Embedded Real-Time Systems (AME 3623) Homework 2 Solutions

March 6, 2005

Question 1

Consider the following binary number:

$0b01001101$

1. What is the decimal equivalent?

   $1 + 4 + 8 + 64 = 77$

2. What is the octal equivalent?

   Collect the digits in the original binary number into groups of 3 (starting from the right) and then convert the individual groups into octal digits.

   0115

   Note the first "0" simply indicates that this is an octal number.

3. What is the two's complement of the number (in binary)?

   In this case "two's complement" refers to the additive inverse.

   10110011

Question 2

Consider the following binary number:
  $0b10110001$

1. Assuming that this is a two's complement number, what is the decimal equivalent?

   Since this is a two's complement number, we will look at the most significant bit to determine it's sign. This bit is "1", so we have a negative number. First, let's convert this to a positive number, then read out the decimal value, and then convert it back to a negative number (in decimal).

   Taking the additive inverse of the number, we have:

   01001111

   Converting to decimal, we have:

   $1 + 2 + 4 + 8 + 64 = 79$

   Taking the additive inverse again, we have:

   -79

2. What is the binary result of multiplying this number by 2?

   Multiplying by 2 is simply a matter of shifting left. But - since this is a two's complement number, we have to preserve the sign bit (the most significant bit). So, we have:

   11100010

   Which *should* be -158 - but we have "blown out" our representation (with 8-bit two's complement, we can only represent numbers as low as -128).

3. Assuming that this is an unsigned number, what is the decimal equivalent?

   $128 + 32 + 16 + 1 = 177$

4. What is the binary result of multiplying this number by 2? State any assumptions that you must make.

Multiplication by two of an unsigned binary number is implemented as a left shift.

Assuming that we have a carry bit available:

101100010

Assuming that we do not have a carry bit available:

01100010

Question 3

Given the following two numbers in two's complement:

$X = 0xCF$ and $Y = 0x42$

1. What is the binary equivalent of X?

   11001111

2. The binary equivalent of Y?

   01000010

3. What is the binary result of adding X and Y? (show your work)

   ```
    11001111
   +01000010
   ```

   100010001

   But: these are two's complement numbers, so we drop the carry bit:

   00010001

4. What is the decimal equivalent?

   $16 + 1 = 17$

5. What is the binary equivalent of subtracting Y from X? (show your work)

   First, let's take the additive inverse of Y:

   10111110

Now, add this with X:

```
  11001111
+10111110
110001101
```

But: this is a two's complement number, so we drop the most significant bit:

10001101

6. What is the hex equivalent?

   $8D$

Question 4

Given the following decimal number: 53

1. What is the binary equivalent of this number?

   We will assume (for the next stages) that we are dealing with 8-bit two's complement numbers.

   00110101

2. What is the hexadecimal result of subtracting 053 from this number?

   053 is octal for binary 00101011

   The additive inverse of this number is:

   11010101

   Now adding the two numbers:

   ```
     00110101
   +11010101
   100001010
   ```

   Since this is a two's complement number, we drop the carry bit, which leaves:

   00001010

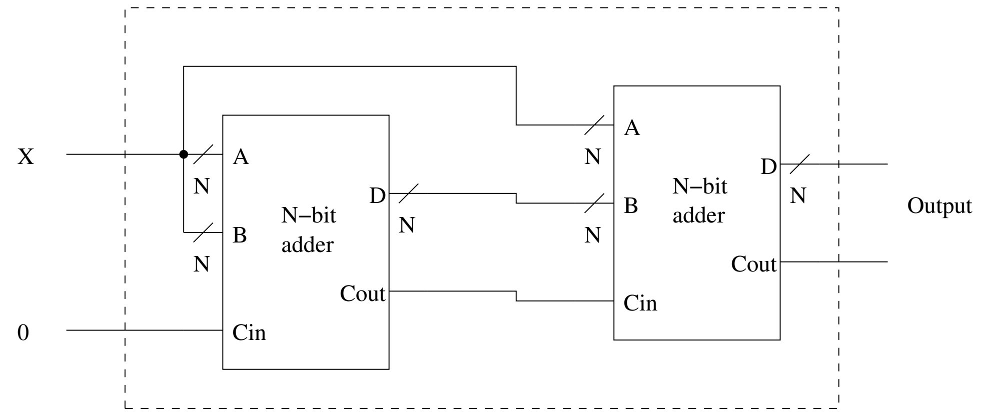
   and in hex, this is:

   $0xA$

Question 5

Assume a "black-box" N-bit adder. Design a circuit that multiplies a number "X" by 3.

