Bill Filipiak
ECE 614
Final Exam

Pipeline ADC Tutorial

Introduction:

This tutorial will cover the design of the pipeline ADC. The pipeline ADC is a very popular design due to its high throughput with limited circuitry. Through simulation, the basic operation will be discussed, as well as some advanced topics such as sample-and-hold design and 1.5 bits per stage.

The tutorial will follow a consistent format. For each subject, a summary will be given to briefly describe each topic. Derivations and equations will be avoided since they can be obtained from Chapter 29 of the CMOS book. A simulation section will follow which will outline a group of simulations that illustrate the topic covered in that section. The simulation name that appears at the beginning of each section points to the simulation file that can be run in LTSpice.

Pipeline ADC Operation:

Summary

The pipeline ADC uses clocked single bit stages to achieve high throughput with a low number of comparators. Each stage includes a sample-and-hold, a subtraction stage, amplification by a factor of two, and a comparator. An example of a 3-bit pipeline ADC can be seen in Figure 1.

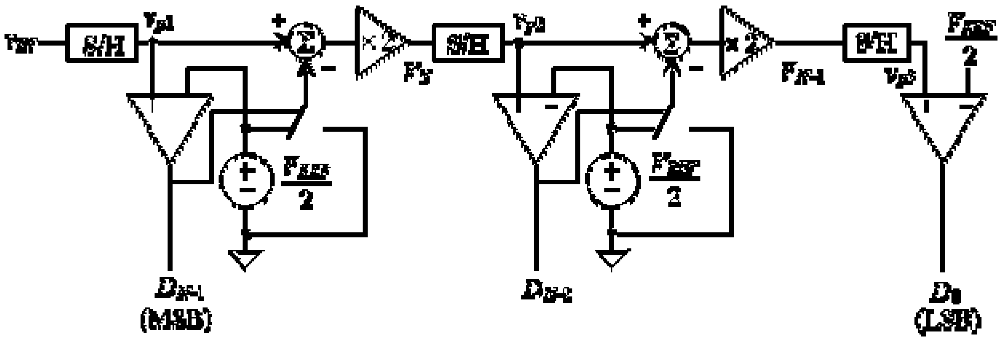

Figure 1: 3-bit pipeline ADC

The pipeline ADC begins by sampling the input on the positive input of the first comparator. The input is compared to $V_{REF}/2$, and the output of the comparator becomes the MSB of the digital output. The MSB of the digital output corresponds to $V_{REF}/2$, so if the comparator output is a digital 1, $V_{REF}/2$ will be subtracted from the original input. Otherwise, the original input will be passed to the amplifier. The so-called "residue" from the first stage is multiplied by two and will be sampled to the input of the comparator of the next stage on the next clock cycle. Since we will be comparing the input of each stage to $V_{REF}/2$, we must multiply the input by two for every stage since each digital bit is worth half as much as the previous bit. It is important to note that the value of the digital output plus the analog value before the amplification by two must always equal the input of that stage. This is the foundation of pipeline ADC operation.

Since each stage of the pipeline depends on the previous stage, there is a latency of N clock cycles for an N-bit ADC. However, after this latency of N clock cycles, a conversion will be completed every clock cycle. This is a significant advantage over other ADC designs. The pipeline is able to complete one conversion per clock cycle with only N comparators for an N-bit design. Of course, flip-flops are required on the outputs of the pipeline so the entire converted word is available on the same clock cycle. Unfortunately, since each stage operates on the residue of the previous stage, errors will propagate through the system.

Simulation

All simulations in this section will use a V_{DD} of 5V, which means V_{CM} will be 2.5V. A 100 MHz clock will be used, and RC delays (100 Ω resistors and 1 pF capacitors) will be added to avoid race conditions and avoid glitching. All components will be ideal to simplify the design and reduce simulation time.

sim1.asc: This simulation shows the operation of a single stage. The input (*vin*) is 0V for the first clock cycle, and is 5V for the second clock cycle. For the first clock cycle, the digital output (*digital_out*) is a 0, and the analog output is 0 (*analog_out*). As was mentioned before, the digital and analog value sum to the original input. For the second clock cycle, the digital output is a 1 since 5V is greater than V_{CM}. This means that we subtract V_{CM} from the original input (*sub*) and multiply by 2 to get the final analog output (*analog_output*). Again, the digital output, which is equivalent to 2.5V, plus half of the analog output, which is 2.5V, is equal to the original input of 5V. Note that the analog output does not appear until the third clock cycle since it is shown after the sample-and-hold of the next stage.

