CS 211: Computer Architecture, Spring 2015
Programming Assignment 4: Circuit Simulator in C

Instructor: Prof. Santosh Nagarakatte

Due: May 3rd, 2015 at 5pm.

1 Introduction

This assignment is designed to get you experience in C programming while also increasing your understanding of circuits. You will be writing a C program to simulate the output of the circuits for both combinational and sequential circuits. There are two parts to this programming assignment. The first part requires you to simulate the combinational circuits, whose outputs are a function of the inputs. The first part of the assignment is compulsory. The second part of this assignment is extra credit. The second part requires you to simulate circuits that have a combinational logic along with *D flipflops*. The outputs of such circuits are function of the inputs and the previous state.

This assignment requires reasonable amount of programming and hence we advise to get started immediately.

2 Simulate Combinational Circuits (100 points)

In the first part, you have to write a C program to produce outputs for a given input circuit description. Your program should be named *comb* and takes two file names as command line arguments: the name of the circuit description file and name of the file with values for the input variables.

Circuit Description Input

The circuit description file provides the description of the circuit. The circuit description file consists of directives for specifying input variables, outputs of the circuits, and the circuit building blocks.

The input variables used in the circuit is provided using the **INPUTVAR** directive in the circuit description file. The **INPUTVAR** directive is followed by the number of input variables and the names of the input variables.

An example specification of the inputs for a circut with three input variables: A, B, C is as follows:

```
INPUTVAR 3 A B C
```

The outputs produced by the circut is specified using the **OUTPUTVAR** directive. The **OUTPUTVAR** directive is followed by the number of outputs and the names of the outputs.

An example specification of the circuit with output P is as follows:

```
OUTPUTVAR 1 P
```

The circuits used in the first part of this assignment will be built using the following building blocks: **NOT**, **AND**, **OR**, **DECODER**, and **MULTIPLEXER**.

The building blocks can produce temporary variables as outputs. Further, these building blocks can use either the input variables, temporary variables, a boolean '1' or a '0' as input.

All the temporary variables will be named with lower case alphabets (i.e., a, b, c, ...).

The specification of each building block is as follows:

- **NOT**: This directive represents the *not* gate in logic design. The directive is followed by the name of an input and the name of an output.

  An example circuit for a NOT gate ($B = \bar{A}$) is as follows.

  ```
  NOT A B
  ```

- **AND**: This directive represents the *and* gate in logic design. The directive is followed by the names of the two inputs and the name of the output.

  An example circuit for an AND gate ($C = A.B$) is as follows:

  ```
  AND A B C
  ```

- **OR**: This directive represents the *or* gate in logic design. The directive is followed by the names of the two inputs and the name of the output.

  An example circuit for an OR gate ($C = A + B$) is as follows:

  ```
  OR A B C
  ```

- **DECODER**: This directive represents the *decoder* in logic design. The directive is followed by the number of inputs, names of the inputs, and the names of the outputs. The output are ordered in **gray code** sequence.

  An example decoder with two inputs A and B is specified as follows:

  ```
  DECODER 2 A B P Q R S
  ```

  P represents the $\bar{A}\bar{B}$ output of the decoder, Q represents the $\bar{A}B$ output of the decoder, R represents the AB output of the decoder, S represents the $A\bar{B}$ output of the decoder. Note that the outputs of the decoder (i.e., P, Q, R, and S) are in gray code sequence.

  MULTIPLEXER: This directive represents the *multiplexer* in logic design. The directive is followed by the number of inputs, names of the inputs, names of the selectors, and the name of the output. The inputs are ordered in **gray code** sequence.

  A multiplexer implementing a AND gate ($P = A.B$) using a 4:1 multiplexer is specified as follows:

```
MULTIPLEXER 4 0 0 1 0 A B P
```

The above description states that there are 4 inputs to the multiplexer. The four inputs to the multiplexer in gray code sequence are 0 0 1 0 respectively. The two selector input signals are A and B. The name of the output is P.

Input Values File

The second argument to the program is a input values file. Each line of the input values file is a assignment of the values to the variables in the INPUTVAR specification in the circuit description file.

An example input values file for a circuit that has three input variables is as follows:

```
1 0 1
1 1 1
```

The program should output the values produced by the circuit for the output variables specified in the circuit description file.

Example Execution 1

A sample circuit description file named circuit.txt for the circuit $Q = AB + AC$ is as as follows:

```
INPUTVAR 3 A B C
OUTPUTVAR 1 Q
AND A B w
AND A C x
OR w x Q
```

The sample input file named input.txt for the above circuit is as follows:

```
1 0 1
0 0 1
```

On executing the program with the above circuit description file and input file, your program should produce the following output (one line for each input in the input file).

```
./comb circuit.txt input.txt
1
0
```

The output of the circuit $Q = AB + AC$ when A is 1, B is 0 and C is 1 is 1. Hence the first line is 1 in the output. Similarly, the output of the circuit is 0 when A is 0, B is 0, and C is 1.

Note the use of the temporary variables in the circuit description file to represent intermediate outputs.

