Wang, He Qian – 260688073
Yadollahi, Abbas – 260680343

October 13, 2017
Team 06

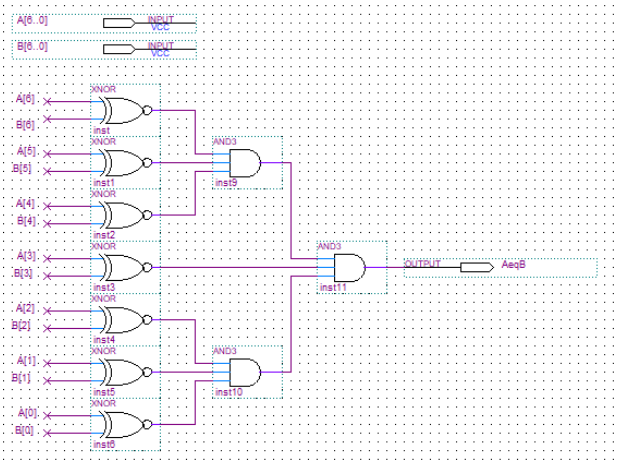
Lab 1 Report: g06 Modulo 13

PART 1: 7-BIT COMPARATOR

This first part of the lab required us to build a 7-bit comparator using 7 instances of XNOR gates. The outputs will then be funneled into three 3-input AND gates to determine whether each bit of the two 7-bit inputs A and B are the same. The inputs of each XNOR gate correspond to the same position bit of the inputs A and B. For example, the first bit of A and first bit of B are the inputs to the first XNOR gate, second bit of A and second bit of B are the inputs to the second XNOR gate and so on. The comparator will output 1 if A=B and 0 otherwise.

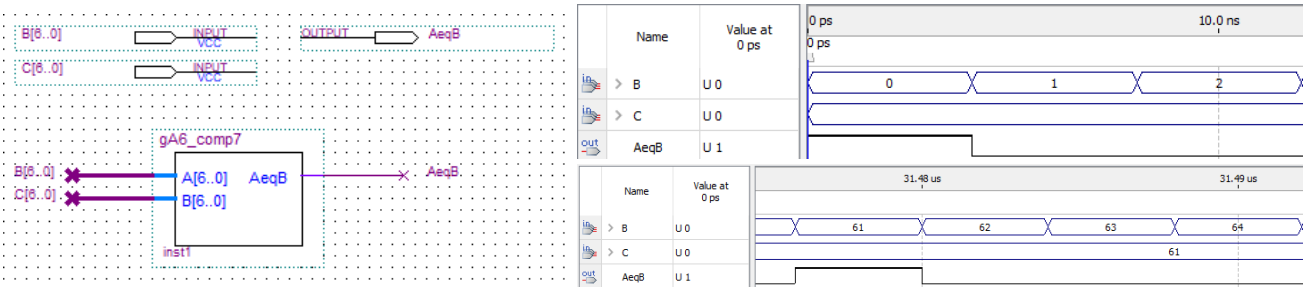
Schematics

7-Bit Comparator



The simulation required us to check every single possible combination of the bits A and B. With 7 bits giving 128 possible inputs (0 to 127), we simulate A changing every 4ns and B every 512ns (4ns*128). Like so, we cover every possible value of A with one value of B. To cover the other values of B, we run the operation a total of 128 times. In total, the simulation will take 65536ns.

Waveforms



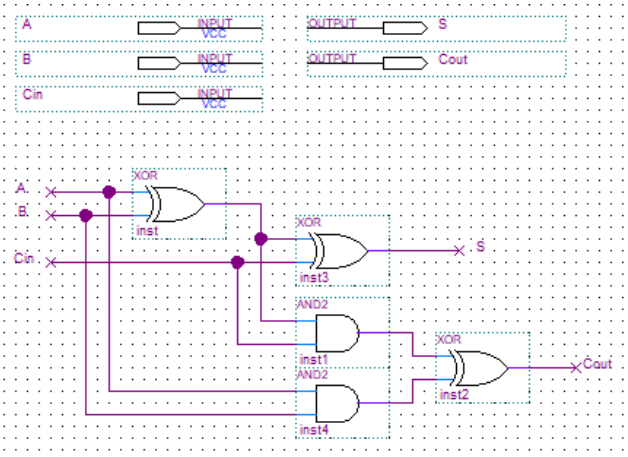
Here we can see the results of the simulation on the 7-bit comparator illustrated by the schematics on the left. According to the two waveform graphs on the right, the outputs are 1 when the inputs A and B match (first A=B=1 and second A=B=61) and 0 otherwise, which follows our desired result.