sim2.asc: This simulation shows the operation of a 3-bit pipeline ADC. For illustration purposes, three values will be used for the input: 2V, 3V, and 4.5V. For each stage, the digital output, which appears 3 clock cycles later, plus the half of the analog output (*in2, in3*) equals the input to the stage. As expected, there is a latency of 3 clock cycles before an output appears. However, after this initial latency, one conversion is completed per clock cycle. Notice that, as with any ADC, there will be quantization error associated with the system. An ideal DAC is used to convert the digital output back to analog (v*out*) so it can be compared with the input. This analog output shows the quantization error of the ADC. The quantization error can be as large as a LSB, unless we decide to shift our output by half a LSB to reduce the maximum quantization error to half a LSB. These simulations will not shift the output, so the quantization error can be as high as a LSB.

sim3.asc: This simulation shows the operation of a 3-bit pipeline ADC when the input is a 2 MHz sinusoid. Again, the simulation uses an ideal DAC to convert the digital output to an analog output so it can be compared to the input. The delay between input and output is the latency involved with using a pipeline design. As expected, there is quantization error as in any ADC design since we have a discrete number of output combinations.

Switching Points:

Summary

Analyzing the switching points of each comparator used in a pipeline ADC illustrates the binary weighting necessary for proper conversion. As already discussed, the analog output of each stage is multiplied by two so that it can be compared with $V_{REF}/2$ in each stage. This is necessary because each bit is worth half as much as the previous bit. For example, the MSB is worth $V_{DD}/2$, and the next bit is worth $V_{DD}/4$. This can be understood by simulating the switching point of each comparator.

In addition, it is useful to compare the ideal switching points of each comparator to the non-ideal switching points. The non-ideal switching point accounts for offsets and gain errors of each component. This will lead to the INL and DNL of the pipeline ADC, which tells us how accurate the converter is. We usually require our converter to be within a half LSB for accuracy, so the INL and DNL will tell us how difficult it will be to meet the accuracy requirements. As mentioned, errors will propagate through the pipeline, so the components at the beginning of the pipeline will be more important than the components at the end. This allows us to relax our design of the later stages to save power or area.

Simulation

Again, all simulations in this section will use a V_{DD} of 5V, a V_{CM} of 2.5V, and a 100 MHz clock. All components will be ideal to simplify the design and reduce simulation time.

sim4.asc: This simulation shows the switching point of the MSB comparator. Since we are using $V_{DD}/2$ as our comparison voltage, we expect the switching point of the first comparator to be 2.5V. The simulation sweeps the input (*vin*) from 0V to 5V and monitors the digital output of the first comparator. Note that the sample-and-hold samples the input on the rising clock edge (*vin2*). As expected, the digital output of the comparator switches at $V_{DD}/2$, or 2.5V.

sim5.asc: This simulation shows the switching point of the next stage, which is D_1. Since this stage operates on the residue of first stage, its switching point is dependent on the digital output of the MSB. We expect the second comparator to switch twice as often as the MSB, which is every 1.25V, or every$V_{DD}/4$. Again, this simulation sweeps the input (*vin*) from 0V to 5V. The sample-and-hold for this stage samples the analog output of the first stage (*vin1*), which will depend on the digital state of the first stage. As expected, we see *vin1* climb from 0V to 5V when the input is 0V to 2.5V, and then the *vin1* drops back to 0V and climbs to 5V when the input is 2.5V through 5V. This is a result of the subtraction that occurs when the MSB is a digital 1. This results in switching every 1.25V. Of course, there is a latency of a single clock cycle before this stage switches compared to the input because of the pipeline design.

sim6.asc: This simulation shows the switching point of the LSB. Since this bit is equivalent to $V_{DD}/8$, we expect it to switch every 0.625V. Like the previous simulations, the input (*vin*) is swept from 0V to 5V. The sample-and-hold of this stage (*vin0*) samples the analog output of the previous stage. The subtraction can be seen in *vin0* as it is reset back to 0V and climbs to 5V as the input changes by 1.25V. For the LSB, there is a latency of two clock cycles before this stage switches when it is compared to the input.