We can "unroll" this multiplication by adding X to itself two additional times. Here is a circuit that performs this:

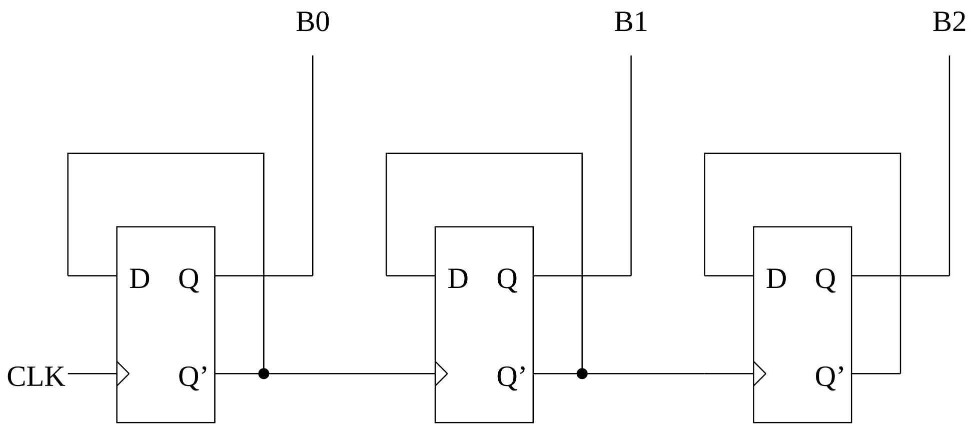


Question 6

In class we designed a 3-bit ripple counter (yes, the ripple counter, not the sequential counter).

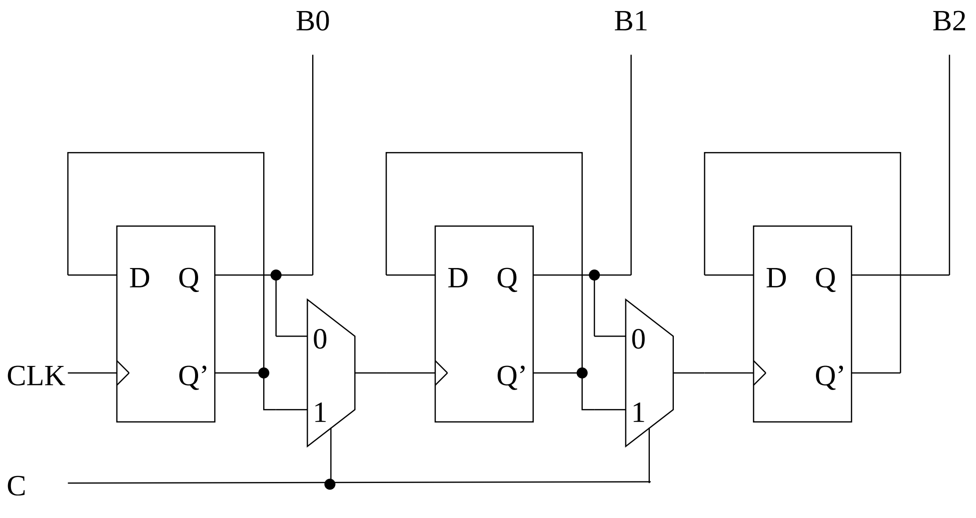
1. Modify the circuit so that the counter counts down with each downward edge of the clock.

   We want $B1$ to change state every time $B0$ transitions from low to high. Likewise, we want $B2$ to change state every time $B1$ goes from low to high. We therefore want to modify our ripple counter so that each stage receives the inverted state from the previous stage.

