Group# 39 Saleh Bakhit (260632353) Adham El-Khouly (260601197)

ECSE 323 – Digital System Design
g39_Modulo_13

The Modulo_13 circuit takes an input A and produces an output Amod13 representing the remainder of its division by 13. It also has an output floor13 representing the largest integer $\leq \frac{A}{13}$.

A: 6-bit input
Amod13: 4-bit output
floor13: 3-bit output

Algorithm:

To find the remainder, we will use the formula: $AmodX = A - floor\left[\frac{A}{X}\right] * X$

$$Amod13 = A - floor\left[\frac{A}{13}\right] * 13$$

$= A - shift\_right[A * 5, 6] * 13$ → $floor\left[\frac{A}{13}\right] \approx$ A*5 shifted 6-bits to the right

$= A - shift\_right[(A * 4 + A), 6] * 13$ → A*5 = A*4 + A*1

$= A - shift\_right[(shift\_left(A, 2) + A), 6] * 13$ → A*4 = A shifted 2-bits to the left

Let $floor13 = shift\_right[(shift\_left(A, 2) + A), 6]$
Then, $floor13 * 13 = floor13 * 2^3 + floor13 * 2^2 + floor13 * 2^0$
So, $floor13 * 13 = shift\_left(floor13, 3) + shift\_left(floor13, 2) + floor13$

→ $Amod13 = A - [shift\_left(floor13, 3) + shift\_left(floor13, 2) + floor13]$

Algorithm:

1. $A * 5 = shift\_left(A, 2) + A$
2. $floor13 = shift\_right[A * 5, 6]$
3. $shift\_left[floor13, 3] + shift\_left[floor13, 2]$
4. $floor13 * 13 = result_3 + floor13$
5. $Amod13 = A - floor13 * 13 = A + 2's\ complement(floor13 * 13)$

Design:

As seen in the algorithm above, we need to perform 4 shifts, 3 adding operations and one subtraction. However, both the shifting and subtraction operations can be achieved using adder circuits. Shifting is done by connecting the correct bits to the adder input, and subtraction is done using 2's complement. So this modulo circuit is done using 4 adding operations.

To find the maximum number of bits we consider the largest possible 6-bit input namely, 63.

$63_{10} = 111111_2$

After step(1): We will have shifted it by 2 and added it to its self.

$1111\ 1100_2 + 11\ 1111_2 = 1\ 0011\ 1011_2 = 63*5 = 315_{10}$

→ step(1) produces a 9-bit output.

After step(2): when 63*5 is shifted to the right by 6, we get a floor13 of 100_2

After step(3): $10\ 0000_2 + 1\ 0000_2 = 11\ 0000_2$

→ step(3) produces a 6-bit output

After step(4): $11\ 0000_2 + 100_2 = 11\ 0100_2$ = floor13*13 = 52_{10}

→ step(4) produces a 6-bit output

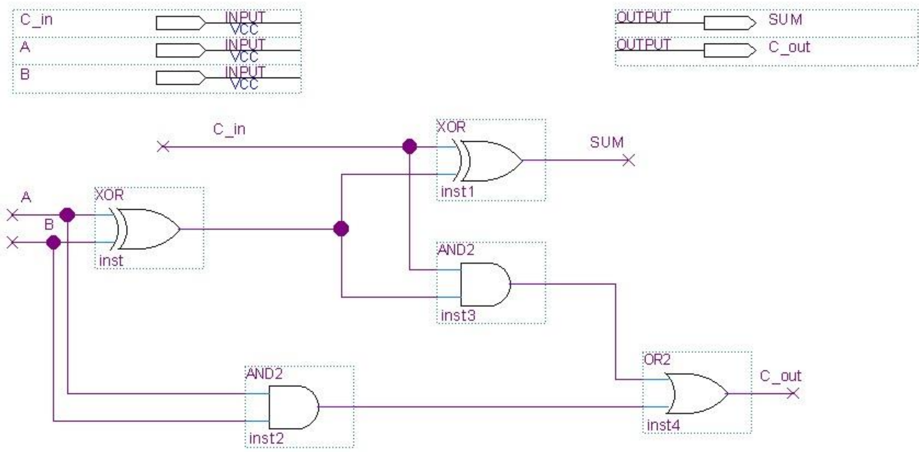
After step(5): $11\ 1111_2 – 11\ 0100_2 = 11\ 1111_2 + 1_2 + 00\ 1011_2 = 100\ 1011_2$