sim7.asc: This simulation compares the ideal and non-ideal switching point of the MSB. A 200 mV offset is added to the positive input of the first comparator. As expected, the digital output (*d2*) switches 200 mV before the ideal switching point (*d2_ideal*). As mentioned, we expect errors to propagate through the entire pipeline. However, a comparator offset error will only have an effect on the later stages at the switching point of the first stage, which is at 2.5V. The comparator offset will not change the other switching points. The simulation results show the output of the second stage (*d1*), which switches at the same point as the ideal switching point (*d1_ideal*), except at 2.5V, where it will switch 200 mV early due to the comparator offset. The same can be said for the LSB stage. The offset does not have an effect on any of the switching points except for the one at 2.5V.

sim8.asc: This simulation compares the ideal and non-ideal switching points when the 200 mV offset is moved to the first sample-and-hold. In this case, we still see the 200 mV difference between the actual and ideal switching point of the first stage. However, unlike the previous simulation, the offset of the sample-and-hold propagates through the entire system. This is because the offset of the sample-and-hold affects the residue that is passed to the next stage. Each switching point exhibits the 200 mV difference between ideal and non-ideal.

sim9.asc: This simulation compares the ideal and non-ideal switching points when the 200 mV offset is moved to the second sample-and-hold. As expected, we do not see any difference in the actual switching point for the first stage. The second stage switches 100 mV early, except for the switching point at 2.5V, which is dependent on the first stage, which has no error. Even though the offset is 200 mV, we see only 100 mV of switching error because of the multiplication by 2 that is required for our binary weighting. This also applies for the LSB, which only sees 100 mV of switching error in all switching points except for the one at 2.5V. From these results, it is evident that the sample-and-hold from the first stage is more important than the later stages. In fact, because our ideal gain of the amplifier is 2, each stage's offset will have half as much effect on the switching points as the previous stage. This means we can save power and area by using less accurate designs for the later stages of the pipeline.

sim10.asc: This simulation shows the effect of gain error by changing the gain of all amplifiers to 1.9 instead of 2. Since we are lowering the gain, we expect the actual switching point to occur after the ideal switching point, which is what we see in simulation. Since we have gain error in each amplifier, we see a doubling of the error for the LSB when compared to D_1 since the LSB sees the gain error from both amplifiers. The important point of these simulations is that any error from the components of each stage can have an affect on later stages, but the most significant stages of the pipeline influence the switching points more than later stages.

<u>Sample-and-Hold Design:</u>

Summary

The pipeline ADC requires a sample-and-hold, as well as a subtractor and an amplifier to achieve the necessary binary weighting. Since the subtractor and the amplifier feed into the sample-and-hold, it would be convenient to include the subtraction and amplification in the sample-and-hold design. By adding a capacitor to the standard sample-and-hold design, and connecting the bottom plate to the voltage determined by the digital output of the comparator, we can combine these components in the sample-and-hold. Figure 2 shows the fully differential design.

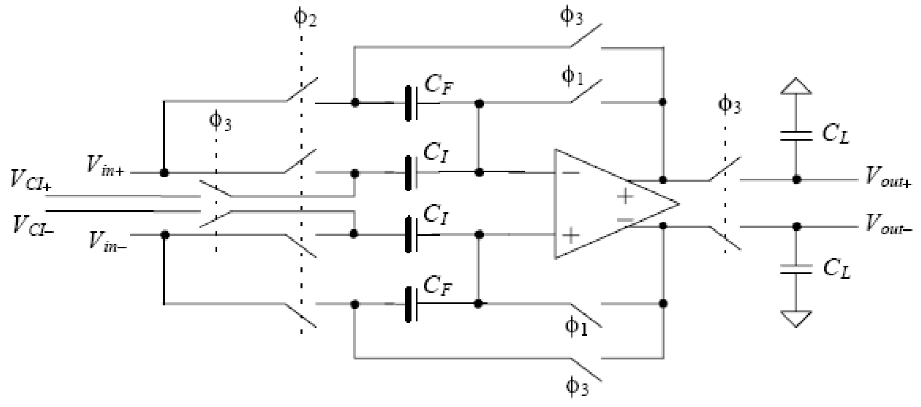

Figure 2: Fully Differential S/H Design

Of course, there are situations where it is not practical to use a fully differential design, but this circuit can also be implemented in a single-ended configuration. It is also important to note that this circuit actually performs the multiplication before the subtraction, which means that the voltage we supply from the comparator (V_{CI}) needs to be twice as big to subtract the correct value. Thus, we must supply V_{DD} to the input V_{CI} if we are using this design to subtract $V_{DD}/2$ and multiply by 2. The derivation of this circuit's transfer function can be found in Chapter 34 of the Mixed Signal book.