PART 2: MODULO 13

The second part of lab 1 consisted of building a digital circuit which could output the result of a modulo_13 operation. The circuit is given a 6-bit input A and should output a 4-bit output. The algorithm we used for this modulo circuit was to take the 6-bit input and subtract the highest multiple of 13 smaller than A. To do so, we needed a few smaller scale circuits.

Schematics

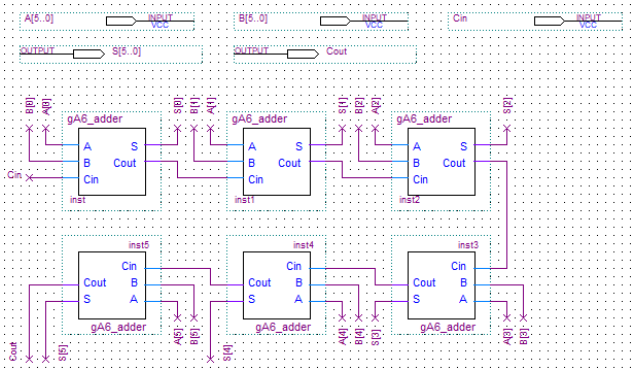
1-Bit Full Adder



The base of the entire modulo circuit was the 1-bit full adder. The adder takes three 1-bit inputs (A, B, C_{in}) and outputs two bits (S and C_{out}). The circuit would add the three input bits and output the sum to S and any carry to C_{out}. To build the full adder, we make use of 3 XOR gates and 2 2-input AND. Through testing, we get the following truth table. All outputs are valid.

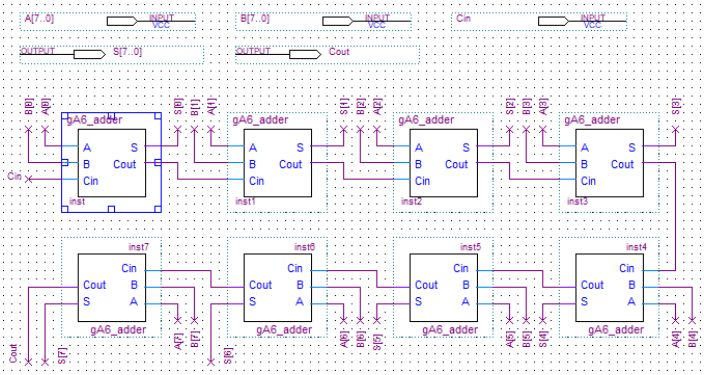
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

6-Bit Full Adder



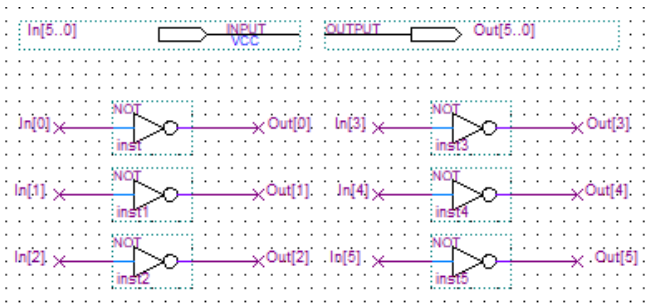
To design our 6-bit adder, we cascaded 6 full adders together by attaching the C_{out} of the most significant adder to the C_{in} of the next most significant adder, and so forth until the last adder. The output C_{out} of the last adder is the output C_{out} of the entire 6-bit full adder. The output S of each adder is one of the bits of the 6-bit output of the 6-bit full adder. The input of each adder is one bit of the two 6-bit inputs A and B, starting from the most significant bit.