Since this is a 2's complement operation we keep 6-bits only. However the largest remainder we can have for a 6-bit input is 63mod13 = $11_{10} = 1011_2$

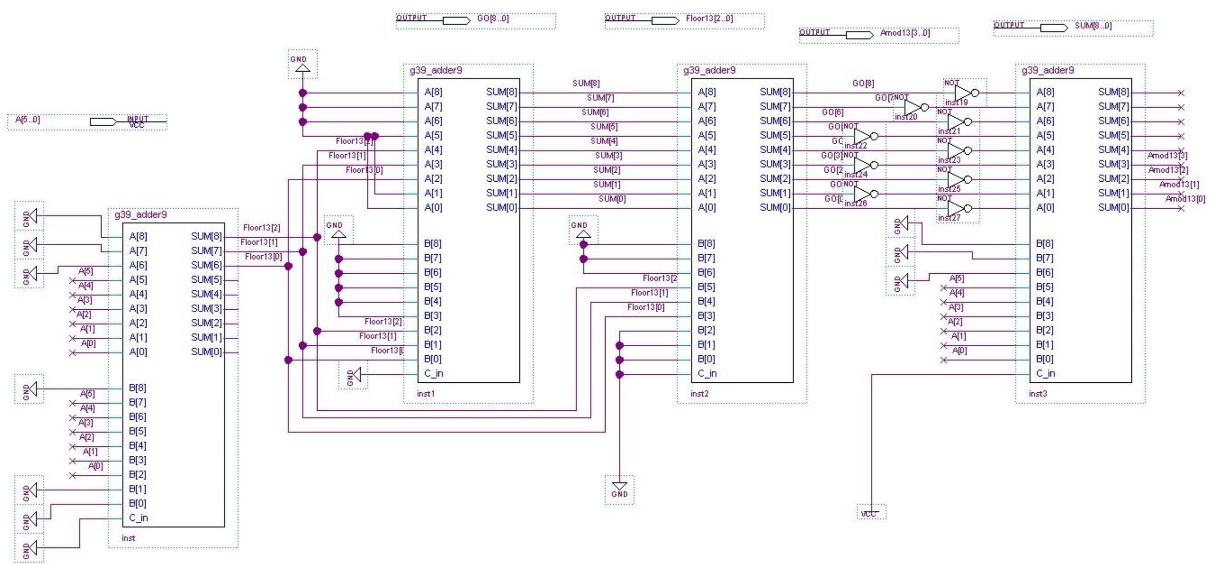
→ step(5) produces a 4-bit output

We can see that the largest number of bits we need is 9 and therefore we will implement a 9-bit adder.

