Philip Nevins
5/5/2022
ENGR 271

Lab 11 - Analog-to-Digital Conversion

Introduction

In the last lab we did, we explored how to convert from Digital to Analog (DAC circuits). In this lab, we explore the opposite, or Analog to Digital conversion (ADC circuits). This is a circuit that converts from an analog signal to a digital signal. There are many ways to accomplish this but we explore flash ADCs and the SAR ADC in this lab.

During this lab, keep in mind these traditional engineering characteristics.

Resolution: The resolution is the number of discrete values that can be produced over a particular analog input range.

Accuracy: Can the ADC convert values without errors? How can this be characterized and mitigated? **This can be mitigated by adding additional circuitry to force the ADC to produce the desired result when a conversion error, known as a bubble error, occurs. We explore this in Procedure 5**

Sampling Rate: The rate at which digital values are sampled from the signal, also known as sampling frequency. If the sampling frequency is greater than twice the highest frequency signal, the Nyguist-Sannon sampling theorem states an exact reproduction of the original signal is possible.

Part A: Flash ADC

Procedures

1. In Procedure 1, we create the Flash ADC given in the lab instructions, shown in Figure 1. It is tested and confirmed to work as expected (shown in Figure 2, from the 74148N data sheet). The test cases we chose can be seen in Figure 3a - c. We chose 3 different test cases that would put at least one High value on the output to confirm all 3 outputs are working correctly.

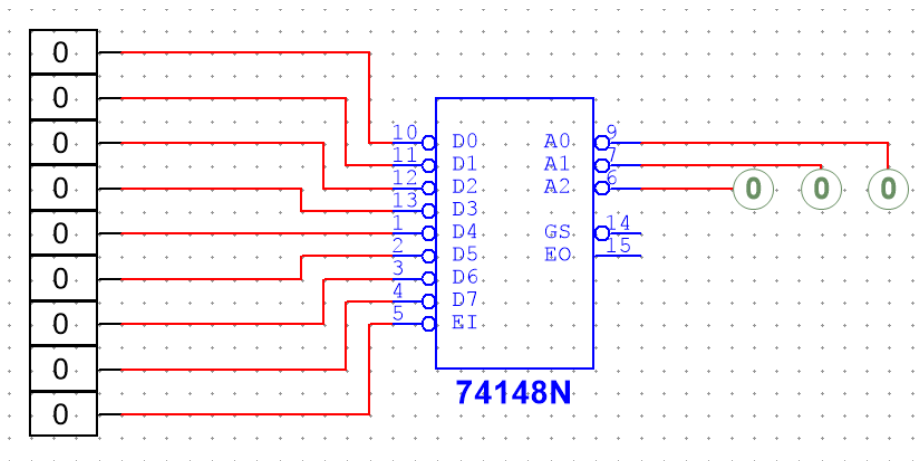
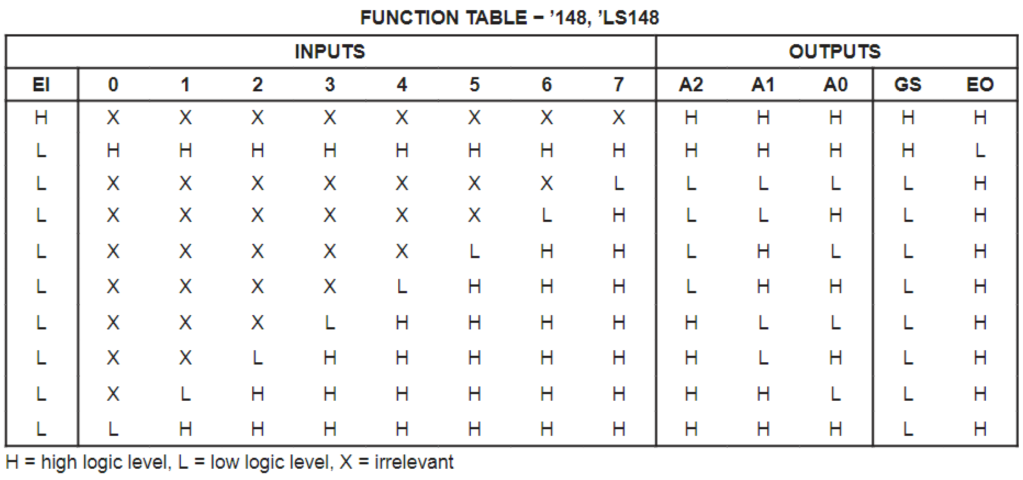