Simulation

All simulations in this section will use a V_{DD} of 5V and a V_{CM} of 2.5V. The comparator and switches used in this design will be ideal.

sim11.asc: This simulation shows the operation of the single-ended design with the subtraction and multiplication by 2 integrated into the sample-and-hold. The simulation sweeps the input (*vin*) and monitors the output (*vout*). The voltage on V_{CI} (*vci*) is 0V until V_{IN} reaches 2.5V, where it switches to 5V. This mimics the operation of the switch that is controlled by the output of the comparator. As discussed, the 5V supplied on V_{CI} results in a subtraction of 2.5V from the input. As expected, the circuit subtracts $V_{CI}/2$ from the input and multiplies the result by 2.

sim12.asc: This simulation shows the operation of the fully differential design. The operation is exactly the same as the single-ended design, but we shift the input and output by -2.5V, which is $V_{DD}/2$. To do this, the voltage on V_{CI} (*vcip-vcim*) needs to be -2.5V when the input (*vinsp-vinsm*) is less than 0V, and 2.5V when the input is greater than 0V. This will add $V_{CM}/2$, or 1.25V, when the comparator output is a 0, and subtracts $V_{CM}/2$ when the comparator output is a 1.

1.5 Bits/Stage:

Summary

1.5 bits per stage is a strategy used in many applications today to improve the accuracy of the pipeline ADC and make it less dependent on comparator design. The traditional design uses a single comparator to determine the state of each digital output. This results in two distinct levels, a logic 0 and a logic 1. If we were to employ two bits per stage, we would require two comparators and would have four distinct logic levels (00, 01, 10, and 11). 1.5 bits per stage refers to the case where we use two comparators, but only use three logic levels based on the thermometer code (00, 01, and 11).

The design of 1.5 bits per stage becomes very difficult in the single-ended case, so only the fully differential case will be discussed. The transfer curve of a fully differential ADC looks the same as the single-ended design, except the input and output are shifted by –V_{CM}. The fully differential inputs, swinging around V_{CM}, are sampled in the first stage and appear at the inputs of the two comparators. One comparator compares the positive input with $3V_{CM}/4$, and the negative input is compared to $3V_{CM}/4$. Since the inputs are fully differential, the only possibilities for the digital outputs from these comparators are the thermometer code values. These output combinations are then fed into a multiplexer to determine how much voltage needs to be subtracted from each input to go to the next stage. Again, the rules of the standard design still apply. The digital output plus the half of the analog output of each stage must equal to original input.

Once the conversion is complete, the two bit digital outputs from each stage must be converted back to single bit results. This can be done using something similar to a full adder. The full derivation for the logic can be found in the Chapter 34 of the Mixed Signal book. After reviewing the logic, the first thing to note is that the output has an extra bit. So, in the case of a 3-bit ADC, we would get a 4-bit output. We will ignore the MSB of this result and assume it is an out of bounds condition. Also, after working through the results of this implementation, it is apparent that the output is shifted by V_{CM}-LSB/2, which is a digital 0011. Thus, we must subtract this value from the output of this circuit to determine the final digital output. We do this by adding the two's complement (1101) to the output and ignoring the MSB of the result.

As we will see in our simulation results, the advantage of using 1.5 bits per cell is that errors do not propagate through the system. Since we get two bits out from each stage and reconstruct them into the final digital output, we have a form of error correction that eliminates mistakes made by the comparators. This means that we can design the comparators to less demanding constraints and still maintain the required accuracy. The tradeoff is layout area for two comparators and logic overhead per stage compared to a single comparator. However, since the two comparators can be less accurate, we can save power, and possibly keep the area nearly the same using a less aggressive design. As technology nodes shrink and comparators become less accurate, a design that corrects mistakes becomes essential.