8-Bit Full Adder



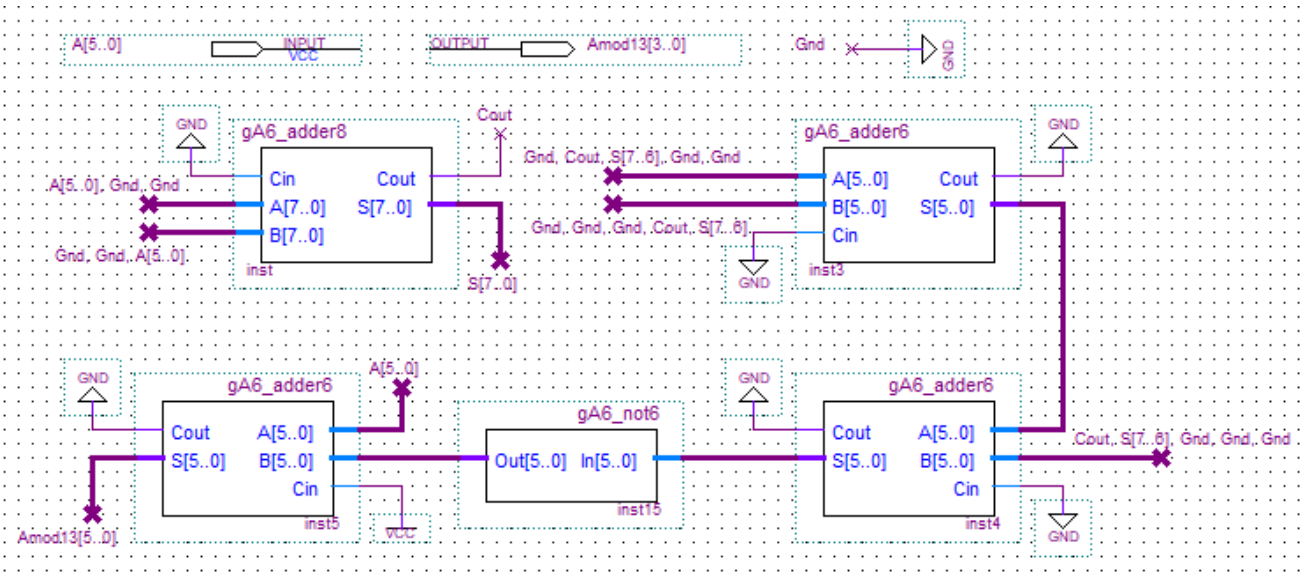
To design our 8-bit adder, we cascaded 8 full adders together by attaching the C_{out} of the most significant adder to the C_{in} of the next most significant adder, and so forth until the last adder. The output C_{out} of the last adder is the output C_{out} of the entire 8-bit full adder. The output S of each adder is one of the bits of the 8-bit output of the 8-bit full adder. The input of each adder is one bit of the two 8-bit inputs A and B, starting from the most significant bit.

6-Bit Inverter



The 6-bit not circuit takes a 6-bit input and uses 6 NOT gates to invert every single bit of the input. The output is the 6-bit input inverted.

6-Bit Modulo 13



The equation we use for modulo 13 on a 6-bit input A goes like so: $Mod13(A) = A - floor\left(\frac{A}{13}\right) * 13$

To perform this using a digital logic circuit, we had to be creative.

Instead of dividing A by 13, which would require an extremely large and complicated circuit, we decided to multiply A by 1/13. To make things easier, we approximate the denominator of 1/13 to the nearest power of 2, giving us 5/64. Using the approximation, we can use shifting to find the result of A/13.

1- A*5

The first step in the modulo 13 circuit is to multiply A by 5. To do so, we look at 5 in binary, which is 101_2. Using this information, we see that the multiplication by 5 is simply the addition of the 6-bit input A shifted left twice with the unshifted 6-bit input (100_2*A + 001_2*A). For this operation, we used an 8-bit adder since the shifted 6-bit input now has a total of 8 bits. The input A of the 8-bit adder will have the two least significant bits (A[0] and A[1]) grounded thanks to the shift left by two digits; the remaining 6 bits will correspond to the 6-bit input A. The input B of the 8-bit adder will have the two most significant bits (A[7] and A[6]) grounded to pad the input since it requires 8 bits; the remaining 6 bits will be from the 6-bit input A. The input C_{in} will be grounded since there is no carry from the start. The adder will then sum the inputs A and B and we will get our multiplication by 5. Due to the output C_{out}, the overall output will be a 9-bit number.