Figure 1: Flash ADC Base Circuit



FUNCTION TABLE – '148, 'LS148

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

H = high logic level, L = low logic level, X = irrelevant

Figure 2: Expected I/O for 74148N

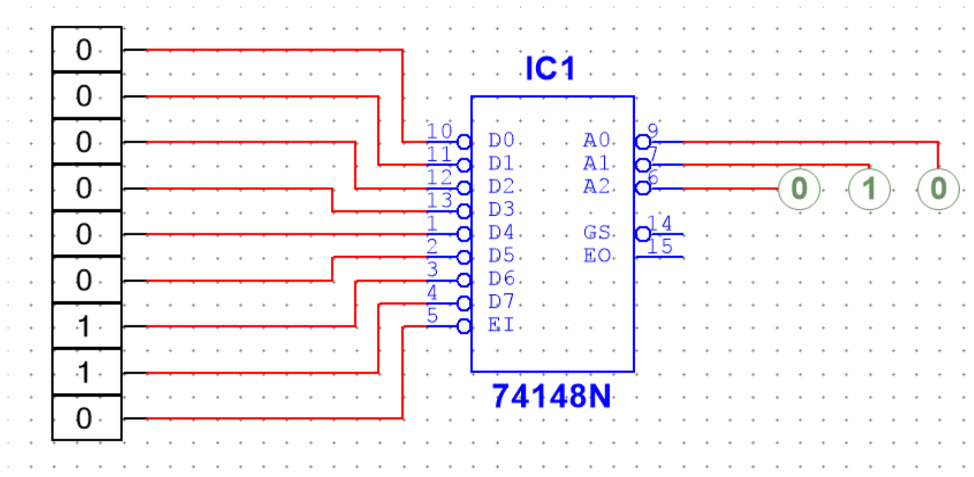


Figure 3a: Test Case 1

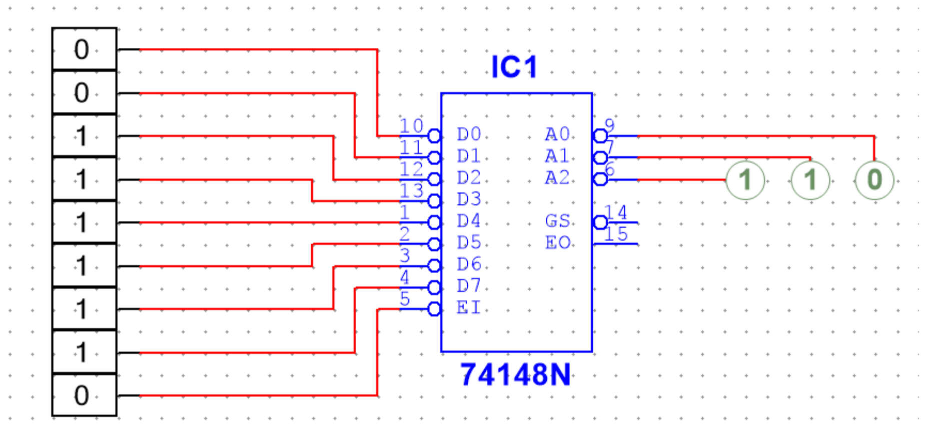


Figure 3b: Test Case 2

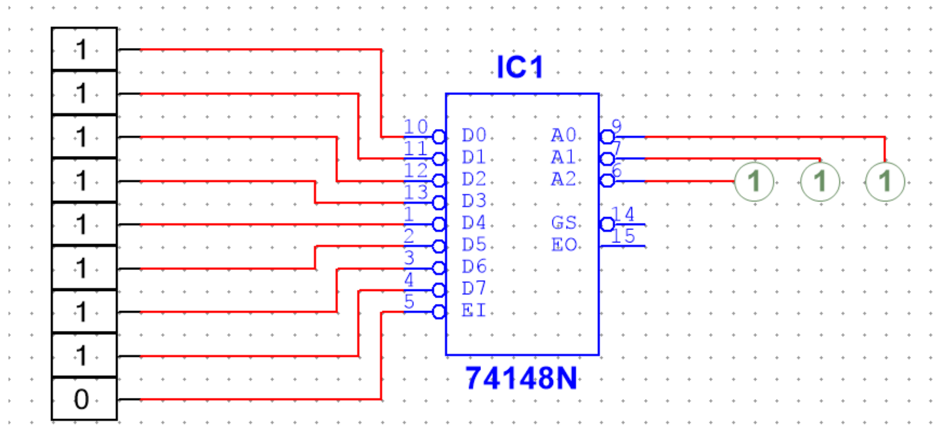
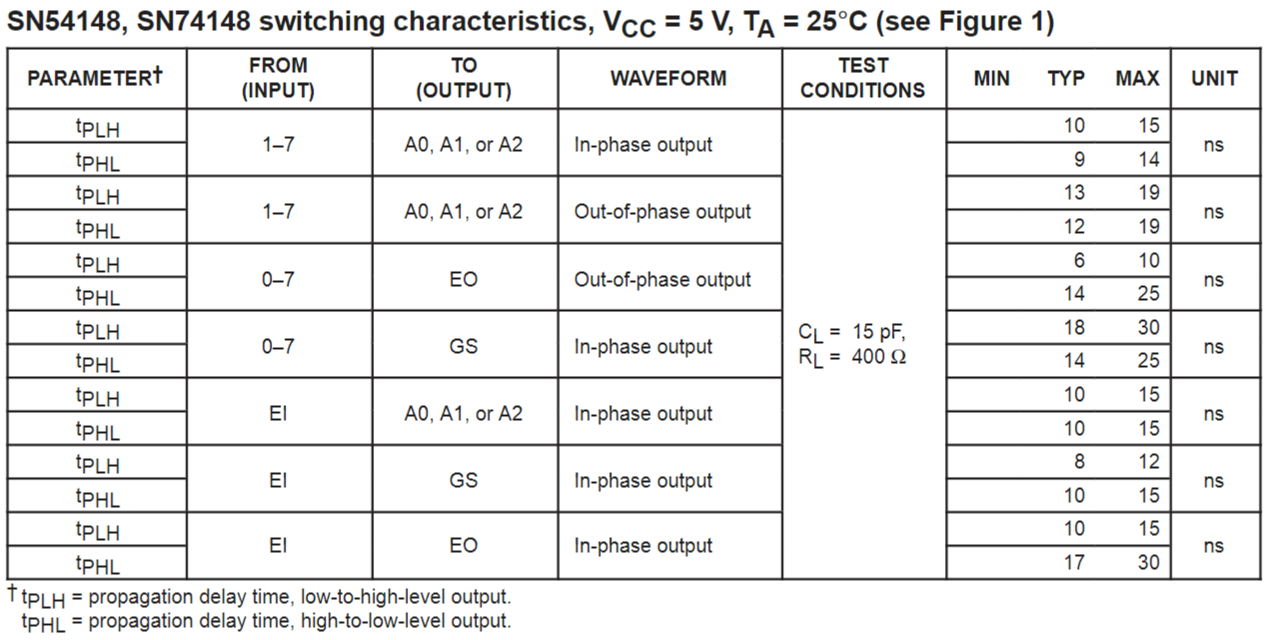


Figure 3c: Test Case 3

2. In Procedure 2, we are tasked with calculating how fast the Flash ADC can work. We use the propagation delay shown in the switching characteristic section, from the data sheet of the 74148N, which is shown in Figure 4. We calculate the Cycles Per Minute (Hertz), which can be shown in Equation 1, which shows our maximum speed is 33.33 MHertz.


SN54148, SN74148 switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 1)

PARAMETER†	FROM (INPUT)	TO (OUTPUT)	WAVEFORM	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	1–7	A0, A1, or A2	In-phase output	C_L = 15 pF, R_L = 400 Ω		10	15	ns
t_{PHL}						9	14	
t_{PLH}	1–7	A0, A1, or A2	Out-of-phase output			13	19	ns
t_{PHL}						12	19	
t_{PLH}	0–7	EO	Out-of-phase output			6	10	ns
t_{PHL}						14	25	
t_{PLH}	0–7	GS	In-phase output			18	30	ns
t_{PHL}						14	25	
t_{PLH}	EI	A0, A1, or A2	In-phase output			10	15	ns
t_{PHL}						10	15	
t_{PLH}	EI	GS	In-phase output			8	12	ns
t_{PHL}						10	15	
t_{PLH}	EI	EO	In-phase output			10	15	ns
t_{PHL}						17	30	

† t_{PLH} = propagation delay time, low-to-high-level output.
t_{PHL} = propagation delay time, high-to-low-level output.

