CS 51 Homework 2

Jessie Li

September 29, 2024

1.

A. `0xA023` $= 10 \cdot 16^3 + 0 \cdot 16^2 + 2 \cdot 16^1 + 3 \cdot 16^0 = 40995$

B. `0x0BED` $= 0 \cdot 16^3 + 11 \cdot 16^2 + 14 \cdot 16^1 + 13 \cdot 16^0 = 3053$

C. `0x12EF` $= 1 \cdot 16^3 + 2 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0 = 4847$

D. `0x2100` $= 2 \cdot 16^3 + 1 \cdot 16^2 + 0 \cdot 16^1 + 0 \cdot 16^0 = 8448$

E. `0xC003` $= 12 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + 3 \cdot 16^0 = 49155$

2.

A. $257 = 1 \cdot 16^2 + 1 \cdot 16^0 =$ `0x0101`

B. $-17,203 =$ `0xBCCD`

$17,203 = 4 \cdot 16^3 + 3 \cdot 16^2 + 3 \cdot 16^1 + 3 \cdot 16^0 =$ `0x4333`. Inverting each bit by subtracting it from F, then adding 1 gives `0xBCCC` + 1 = `0xBCCD`. Confirm that `0xBCCD` + `0x4333` = `0x0000`.

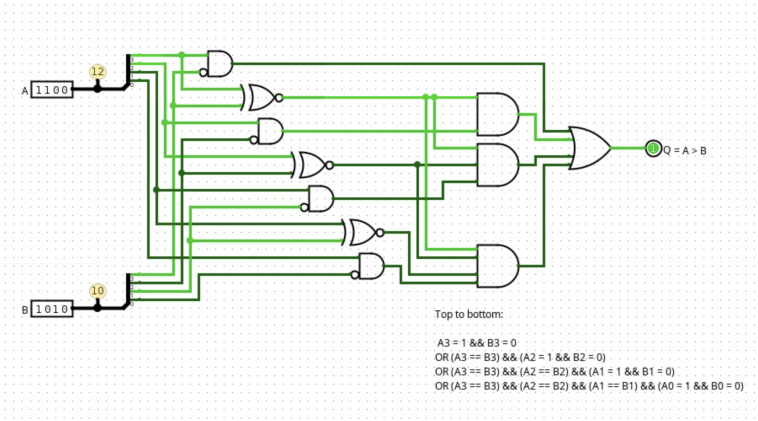
C. $21,034 = 5 \cdot 16^3 + 2 \cdot 16^2 + 2 \cdot 16^1 + 10 \cdot 16^0 =$ `0x522A`

D. $-916 =$ `0xFC6C`

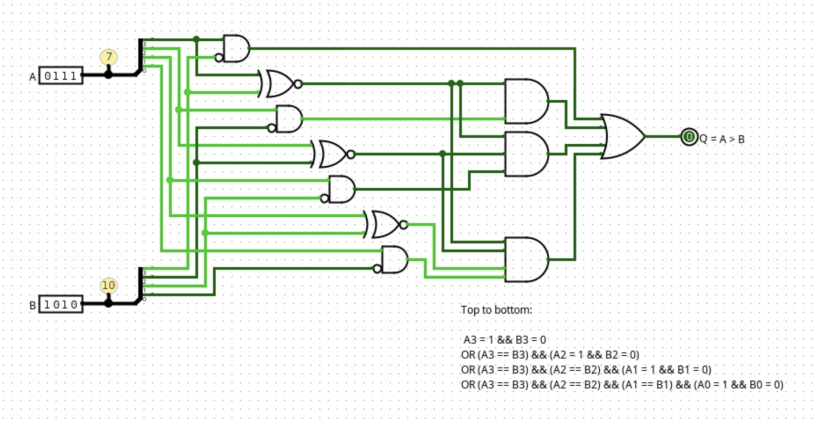
$916 = 3 \cdot 16^2 + 9 \cdot 16^1 + 4 \cdot 16^0 =$ `0x0394`. Then, `0xFFFF` − `0x0394` + 1 = `0xFC6B` + 1 = `0xFC6C`.