2- floor(5*A)/64

The second step requires us to divide the result of the input multiplied by 5 by 64. The highest value that the 6 digit input A can reach after multiplying it by 5 and dividing by 64 is 4.92. Since we are flooring the result of the division, we can assume that the highest value will in fact be 4. As a result, it is safe to perform the division by 64 by shifting the output of the 8-bit adder right by 6 bits (64 is represented by 1000000_2 in binary, thus 2^6). The result will be a 3-bit number, represented by C_{out} and the two most significant bits of the 8-bit adder's output (S[7..6]), which has a maximum value of 4. Thus, there is no need to use any circuit elements for the division by 64 and the flooring operation comes for free.

3- floor(5*A/64)*13

The third step is to multiply by the floored result of A/13 by 13, which is represented as 1101_2 in binary. To perform this multiplication, we require two separate steps of addition, all while using Cout and S[7,6] which as the 3-bit output of the floor operation which we will call F_{out}. In brief, the entire multiplication operation will see the addition of the 3-bit number F_{out} unshifted with F_{out} shifted left two bits and with F_{out} shifted left three bits, all with the help of two 6-bit full adders.

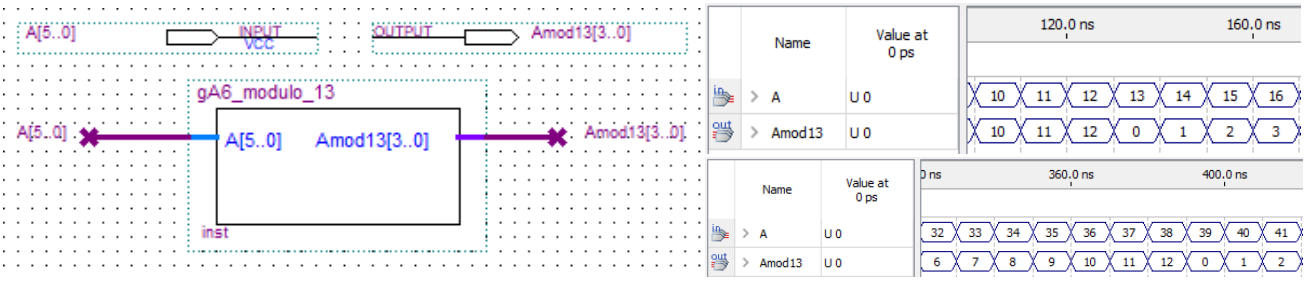
We begin the first addition with F_{out} unshifted as input B, F_{out} shifted left twice as input A and, since there is no carry in, the input C_{in} is grounded. Both inputs A and B are padded with zeros on the left to become a 6-bit number. After the first addition is finished, we redirect the output of the first 6-bit adder into the input A of the second 6-bit adder with input B being F_{out} shifted left three bits. Since there is no possible way that the output of the first 6-bit adder exceeds 6 bits, we ground the output C_{out} of the first 6-bit adder and the input C_{in} of the second 6-bit adder (100_2 + 10000_2 < 1000000_2). The output of the second 6-bit adder will then be the result of floor(A/13)*13. Once again, we ground the output C_{out} since the result will not exceed a 6-bit number.

4- A - floor(5*A/64)*13

The final remaining step is to subtract the initial input A by the result of the floored multiplication. The simplest way of performing this is by using two's complement and thus adding the initial 6-bit input A and the inverted 6-bit output of the 6-bit adder.

We begin by inverting every bit of the output of the 6-bit adder by using the 6-bit inverter. Afterwards, we redirect the output of the 6-bit inverter into the input B of the last 6-bit adder and use the initial input A for the input A of the last 6-bit adder. A voltage source (1) is attached to the input C_{in} since we need a carry in to perform two's complement. The final output Amod13 will be a 4-bit number equal to the result of our initial equation Mod13(A). We can ground the output C_{out} since any carry out should be discarded during two's complement subtraction.

Waveform



To test the modulo 13 circuit, we created a simulation by having the 6-bit input A varying from 0 to 63. As we can see from the results of the simulation waveform graphs on the right, the output Amod13 does in fact correspond to the modulo 13 of A. Using the equation $Mod13(A) = A - floor\left(\frac{A}{13}\right) * 13$, we can validate all points to show our modulo 13 circuit follows our desired results.