Figure 4: 74148N Data Sheet (Response Time / Switching Characteristics)

$$Speed\ (Hertz)\ =\ 1\ (cycles)\ /\ propegation\ delay\ (seconds)$$
$$Speed\ (Hertz)\ =\ 1\ (cycles)\ /\ 30\ *\ 10^{-9}\ (seconds)$$
$$Speed\ (Hertz)\ =\ 33.33333\ *\ 10^{6}\ (Hertz)$$

Equation 1: Flash ADC Speed Calculation

3. As the number of bits increases, the number of components and amount of space required on-silicon increases exponentially. Approximately, every bit requires 3 more components. As you add more components, you also have to link every 4 together through 4 ANDs and 1 OR gate, which causes an exponential increase. We can see this when comparing the data sheets of the 74148N (8-bit) to the 74147N (10-bit input).
(For reference, https://pdf1.alldatasheet.com/datasheet-pdf/view/27389/TI/74148.html)

4. As we increase the bits, other complications we can expect is the reduction in the reliability due to the resolution being higher. When we have significantly higher bits, the difference in the discrete values that can be produced is significantly smaller, which can cause errors known as bubble errors. We would also expect power requirements to increase, which can be an issue, depending on our design parameters. When increasing the number of comparators in the design see in the next Procedure (Figure 5), they also increase exponentially. for 10-bits, we would need about 1,000 components.

5. In Procedure 5, we address the issue that can arise, known as bubble errors and we add in additional circuitry (comparators). This can be seen in Figure 5, and Figure 6a - d shows our test cases to confirm the ADC still works properly. We get all expected values from the 4 test cases. When we have 5V (max) on the variable input, we would expect 111 (15) as the output. When we have 0v, we would expect 000 (0). When the comparator goes up by 0.5V each click, we expect 0.5v = 001 (1), 1v = 010 (2), 1.5v = 011 (3). As the voltage gets higher, the rounding goes up instead of down, which is why we get to 5v = 111 (7).

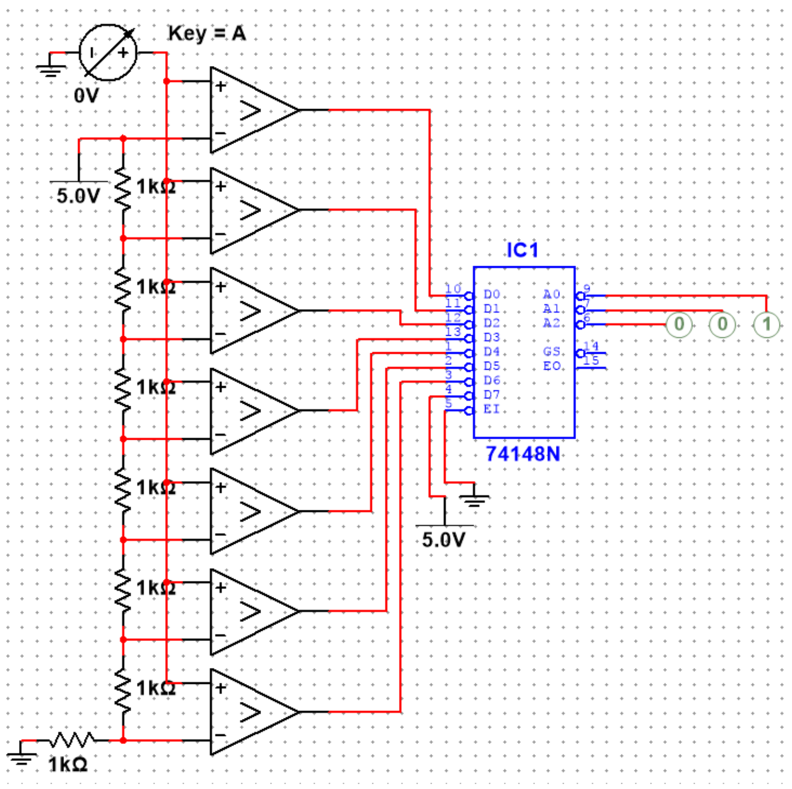


Figure 5: Flash ADC with Comparators

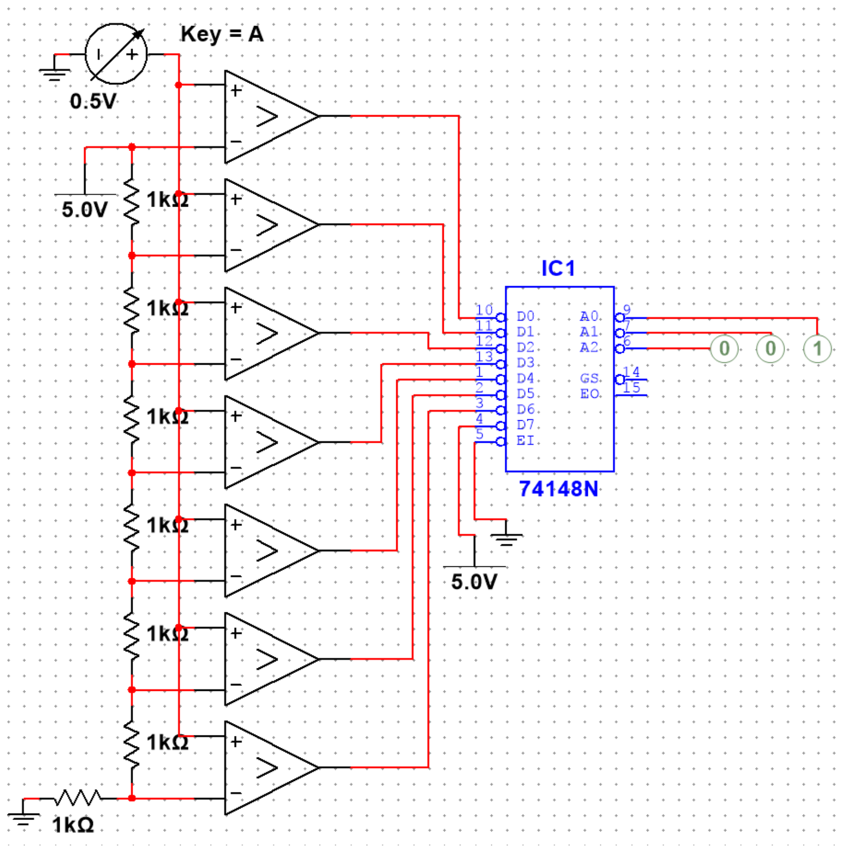


Figure 6a: Test Case 1 (0.5v = 001)

