New York University Polytechnic School of Engineering

Electrical and Computer Engineering

Assignment 1

(Assigned Monday 9/21, Due Monday 10/05)

Note About Assignment Submission*: the assignment is meant to be done individually. You are free to discuss with your colleagues, but you must write the code yourself. Any violation of this policy result in an automatic zero on the class.*

Verilog files for your assignment are to be submitted in a single zipped folder on newclasses.nyu.edu before midnight of the due date. The folder will no longer be available after that time for submissions.

You will then fix a time to meet with one of the TAs and demo your solution (after downloading files from the newclasses folder).

P1) ***Huffman Decoder*****.** A Huffman code is a prefix code that assigns a variable length codeword to symbols in a dictionary. In the Table below, the letter A is represented using a single bit 0. The letter B is represented using two bits 10, and so on.

Importantly, a Huffman code is a type of Prefix code. No codeword contains another codeword as a prefix.

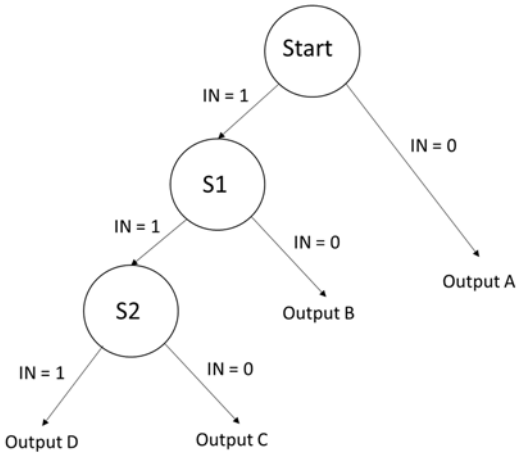
Codeword	Original Word
0	A
10	B
110	C
111	D

Now consider the input sequence of codewords below and how it is decoded.

0 0 1 1 0 1 0 1 1 1 0
A A - - C - B - - D A

Moving from left to right, we look at the current input and match against the decoding table. Our first input is 0 and we find a match, A, so we output it. Our next input is 0 again, so we output A again. The next input is 1. This does not match with any codeword. Neither does 11. Finally, when we see 110, we can output C.

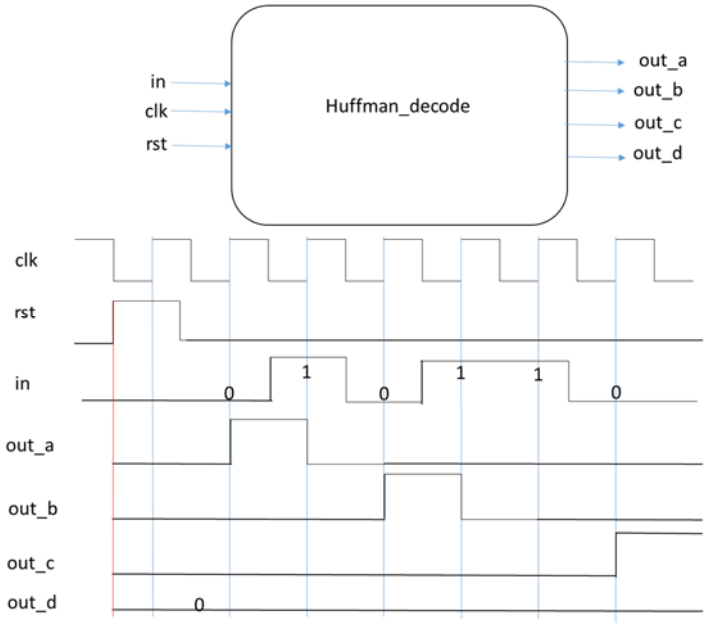
In fact, Huffman decoding works by traversing down a decision tree, as shown below. We start at node marked START (the root) and when we reach a leaf node, we output the corresponding symbol and go back to the root.



Write a Verilog description of a module, with inputs and outputs as shown below, that implements an FSM to decode an input sequence of bits. The table below shows the relationship between the output symbol {-,A,B,C,D} and the Boolean values of the output ports.

A sample input sequence and the corresponding output waveforms are shown for illustration. Note that the outputs only change at positive clock edges. (Hint: is this a Mealy or a Moore machine?).

Output Symbol	out_a	out_b	out_c	out_d
-	0	0	0	0
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1



P2) ***Long division.*** As we discussed in class, the division operation in Verilog is not synthesizable. In this problem, you will implement a finite state machine that that performs division using repeated subtraction.

The pseudo-code for division using repeated subtraction is shown below:

```
// {quotient, remainder} = a/b;
quotient = 0;
while a>=b {
   a = a – b;
   quotient = quotient + 1;
}
remainder = a;
```

Your module must have two inputs a and b with parameterized width W (i.e., a and b should both be W bits long), a clock, a reset input, and must have three outputs: quotient, remainder and a done signal that is asserted when the quotient and remainder are valid. The picture below shows the module inputs and outputs and a sample timing diagram.

You are free to use any approach to implement your FSM (for example, the data-path/control-path approach discussed in class).