Simulation

All simulations in this section will use a V_{DD} of 5V, which means V_{CM} will be 2.5V. These simulations will use a fully differential design, which means the inputs and analog outputs will swing around V_{CM}.

sim13.asc: This simulation shows the operation of a single stage using 1.5 bits per stage. Since 1.5 bits per stage allows for three different output levels, this simulation steps through each one using inputs of -2V, 0V, and 2V. The input (*vinp-vinm*) is -2V for the first clock cycle, which results in the digital output (*digital_b and digital_a*) of 00. The analog output (*analog_outp-analog_outm*) is 1 V. Since the digital output 00 corresponds to a voltage of -2.5V, the digital output and half of the analog value sum to the original input. This is true for the two other input values as well, so the single stage is working correctly. These two bit outputs will be converted to single bit outputs with an adder once all bits of the conversion are available.

sim14.asc: This simulation shows the operation of a 3-bit pipeline ADC using 1.5 bits per stage. Again, we use the same values as were used in *sim2*.asc, but they have been converted for the fully differential case. The inputs become -0.5V, 0.5V, and 2V. The ideal DAC used to reconstruct the input has also been changed so that the output swings from $–V_{CM}$ to V_{CM} so it can reconstruct the fully differential input. As in the standard design, there is a latency of 3 clock cycles before an output appears. We still see the quantization error seen in the standard design when we compare the input and the output from the ideal DAC. As expected, the digital outputs are the same for this design as they were in the standard design, proving that 1.5 bits per cell is a valid method to employ in a pipeline ADC.

sim15.asc: This simulation shows the operation of a 3-bit pipeline ADC when the input is a 2 MHz sinusoid. In this case, since the input is fully differential, the input signals swing out of phase to create an input signal that swings from 0.5V to 4.5V, similar to *sim3.asc*. From this simulation, we can see that the ADC works correctly when we employ 1.5 bits per stage. We expect the delay between input and output due to the latency in the pipeline design. The first three clock cycles give invalid data since the pipeline has not completed a conversion yet. As expected, there is quantization error as in any ADC design since we have a discrete number of output combinations.

sim16.asc: This simulation emulates a comparator mistake using the standard design with one comparator per stage. In order to do this, an offset of 2.5V was added to the input of the first comparator, which forces it to make the wrong decision for all voltages lower than 2.5V. As the waveform shows, the errors propagate through the system and the output is 2.5V for any input below 2.5V. This is because the comparator makes the wrong decision and attempts to subtract 2.5V from the input and pass that value to the next stage. However, the since the input is less than 2.5V, the output to the next stage will be clipped by ground. Since ideal components were used in simulation, the voltage actually goes negative, but this is not possible in this design since it is not fully differential. However, the final digital outputs in this simulation are the as they would be if the analog output were clipped by ground. It is apparent that errors propagate through the system with this design and the output is invalid if a comparator makes a mistake.

sim17.asc: This simulation illustrates the "error correction" abilities of 1.5 bits per stage. Again, the simulation setup forces the first comparator to make a mistake. Since we are using 1.5 bits per stage, the input only needs to shift by 1.25V in order to move to the next digital output. To do this, offsets of opposite polarities with a magnitude of 0.625V were added to the fully differential inputs. Unlike the standard design, the final output of the ADC is still correct, even though the first stage makes a mistake. This mistake does cause the output of each stage to become more negative, but since we are using a fully differential design, the output does not get clipped by the rails. The reconstruction of the digital output from the two bit outputs of each stage corrects the mistake made by the first comparator so the final result is the same as it would be if a mistake had not been made. Obviously, this is a huge advantage since the comparators can make mistakes without affecting the final output of the ADC.

<u>Conclusion:</u>

The simulations in this tutorial have illustrated the basic operation of the pipeline ADC. The pipeline ADC uses a sample-and-hold, a subtractor, an amplifier, and a comparator for each stage to perform calculations quickly without using a large number of comparators. It is able to complete a conversion on every clock cycle after a latency of N clock cycles for an N-bit ADC. Since each stage operates on the residue of the previous stage, errors will propagate through the system. In fact, the first stage becomes much more important than the subsequent stages, so less accurate designs can be used on the later stages to save power or area. The simulations in this tutorial have demonstrated this important point.

In addition, some advanced design topics were covered, including sample-and-hold design and 1.5 bits per stage. In the sample-and-hold design, we were able to incorporate the subtraction and amplification into the design by adding an extra capacitor and input signal. The circuit is well behaved, and is very advantageous for layout area. Perhaps the most important topic discussed is 1.5 bits per stage. 1.5 bits per stage is very common in designs today where comparator accuracy becomes questionable due to poor matching. It allows the comparators to make mistakes without affecting the final output. The simulations in this tutorial have shown the basic operation of 1.5 bits per stage, and have also shown examples of its error correcting abilities. This is a very important topic as it allows us to use less accurate comparator designs to save power or area.