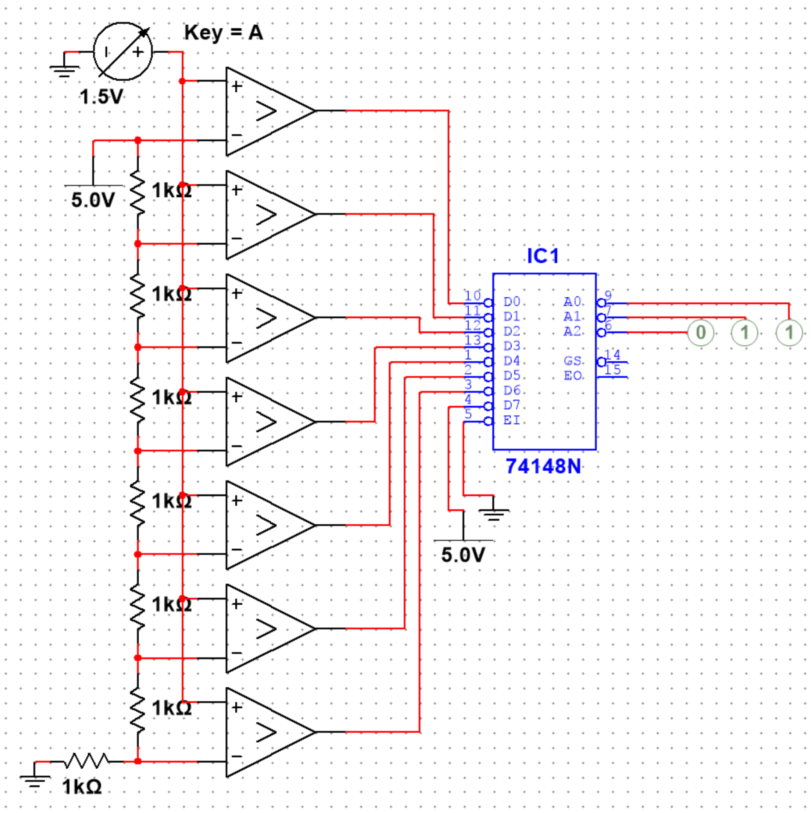


Figure 6b: Test Case 2 (1.5v = 011)

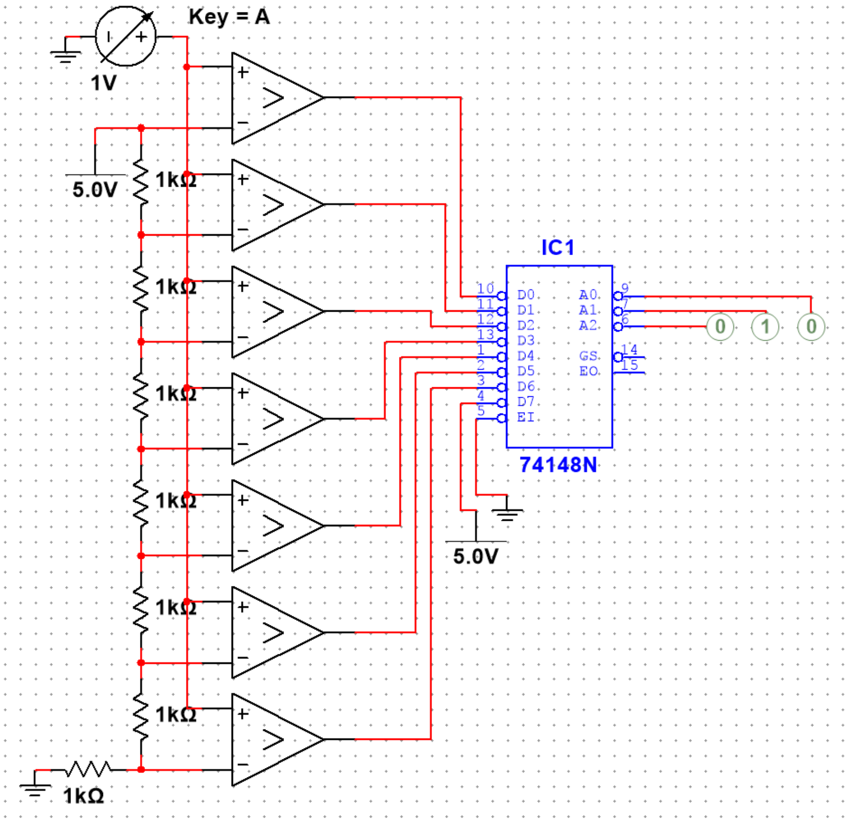


Figure 6c: Test Case 3 (1v = 010)

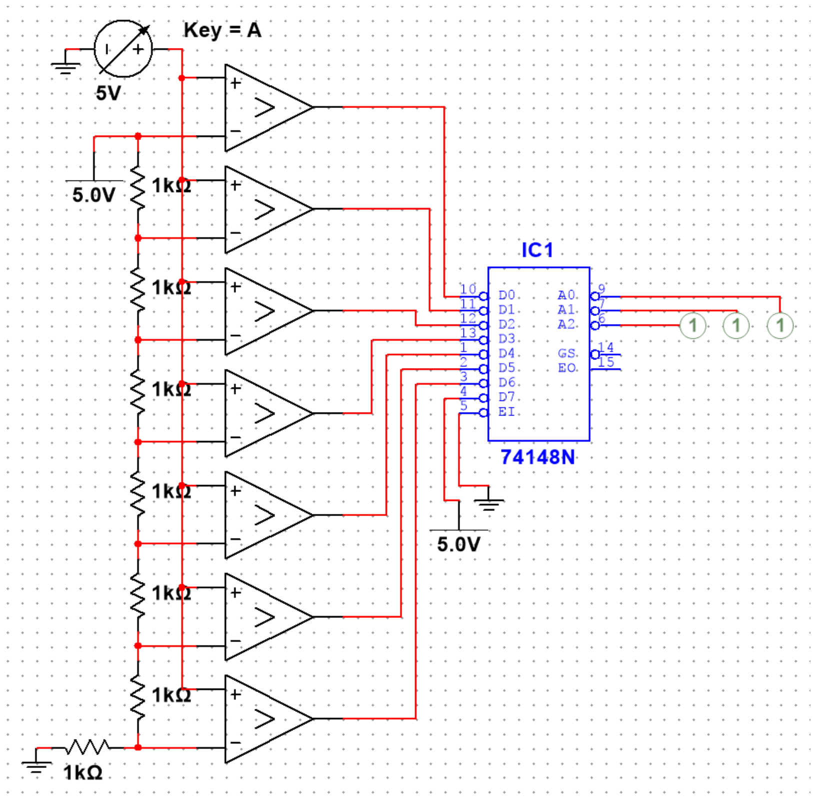
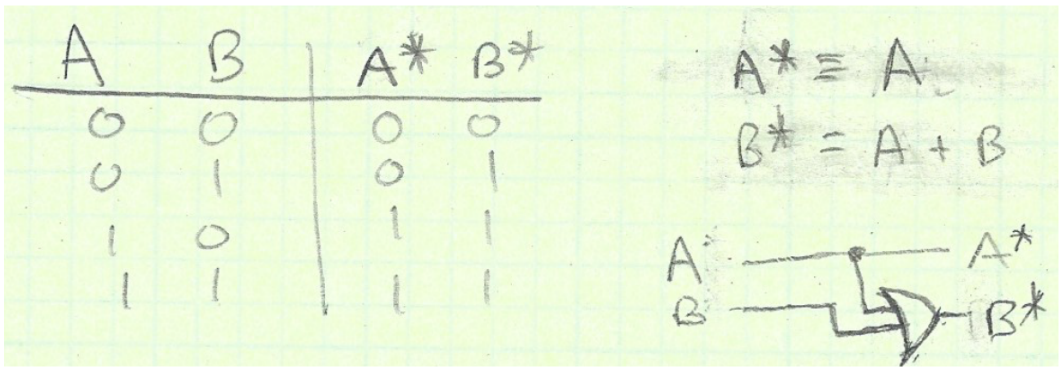


