test.v asu_gate.v adder_gate.v barrel_shifter_gate.v
+access+r
```

Show your waveform result in the report.

(3) (5%)

Based on the result of the previous problems, the critical path may be on the adder or the barrel-shifter, followed by the mode multiplexer. Try to optimize the slower part with different structure (*e.g.* Carry Look Ahead Adder) in **"adder_gate_opt.v" or "barrel_shifter_gate_opt.v"**. Calculate the sum of the delay on the critical path of the optimized design. Verify your result simulation and show your waveform result in the report.

```
ncverilog asu_gate_test.v asu_gate.v adder_gate_opt.v barrel_shifter_gate
_opt.v +access+r
```

You need to describe how you optimize your design in the report.

(4) (5%)

Describe how to calculate unsigned multiplication with the adder-shifter unit. You can assume there are other registers for storing temporary value.

Submission requirement:

1. **The homework is for your own practice, and no submission is required.**