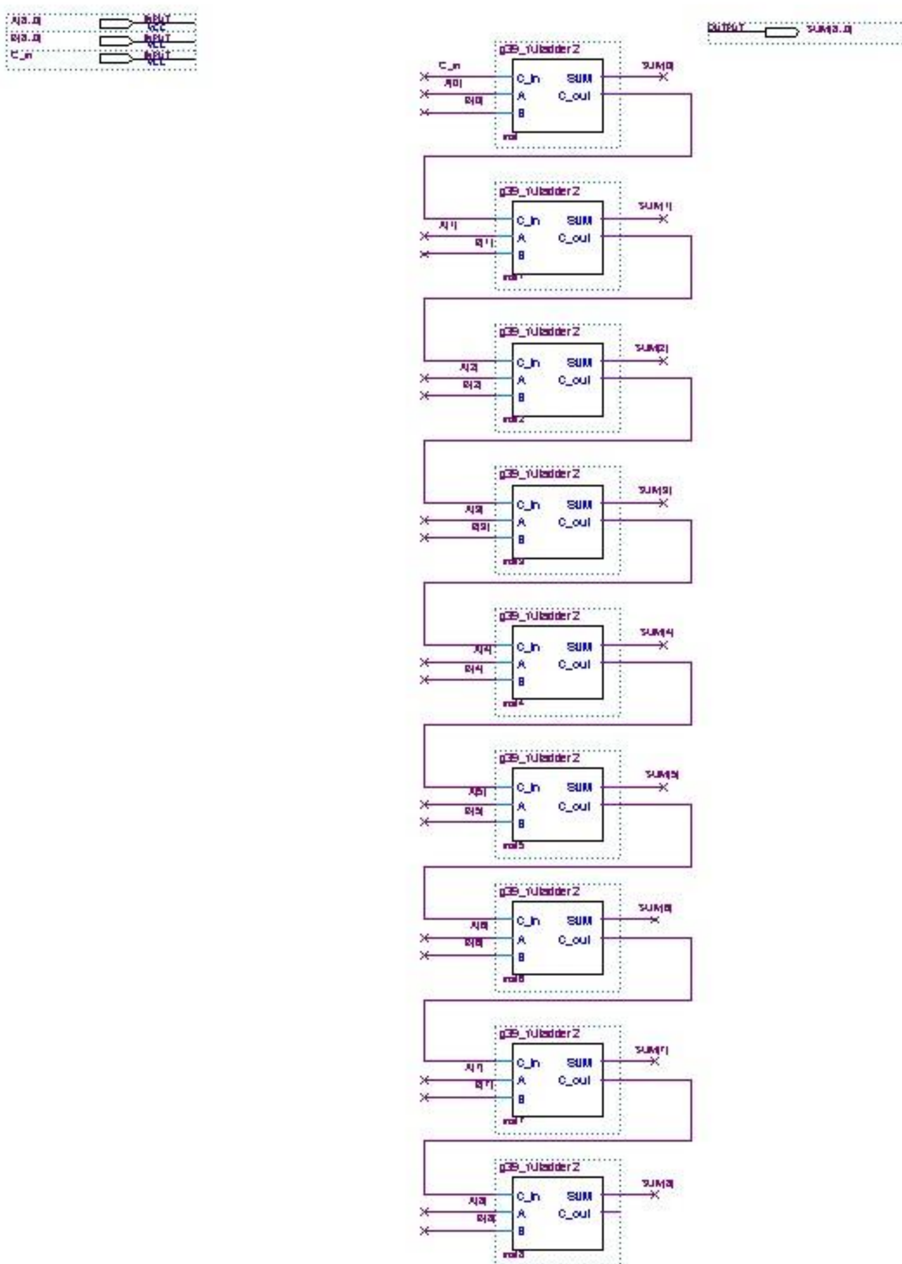
fulladder2:



Modulo_13



fulladder9:

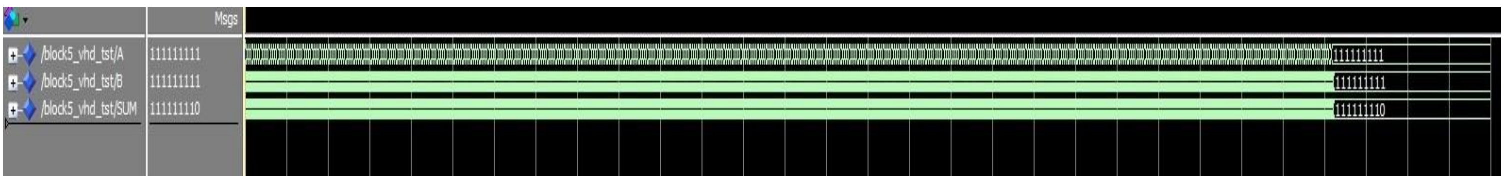


Testing:

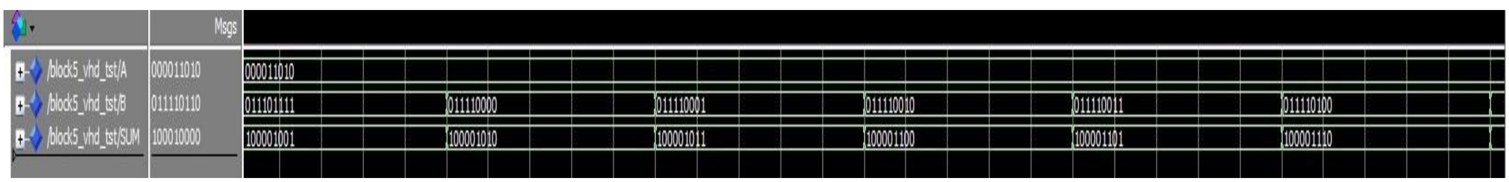
To ensure the circuit works properly, we first tested the algorithm on paper for a couple of cases. Then we carried out a full simulation of the adder circuit. Finally, we simulated the Modulo_13 circuit using different inputs including edge cases.

Below is test cases and simulation plots:

Full simulation of the fulladder9:



A closer look:



Modulo_13 8 test cases: