1.4 Day 4 - RTL Design

Today we will undertand what RTL design is. RTL stands for "Register-Transfer-Level" and in an RTL design, we use components such as registers, adders, multipliers, comparators, multiplexers, etc. In an RTL design, we have two components - (a) the data path which includes an interconnection of RTL components and (b) the controller which sequences the operations and data transfers in the data path so that the circuit does something meaningful. At the end of the exercise, you should be able to appreciate how to construct RTL circuits, how to design a controller, how to verify/debug an RTL circuit, and what is meant by timing analysis.

1.4.1 Problem Statement

Assume that we must multiply two 4-bit unsigned numbers. Each number can range from 0 to 15. The maximum value of the producet is 225 and it requires 8 bits for its representation. The shift-and-add method of multiplication can be used for multiplying two 4-bit numbers in 4 steps.

The datapath for the multiplier includes three 4-bit registers A, Q, and M and a 4-bit adder. A flip-flop C is used to store the carry resulting from the addition. The algorithm for the multiplication is shown below. At the end of 4 steps, the product is found in A,Q.

```
A = 0;
Q = Multiplier;
M = Multiplicand;
COUNT = 0;
WHILE COUNT < 4 DO
      C=0
      IF Q(0) == 1 THEN
            C,A = A+M;
      ENDIF
      A,Q = C,A,Q[3:1];
      COUNT=COUNT+1
ENDWHILE
```

An example of multplication 15x15 is shown below. The result is 11100001.

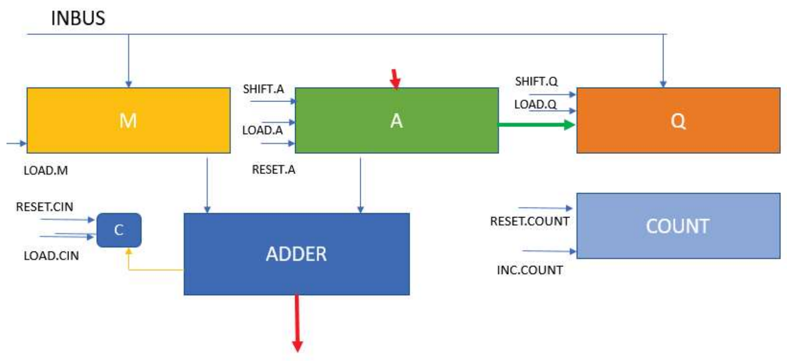
```
    A               Q
[0, 0, 0, 0] [1, 1, 1, 1]
[0, 1, 1, 1] [1, 1, 1, 1]
[1, 0, 1, 1] [0, 1, 1, 1]
[1, 1, 0, 1] [0, 0, 1, 1]
[1, 1, 1, 0] [0, 0, 0, 1]
```

1.4.2 Assignment

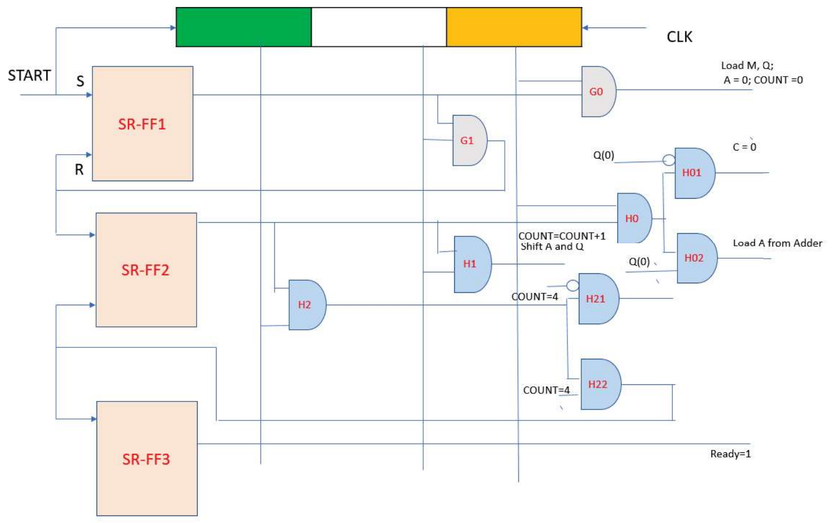
1. Understand the shift-and-add multiplier. Try multiplying 15 and 5 and verify that the result is correct. Here is the pseudocode of the algorithm.

```
STEP        ACTIONS
1             Set A = 0; Load Q and M. Set COUNT = 0, READY=0
2             ACTION IF Q(0) == 1 :  Load adder output to A. Load the carry into C.
2             ACTION IF Q(0) == 0 :  Set C = 0
3             RIGHT SHIFT C,A,Q; Set COUNT  = COUNT + 1
4             ACTION IF COUNT < 4: GOTO STEP 2
4             ACTION IF COUNT == 4: GOTO STEP 5
5             SET READY=1; GOTO Step 5.
```

2. Write down the block diagram of the multiplier. Start with registers A, Q, and M. Assume that an 8-bit input bus called INBUS contains the data to be loaded in Q and M. Assume that the 4 LSB contain the multiplier (Q) and the 4 MSB contain the multiplicand (M). Show a 1-bit register called C. A sketch of the data path is shown below.



3. Understand the design of a controller. The algorithm has three phases - an initialization phase, a loop, and a finalization phase. In the initialization phase, we clear A register, load Q and M registers, and clear the COUNT. We also set READY output to 0. The second phase can be divided into three steps. In Step 1, we check if Q(0) is 0 or 1 and take action. In Step 2, we perform a right-shift operation. In the third step, we check if COUNT has become 4. A sketch of the controller is shown below. The SR flip-flops on the left hand side implement the phases. The START signal sets the Set-Reset flip-flop 1. It also initializes the ring counter to 001. G0 is high when we are in phase 1, step 1. G1 marks the beginning of step 2 in Phase 1. We use G1 to go to Phase 2. H0 marks the beginning of Step 1, Phase 2. H01 and H02 tell us if Q(0) is 0 or 1. Carefully go through the controller design and ensure that you understand it.



4. Make a list of all the components you need to make the multiplier. Work as a team and design different parts of multiplier by distributing the work across the team. Integrate the data path and control path.

5. Create a test case to multiply 15 and 5. Verify your result.

6. Think of how many verification test cases you will need to be convinced that your design is working.