E. $45 = 2 \cdot 16^1 + 13 \cdot 16^0 =$ `0x002D`

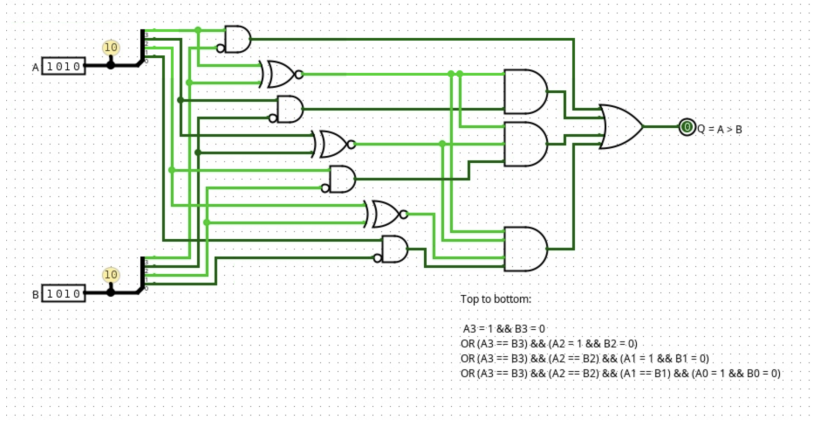
3.



(a) Circuit outputs 1 if $A > B$.



(b) Circuit outputs 0 if $A < B$.



(c) Circuit outputs 0 if $A = B$.

Testing

To test, I wrote a shell script `test_q3.sh` to create a test vector `test_q3.txt` for my circuit and confirmed that the output was correct for all 256 combinations of 4-bit inputs A and B (16 values for A $\times$ 16 values for B).

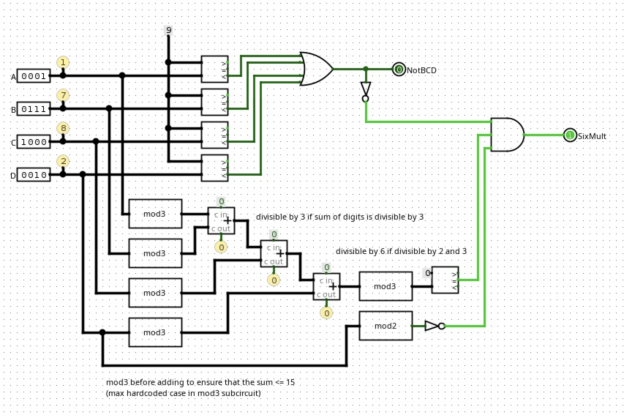
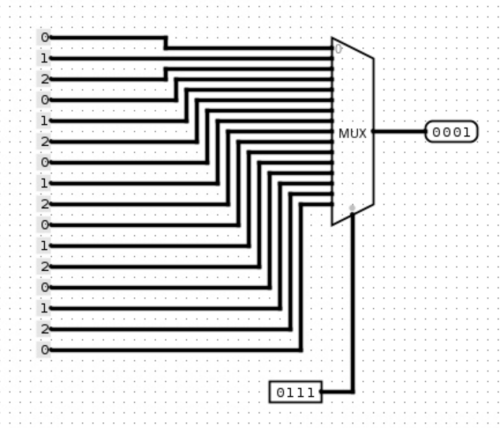


Figure 2: For `1 7 8 2`, the circuit outputs $NotBCD = 0$ and $SixMult = 1$.

4.

The circuit takes four 4-bit inputs, A, B, C, and D and has two 1-bit outputs, $NotBCD$ and $SixMult$. The top half checks whether any of the four inputs is not a BCD digit. The bottom half checks whether the whole input, read as a decimal number, is a multiple of six by checking whether the number is divisible by both 2 and 3. Divisibility by 2 is easily determined by checking the least significant bit of input D; if 0, the number is even, otherwise odd. To determine whether the number is divisible by 3, the circuit checks whether the sum of its digits is divisible by 3. Modulo 3 is applied to each digit before summing to ensure that the sum will not exceed 15 (though the maximum sum after modulo 3 is actually $8 = 2 + 2 + 2 + 2$), the largest hard-coded input to the `mod3` subcircuit. If divisible by 3, the sum modulo 3 will be 0.



(a) `mod3` subcircuit.

(b) `mod2` subcircuit.

Testing

I wrote a script `test_q4.sh` to output 256 randomly generated test cases to a test vector file `test_q4.txt` and added 12 hand-picked cases:

- 5 for $NotBCD = 0, SixMult = 1$:
  - Zero: `0 0 0 0`
  - Sum of digits > 8: `3 0 1 8`
  - Maximum sum of digits: `9 9 9 6`
  - Individual digits not divisible by 3: `1 2 1 2`
  - Random: `0 4 5 0`
- 5 for $NotBCD = 0, SixMult = 0$:
  - Minimum sum of digits: `0 0 0 1`
  - Maximum sum of digits: `9 9 9 9`
  - Divides 3 but not 2: `5 9 0 1`
  - Divides 2 but not 3: `1 2 3 4`
  - Random: `7 0 2 5`
- 2 for $NotBCD = 1, SixMult = 0$
  - Hex is divisible by 6: `1 7 7 C`
  - Random: `A 9 1 E`