Figure 6d: Test Case 4 (5v = 111)

When the number of comparators increases, the difference between two neighboring comparator input values decreases; if the comparator malfunctions, this is when a bubble error occurs. In Figure 7, we can see the design process done to address the bubble error. In Figure 8, we can see the additional circuitry added. We also induced a bubble error by switching the inputs of 2 of the comparators. This can be seen in Figure 8, on the left most set of digital probes. Then the digital proves after the additional circuitry shows the bubble error has been fixed. I attempted to have the additional circuitry fix the bubble error to the correct value expected but my original design failed, so I have just included the design that fixes the bubble error without accounting for having the correct digital output.



A	B	A*	B*
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	1

$A^* = A$

$B^* = A + B$

Figure 7: Design process to fix bubble error

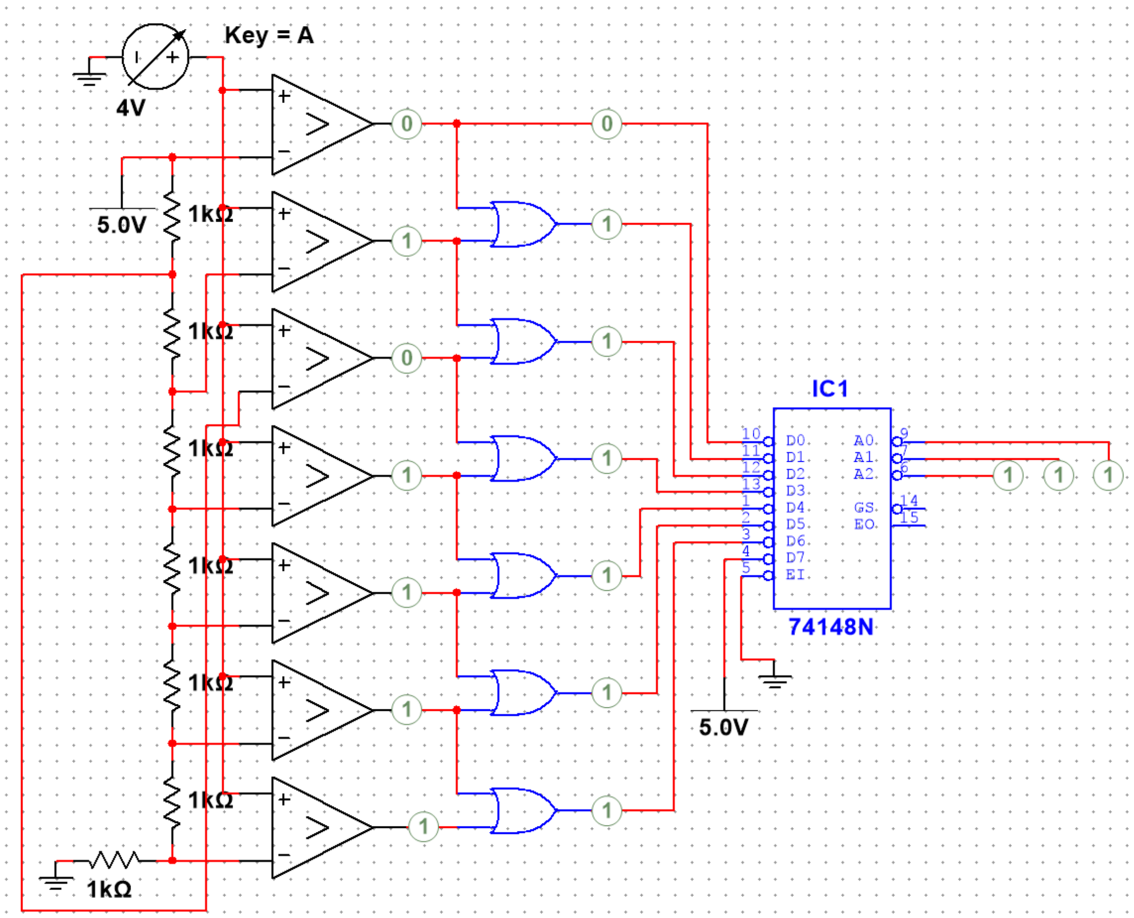


Figure 8: Flash ADC with comparators and bubble error fix

Part B: Successive Approximation Register (SAR) ADC

Procedures

6. In Part B, we are tasked with building a Successive Approximation Register (SAR) ADC. This is one of the other forms of an ADC that are possible to use. This one uses a counter, a 3-8 line decoder, a shift register and some D-flip flops. We will explore how this works in the later Procedures. In Figure 9, we can see the fully built circuit. We then chose 4 different test cases. 0V, 1V, 2.5V and 5V. We chose these values because 0V and 5V are expected to be 0000 (0) and 1111 (15) respectively. Then we can use mathematics to calculate the expected values at 1V and 2.5V.

$$Expected\ Value\ @\ 2.5V\ =\ 15\ /\ (\frac{5}{2.5})$$

$$Expected\ Value\ @\ 2.5V\ =\ 7.5\ \Rightarrow\ rounded\ down\ to\ 7\ (decimal)\ \Rightarrow\ 0111\ (binary)$$

We can also do the same calculation for 1V, which yields 3 $(decimal)$ $\Rightarrow$ 0011 $(binary)$
In Figure 10a - d, we can see we get all of our expected values and the circuit is working properly.