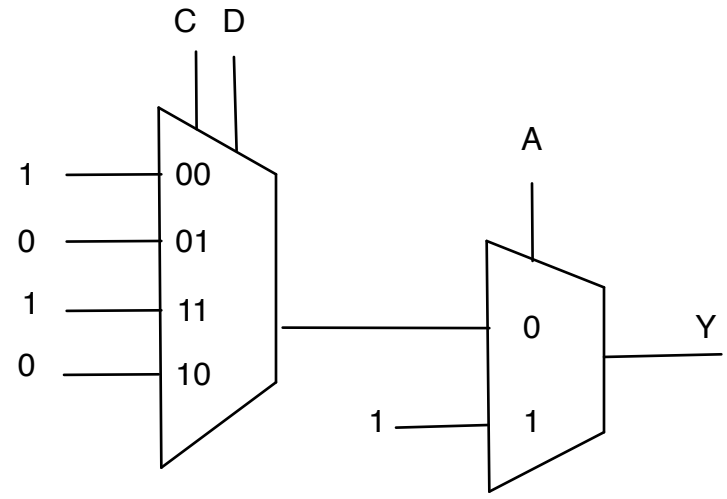

Figure 1: Circuit with Multiplexers from Homework 1

2.1 Example Execution 2

The circuit description file (hw1-circuit.txt) for the circuit in Figure 1 from Homework 1 is as follows:

```
INPUTVAR 3 A C D
OUTPUTVAR 1 Y
MULTIPLEXER 4 1 0 1 0 C D w
MULTIPLEXER 2 w 1 A Y
```

The input value file (hw1-input.txt) for the circuit is as follow:

```
1 1 1
1 0 0
0 1 0
```

When we execute the program the output should be as follows:

```
./comb hw1-circuit.txt hw1-input.txt
1
1
0
```

3 Simulate Sequential Circuits (Extra Credit: 50 points)

In this extra credit part, you have to write a C program to produce outputs for a given circuit description that include sequential logic elements namely *D-flipflops*.

In addition to the circuit building blocks described in Section 2, this part of the assignment has an additional building block called the **DFLIPFLOP** and one of the input variables is specified as the clock.

One of the input variables is specified as the clock variable using the **CLOCK** directive. For example, input variable C is specified as the CLOCK variable below as follows:

```
INPUTVAR 3 A C
OUTPUTVAR 1 B
CLOCK C
```

The **DFLIPFLOP** directive is followed by name of the input to the D-flipflop, the clock variable, and the output of the D-flipflop.

An example D-flipflop that takes A as input with C as the clock variable and produces B as output with a initial value of 0 is specified as follows:

```
DFLIPFLOP 0 A C B
```

In contrast to the first part, the input description file has a series of 0's and 1's for each input. Each line of the input file has a variable name, number of input values for the variable, and a series of 0's and 1's for the variable. For example, a line of the input file below states that A is the input variable and it has 6 input values for the circuit.

```
A: 6 0 1 1 1 0 1
```

3.1 Example Sequential Circuit (2-bit counter)

Consider designing a sequential circuit for 2-bit counter which resets to 00 after reaching 11. When the clock goes from O to 1, the counter changes value from the previous value to the next value.

The circuit description file (seq-circut.txt) for the circuit is shown below.

```
INPUTVAR 1 C
OUTPUTVAR 2 P Q
CLOCK C
NOT P w
AND w Q x
NOT Q y
AND P y z
OR x z r
DFLIPFLOP 0 w C P
DFLIPFLOP 0 r C Q
```

A sample input file (seq-input.txt) for the circuit above is as shown below:

```
C: 8 0 1 0 1 0 1 0 1
```

The output of the circuit when you execute the program is as follows:

```
./seq seq-circuit.txt seq-input.txt
C: 0 1 0 1 0 1 0 1
P: 0 1 1 0 0 1 1 0
Q: 0 0 0 1 1 1 1 0
```

4 Submission

You have to e-submit the assignment using Sakai. Your submission should be a tar file named `pa4.tar`. To create this file, put everything that you are submitting into a directory (folder) named `pa4`. Then, `cd` into the directory containing `pa4` (that is, `pa4`'s parent directory) and run the following command:

```
tar cvf pa4.tar pa4
```

To check that you have correctly created the tar file, you should copy it (`pa4.tar`) into an empty directory and run the following command:

```
tar xvf pa4.tar
```

This should create a directory named `pa4` in the (previously) empty directory.

The `pa4` directory in your tar file must contain 2 subdirectories, one each for each of the parts. The name of the directories should be named comb and seq (in lower case). Each directory should your source files, header files, and a make file).

We will provide a autograder and many more test cases, which can be used to test your submission and your program.

5 Grading Guidelines

This is a large class so that necessarily the most significant part of your grade will be based on programmatic checking of your program. That is, we will build a binary using the Makefile and source code that you submitted, and then test the binary for correct functionality against a set of inputs. Thus:

- You should make sure that we can build your program by just running `make`.
- You should test your code as thoroughly as you can.
- Your program should produce the output following the example format shown in previous sections. Any variation in the output format can result **up to 100% penalty**. There should be no additional information or newline. That means you will probably not get any grade is you forgot to comment out some debugging message.

Be careful to follow all instructions. If something doesn't seem right, ask.