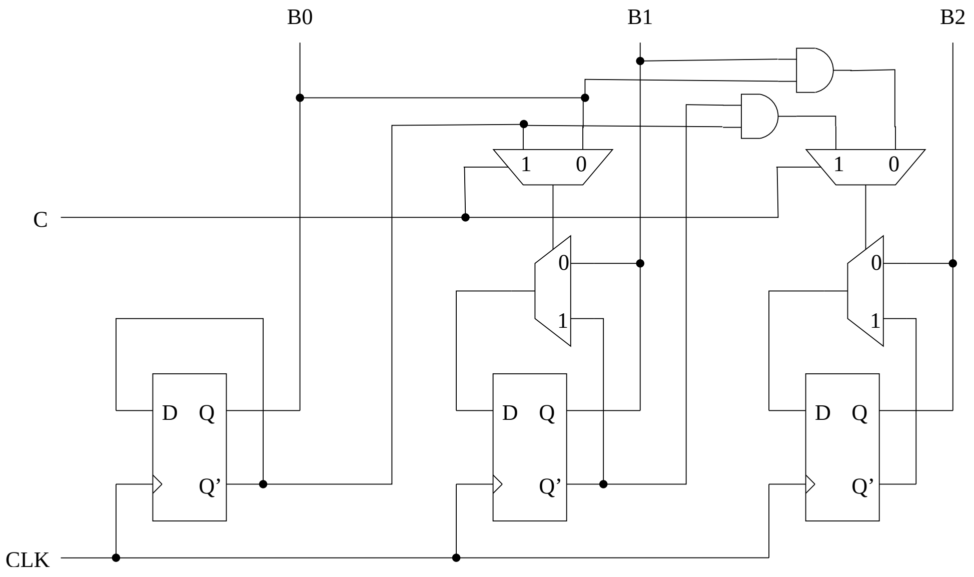


2. Modify the circuit to add one additional "control" input bit. When $C = 0$, the counter must count up with every clock cycle; when $C = 1$, the counter must count down. You may use any macro-level components that we designed in class (i.e., you do not need to restrict yourself to using basic gates).

   We will assume that the control bit is set up before we perform any of the counting (once counting starts, we will not change the state of the control bit). With this assumption, it is a matter of selecting the appropriate clock input to $B1$ and $B2$. We will use a multiplexer to do this:



We do not allow the control bit to change state once counting begins because simply changing this bit's state can result in an apparent clock pulse input into the flip-flops. Here is an alternative design that solves this problem. We take inspiration from our memory element design from class:



Question 7

In class, we designed a circuit that added two N-bit binary numbers which was composed of N 1-bit adders. Design a similar circuit that takes as input a 3-bit number and produces as output the two's complement of the number.

1. For the 1-bit case, show the truth table.

   In general, taking the additive inverse of a two's complement number, we will invert the individual bits and then add 1 to the result. For any individual bit (i), we will have as input the bit value (we call this A_i) and a carry from the previous bit (C_i^{in}). As output, we will have the bit value (B_i) and a carry to the next stage (C_i^{out}). In short, we will add C_i^{in} and $\overline{A_i}$ to yield the output. Here is the truth table:

C_i^{in}	A_i	C_i^{out}	B_i
0	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1

2. Show the corresponding Karnaugh map.

   Since this is a 2-input function, this is not a terribly interesting Karnaugh map, but here is the one for C_i^{out}:

C_i^{in} A_i	0	1
0	0	1
1	0	0

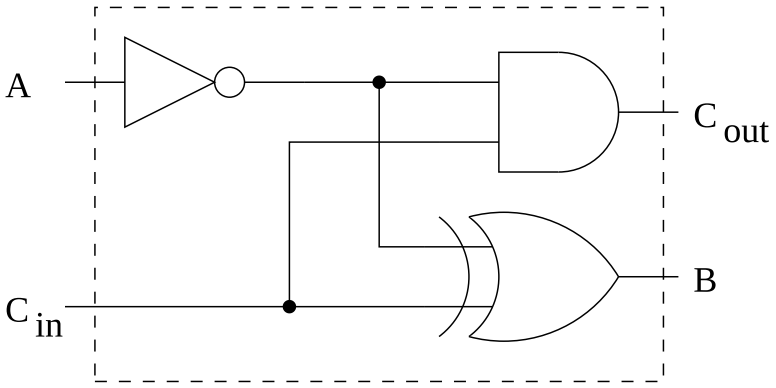
   So: $C_i^{out} = \overline{A_i} C_i^{in}$

   And for B_i:

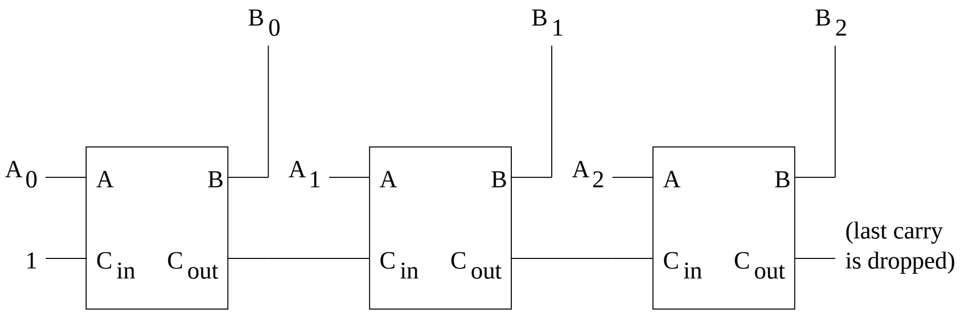
A_i \ C_i^{in}	0	1
0	1	0
1	0	1

So: $B_i = \overline{A_i} \oplus C_i^{in}$

3. Show the 1-bit circuit.



4. Assuming that this 1-bit circuit is a "black box," show the 3-bit circuit.



Question 8

How much time did you spend on this homework assignment?