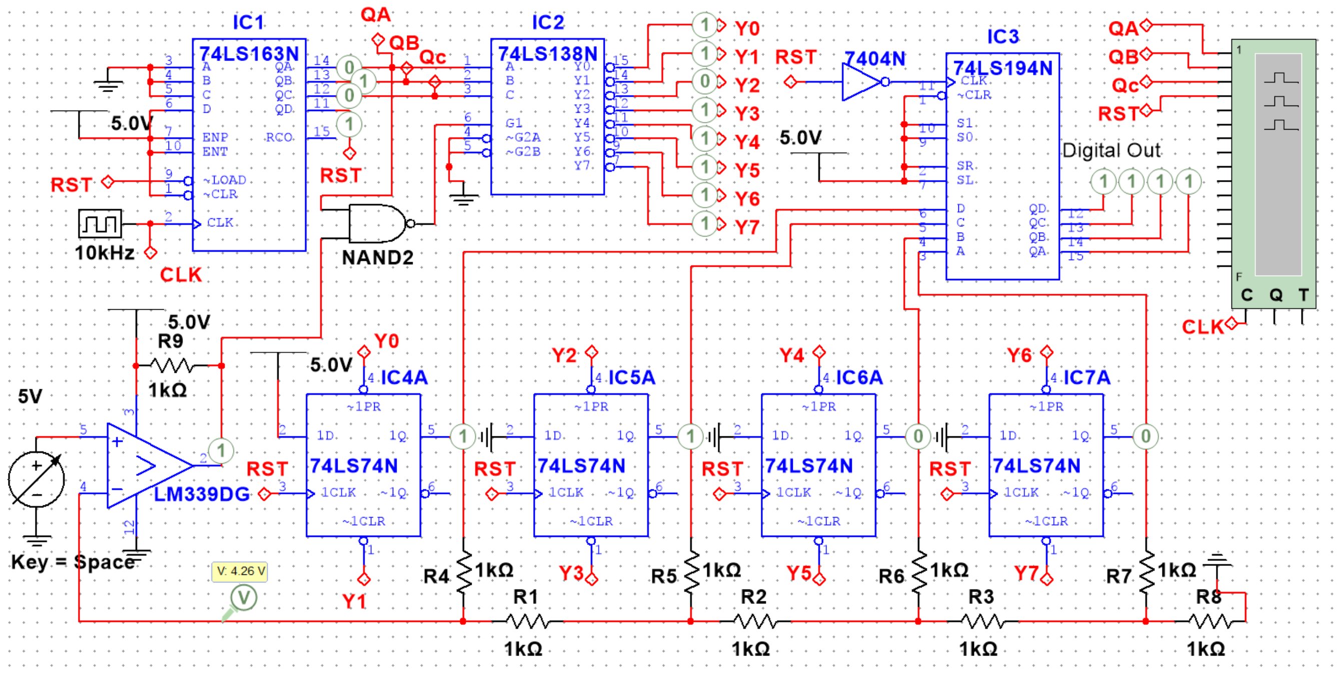


Figure 9: SAR ADC Base Circuit

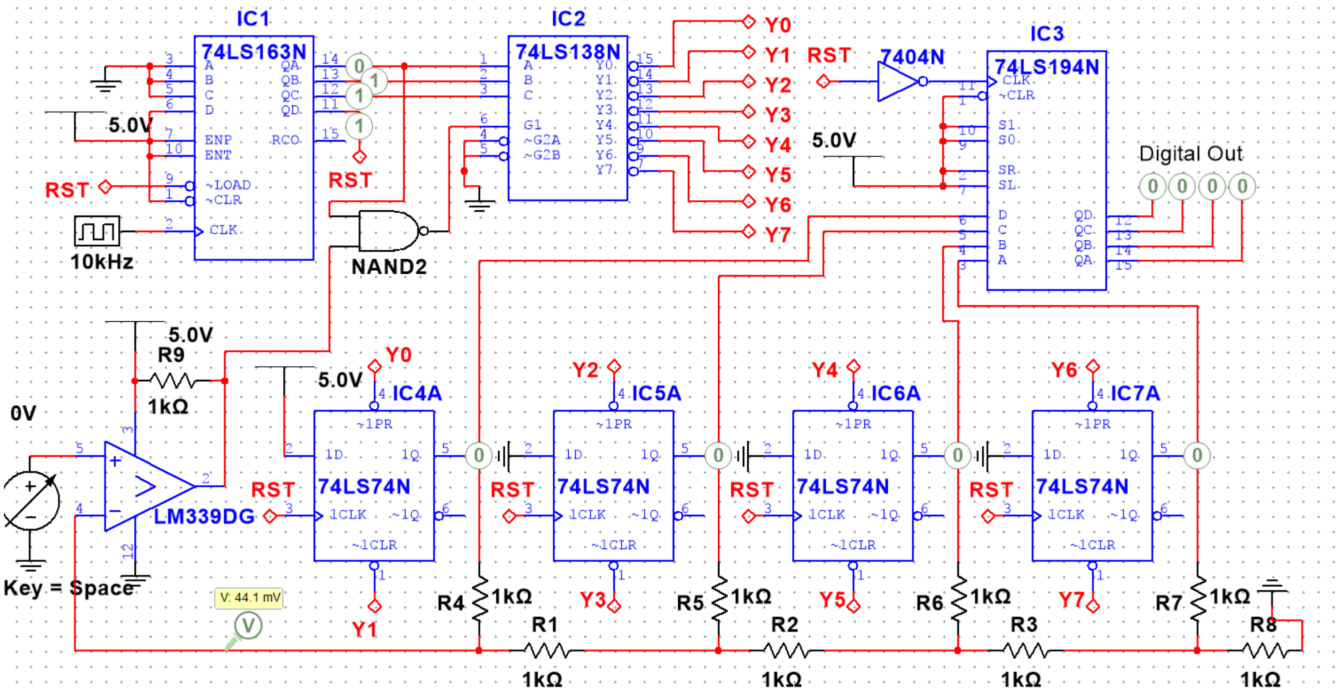


Figure 10a: Test Case 1 (0v = 0000)

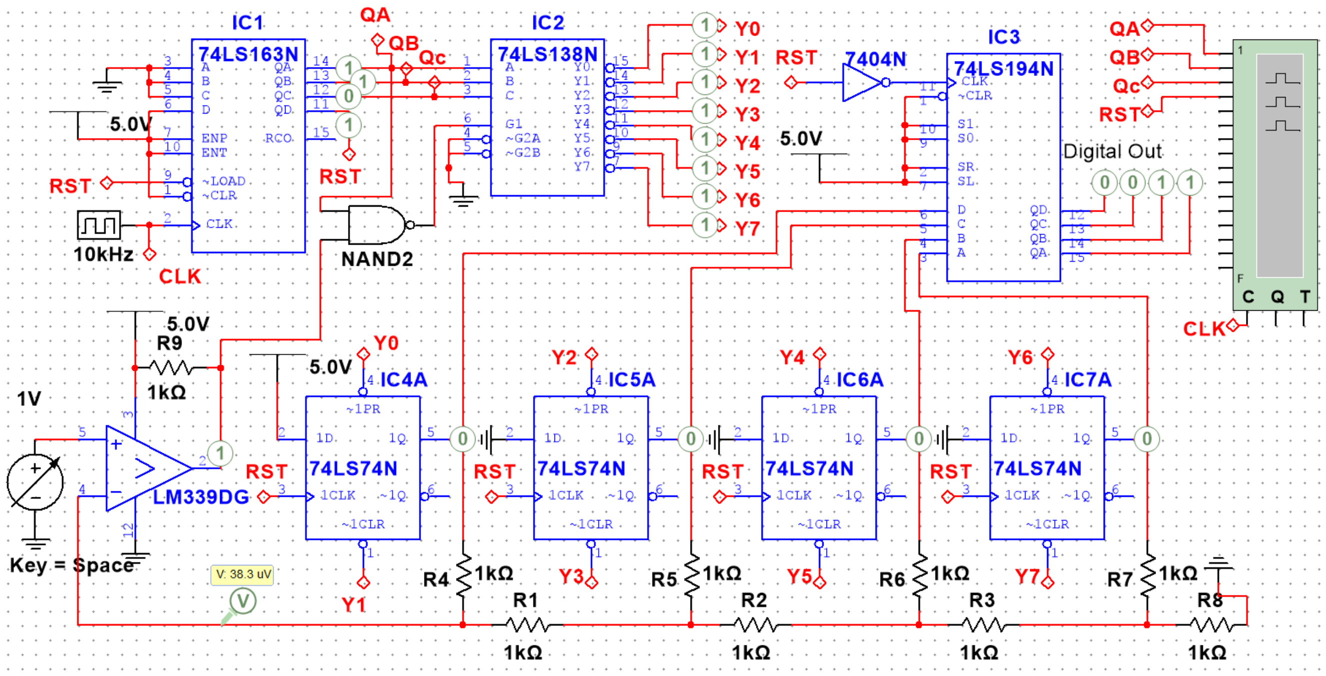


Figure 10b: Test Case 2 (1v = 0011)

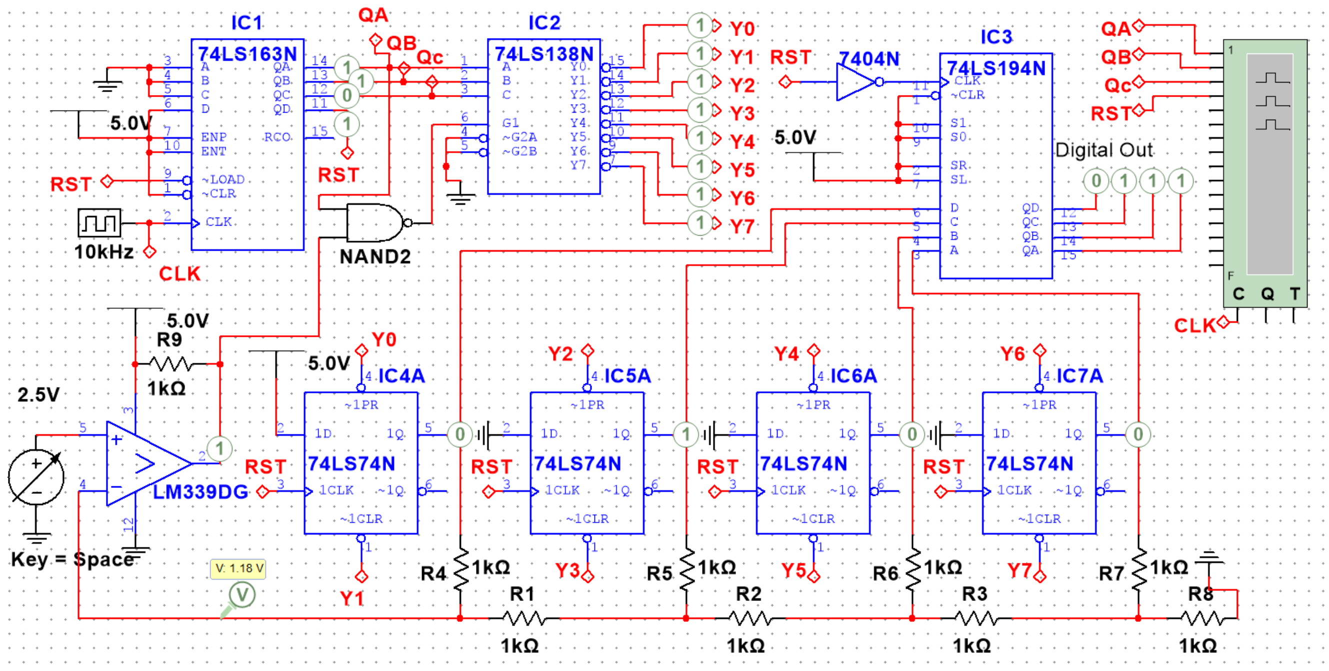


Figure 10c: Test Case 3 (2.5v = 0111)

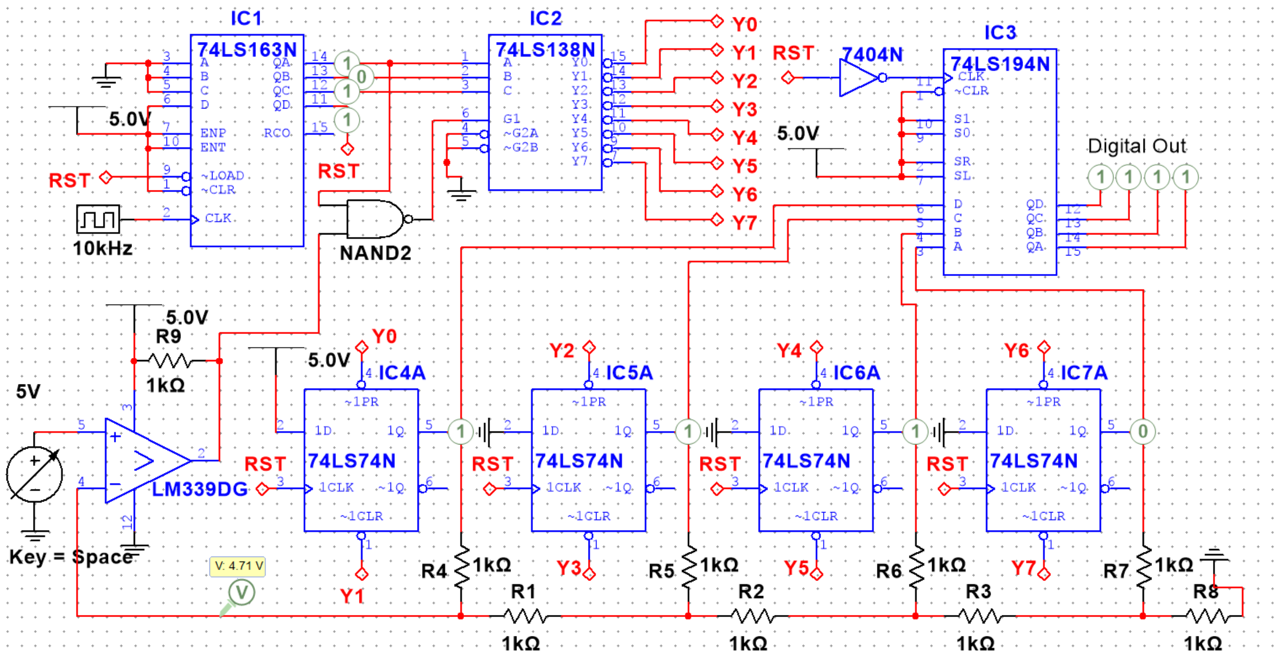


Figure 10d: Test Case 4 (5v = 1111)

7. In this Procedure, we explain what number sequence the 74163N is configured to produce. We can see that it is designed to produce the decimal numbers 0 - 7, or in binary: 000 - 111, then the reset is clocked. This can be seen in Figure 11, on the timing diagram of the 74163N.|

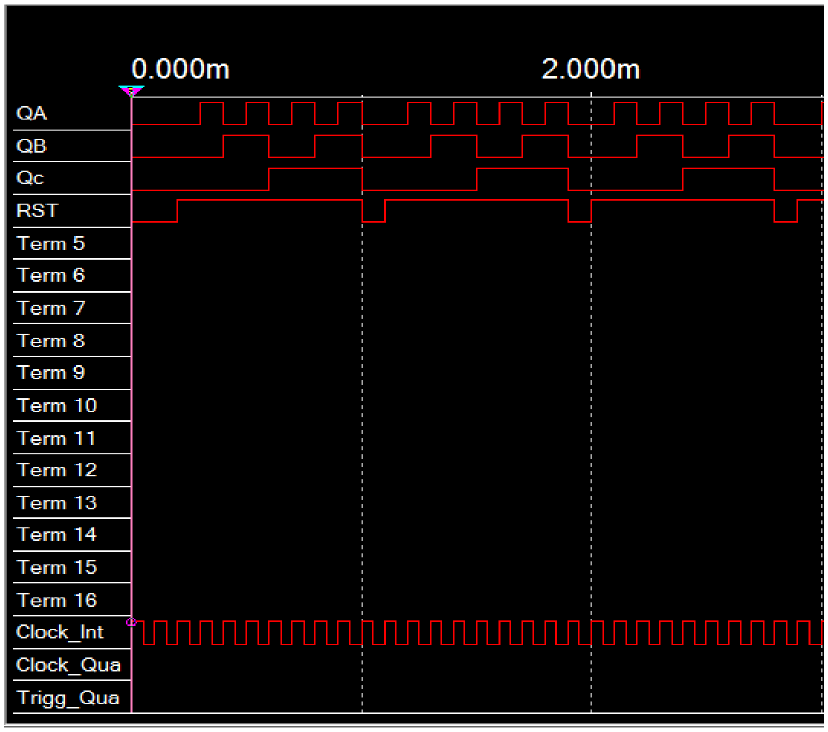


Figure 11: Timing Diagram (74163N)

8. In this Procedure, we explain why there is an inverter on the CLK of the 74194. The inverter is connected to the RST so that the 74163 has time to cycle through its count, and only at that point will the values from the DFFs be passed to the output. Without this, we would get scrambled conversions and lots of errors.

9. The 74163 is a counter. It feeds a 3-8 decoder (74138) that feeds into a shift register (74194), which sets the greater or lesser than toggle on the DFFs. Then the DFFS are fed by the 3-8 decoder value. The DAC on the bottom is fed the value from the DFFs and is compared to the voltage from the comparator. If it's greater (or 1) it signals the nand to output a 1 at the bottom of the count (Qa=1, RST =0 on 74163)

   Qd is what triggers the shifts into the digital out on 74194. So everytime the 74163 counts all the way up, and doesn't trigger a reset 74138 through the NAND (because comparator says it's not greater than) it shifts another bit on the 74138, moving to the next comparison. But when it is greater than, it triggers 74138 to reset and start over on the comparison and RST turns 0 so it clocks the 74194 to shift in 1 bit.

10. In this Procedure, we compare the performance of the Flash ADC and the SAR ADC. The advantages of the Flash ADC is that it uses less complex components and only uses one IC, making it cheaper to produce and consume less power. The downside to the Flash ADC is it is easy to encounter bubble errors and it uses more comparators compared to the SAR ADC.

    The SAR ADC has better performance and accuracy as you are only using 1 comparator, with 4 DFFs, a counter, decoder and shift register, along with a R-2R DAC. The downside to the SAR ADC is it's more complex, has more power consumption and will cost more to produce.

    When comparing the two types, if you can sacrifice accuracy or have a smaller number of bits to run through the ADC, I would choose the Flash ADC. If you need a large number of bits to run through the ADC or need a high level of accuracy, I would choose the SAR ADC.

Conclusion

This was a very interesting lab, seeing how we convert analog signals to digital. We also utilized a R-2R DAC from our previous lab as well. It was interesting seeing the combination of different circuit architectures to achieve a new outcome. We spent quite a bit of time trying to fix the bubble error in the proper fashion, so if we were expecting a 001111, but got a bubble error of 010111, we would produce the correct expected output. This proved to be quite difficult and in the end we were not able to figure it out. We did achieve bubble error correction though, and this was good practice at troubleshooting issues within circuits and designing fixes with the knowledge we've learned from lecture.